

**Natural, Robust, and Multi-Modal Human-Robot
Interaction For Underwater Robots**

**A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY**

Michael Scott Fulton

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy**

Advisor: Dr. Junaed Sattar

Jan, 2023

© Michael Scott Fulton 2023
ALL RIGHTS RESERVED

Acknowledgements

In considering those who have made a contribution to my time pursuing a Ph.D., I am overwhelmed with gratitude for all those who have given me so much. While I am certain that I will forget some of you, I nonetheless must make an effort to thank you all.

First and foremost, my advisor Dr. Junaed Sattar. You were the first person to teach me how to do research, and you gave me the opportunity to continue my academic career through graduate study with you. Without your influence in my life, I would be working in computer security in Rome, NY and would have far less excitement and passion in my life. Thank you for taking a chance on a kid with low grades who just wanted some easy credits, giving me space to grow into a successful researcher, and for always going to bat for me when I needed you to.

Secondly, my thesis committee: Dr. Daniel Keefe, Dr. Andrew Lamperski, and Dr. Feng Qian. Your input throughout my preliminary, proposal, and defense process has been helpful in shaping my abilities as a researcher. I appreciate the time you have spent helping me to make my thesis better piece of research, and telling me when I've bitten off more than I can chew.

My constant partner in research, Jungseok Hong. We entered the department together, and while I'm getting out a little bit sooner than you, I know you'll be following me soon enough. Thanks for the late nights, for the weird questions that make me re-evaluate everything I know about English, for the advice about my research, and for the friendship and support you have given me over these years. I always owe you at least one more meal of orange chicken, come on by any time.

My fellow PhD students: Dr. Jahidul Islam, Dr. Jiawei Mo, Chelsey Edge, Karin de Langis, Sadman Sakib Enan, Corey Knutson, Demetri Kutzke, Sakshi Signh. Some of you I had the privaledge to work with on a number of projects, others I only briefly

interacted with. All of you have continuously inspired me with the quality of your work and your passion for robots. Your friendship made these 6 years much more fun than it would have been. I hope that you continue to make robotics a better place through your presence.

The undergraduate students I have had the pleasure of working with: Tanmay Agarwal, Elsa Forberger, Aditya Prabhu, Jonathan Meshesha, Mazzin Khidir, Muntaqim Methaz, Owen Queeglay, Marc Ho, and others. Each one of you afforded me a unique opportunity to watch someone else grow through their own effort on projects that I gave you to work on. Thank you for trusting me, for putting in so much hard work, and for putting up with my haphazard way of getting to the point. I will always remember that without your efforts, much of the work in this thesis would have taken ten times as long.

The various institutions which have provided me funding over the years: the National Science Foundation, the Graduate Assistance in Areas of National Need Fellowship, the MNRI and MNDrive programs. Thank you for your support over these years and for your confidence in me as a researcher.

My therapist, Dr. Samantha Anders, who helped me to bring significantly greater peace to my life in these last few years. Despite the constant stress and anxiety I was experiencing, working with you has helped me to grow into a better version of myself, and undoubtedly helped me to finish this degree.

Michael Forseth, MD. and the medical team at M Health Clinics and Surgery center who made the last phase of my thesis writing pain-free due to the carpal tunnel release surgery they performed on me. Thanks for restoring the use of my right hand.

My wonderful cats, Harriet and Topher. I promise I'll get you more canned food and fancy toys now that I (hopefully) make more money. Thanks for the snuggles, they were wonderful.

My parents, Scott and Becky Fulton. You taught me everything I know about learning, you gave me a loving home to grow in, and you encouraged my interest in computers and tech even when it meant digging broken telephones out of the trash. I love you so much, I hope that you are proud of the work that I have done here, and I hope you know that I still admire you, even as an adult. Thank you for giving me the basis that I started my life on, and thank you for your support as I have outgrown that base and become my own person.

My dear brother, Spencer Ludlam. You have been the most unexpectedly wonderful addition to my life in these years. I never expected to grow so close to you when you came to live with us, but I have treasured our friendship so deeply. You have given me endless entertainment and comfort, and a whole lot of fast food. Thanks for being the best brother-in-law I could ever ask for. Please stop falling asleep when we watch TV.

Last and by far the most important, my beloved wife Sophie Fulton. I've written over 70,000 words in this thesis, but I still don't have the words to express how much you mean to me, and how much your support has kept me afloat over the years. Thank you for working hard night shift jobs while I learned to build robots, thank you for telling me to say no to people, and thank you for always being with me, no matter how rough the time is. Thanks for your patience, we can get started on the next thing now. I love you more than you can ever know.

Dedication

Dedicated to Scott R. Fulton, who introduced me to Java, taught me to use the correct tools and always measure twice, and to Becky Ann Fulton née Maxwell, who taught me to read, write, and learn. Together you taught me to love those around me, to learn for the fun of it, and to seek the truth.

This thesis was written for my family: Maxwell and Fulton generations past, my parents and siblings, and the family that Sophie and I build together every day.

Abstract

In the mid-twentieth century, robots began to swim in the oceans, lakes, rivers, and waterways of the world. Over the seventy years that have passed since then, autonomous underwater vehicles (AUVS) have slowly been evolving, becoming smaller, more intelligent, and more capable. As they have begun to be deployed in a wider variety of locations and for increasingly complex purposes, excitement over the idea of a collaborative AUV (co-AUV) has begun to grow, with the continued development of the field. Now we stand upon the cusp of a revolution in the world of underwater work. Thousands of divers the world over could be aided in their work by a co-AUV in the coming years, helping humans to better understand and protect the critical water resources of our planet. However, for this dream to come to fruition, these co-AUVs must be capable of natural, robust communication, rich and accurate perception of their human partners and adaptive operation in an ever-changing environment. Though researchers have been making steps toward this goal, this thesis marks a new stage in the development of the co-AUV.

In the following chapters, we present three novel methods of communication, two state-of-the-art perception capabilities, a new capability for diver approach, a new methodology for gestural AUV control, a modular software ecosystem for UHRI, and an adaptive communication controller. Additionally, seven human studies evaluating these systems are presented, five of which were conducted in underwater environments with an unprecedented number of participants. The communication methods presented in Part I are a new direction for the field, emphasizing non-text communication which is easily perceived at a distance, natural and intuitive design over information complexity, and introducing new vectors of communication using motion and sound that have not been previously studied underwater. The perception methods of Part II are more traditional, but push the boundaries of previously developed capabilities in numerous ways: developing a new capability in terms of diver motion prediction, creating a method for estimating the relative distance to a diver using only monocular vision, and creating reconfigurable and dynamic gestural control in a way that has not previously been attempted for AUVs. The capstone of the thesis in Part III is the PROTEUS underwater

HRI software system, which could serve as a foundation for a great deal of future research, as well as the first adaptive communication system for AUVs, ACVS. ACVS uses the perception capabilities presented in Part II to determine which of the communication vectors introduced in Part I should be utilized given the context of an interaction, with all of the components implemented within the PROTEUS framework.

The research contained in this thesis is highly multidisciplinary, encompassing interaction design, software development, hardware fabrication, the design and administration of human studies, quantitative and qualitative analysis of study results, deep learning system design, training and deployment of neural networks, robot design, and general robotics development. The results of these investigations into UHRI reveal an exciting potential for the field. Nearly every method presented in this thesis has achieved sufficient success in testing to indicate that it could be effectively applied in field environments, especially given some further development. The dream of co-AUVs helping divers in their work is already beginning to come to life, and the algorithms and systems presented in this document have brought us ever closer to that goal. The work that is done by divers is critical for human society and the health of our planet's ecosystems and the aid that collaborative AUVs could render in these environments is invaluable, greatly increasing diver safety and task success rates. This thesis provides novel communication methods, a new state of the art in diver perception, an adaptive communication system, and a software architecture that ties them all together, improving the flexibility and robustness of underwater human-robot interaction and providing a basis for further development along these exciting avenues.

Contents

Acknowledgements	i
Dedication	iv
Abstract	v
List of Tables	xi
List of Figures	xiii
Introduction	1
1 AUVs, Underwater Environments, and UHRI	7
1.1 The Advent of the Co-AUV	7
1.2 A Brief History of AUVs	9
1.3 Co-AUVs Used In This Thesis	11
1.4 Underwater Research Environments	13
1.5 The Current State Underwater HRI	15
Part I: Natural Methods for AUV-To-Human Communication	20
2 Motion for AUV-To-Human Communication	21
2.1 Background: Motion As Communication	22
2.2 Kineme Design	24
2.3 RCVM Implementations	27
2.4 Study I: Kineme Viability and Learning	30

2.5	Results of Study I	33
2.6	Study II: RCVM Implementation	37
2.7	Results of Study II	41
2.8	Study III: Viewpoint and Content Effects on RCVM	42
2.9	Results of Study III	47
2.10	Conclusion and Future Directions	52
3	Emitted Light For AUV-To-Human Communication	55
3.1	Background: Light-Based Communication	57
3.2	HREyes: Biomimetic Light-Based Communication	59
3.3	Design of Lucemes	61
3.4	Study IV: Communication and Gaze Indication with HREye	64
3.5	Results of Study IV	66
3.6	Conclusion and Future Directions	71
4	Sound For AUV-To-Human Communication	74
4.1	Background: Sound-Based Communication	75
4.2	SIREN: A Low-Cost AUV Audio System	77
4.3	Design of Sonemes	80
4.4	Study V: How Do TTS & Tonal Sonemes Perform?	83
4.5	Results of Study V	87
4.6	Conclusion and Future Directions	94
Part II: AUV Perception of Humans For Interaction		97
5	Evaluation and Improvement of Diver Detectors	98
5.1	Background: Diver Detection In Video Contexts	99
5.2	VDDC: An Order of Magnitude	102
5.3	Neural Networks For Diver Detection	105
5.4	Traditional Accuracy Evaluations	107
5.5	Analysis of Temporal Stability	111
5.6	Conclusion and Future Directions	114

6 Predicting the Future Motion of Divers	116
6.1 Background: Predicting Human Motion	117
6.2 LSTMs For Diver Motion Prediction	118
6.3 Optical Flow Stabilization	121
6.4 Training Methodology	122
6.5 Results	123
6.6 Conclusion and Future Directions	128
7 ADROC: Autonomous Diver Approach Using Monocular Vision	130
7.1 Background: Diver Following and Approach	131
7.2 Designing Diver Approach	133
7.3 Implementation of ADROC	136
7.4 Study VI: Approaching Divers Using AUVs	140
7.5 Results of Study VI	142
7.6 Conclusion and Future Directions	145
8 POSH-G: Dynamic, Reconfigurable Gestures For AUV Control	147
8.1 Background: Gestural AUV Control	149
8.2 Estimating Diver Body Pose Using DeepLabCut	151
8.3 The Protean Gesture Language and The PGR Dataset	158
8.4 Generating Gesture Data	164
8.5 Recognizing Protean Gestures	168
8.6 Results: Accuracy of One-Shot-Trained Recognizers	173
8.7 Conclusion and Future Directions	176
Part III: Adaptive and Reconfigurable Underwater Human-Robot Interaction	179
9 PROTEUS-HRI: Natural, Adaptive, Underwater HRI	180
9.1 Purpose and Scope of PROTEUS	180
9.2 Language Definitions and Common Structures	183
9.3 Implementation of Communication Methods	187
9.4 Implementation of Diver Context Module	200
9.5 Conclusion and Future Research Directions	204

10 Autonomous Communication Vector Selection	207
10.1 Background: Adaptive Communication	208
10.2 Design of ACVS	209
10.3 Communication Policy Design	213
10.4 Study VII: Adaptive Communication Evaluation	216
10.5 Results of Study VII	227
10.6 Conclusion and Future Directions	236
Conclusion	238
Bibliography	242
Glossary and Acronyms	281
Appendix I: Building LoCO AUV	284
Appendix II: Human Study Research Methods	300
Appendix III: Marine Debris Detection	312
Appendix IV: Depth-Based Bayesian Localization	323
Appendix V: Available Datasets	349

List of Tables

2.1	Development of the RCVM language over time.	24
2.2	Per-kineme results from Study II, along with the total average compared to education level 2 in Study I (Section 2.4.1).	40
2.3	Study conditions marked by whether (green) or not (red) they are tested, with participant numbers for each.	45
2.4	Mean and standard deviation of communication system metrics, averaged over all viewpoints and at the ideal viewpoint for each system. Bold values are the best (min/max respectively) mean for metric in group (overall or at EDU).	46
2.5	Mean and standard deviation of communication system metrics, for all evaluated viewpoints. Bold values are the best (min/max respectively) mean for metric in system group.	50
2.6	Mean and standard deviation of communication system metrics, separated by phrase content. Bold values are the best (min/max respectively) mean for metric in system group.	51
4.1	The effect of combined condition/distance on soneme accuracy, calculated using pairwise Wilcoxon rank sum tests, adjusted using the Holm-Bonferroni method.	93
5.1	Comparison between performance on test sets of the VDD- \bar{C} and DDD with training on either train set. This same data can be viewed in Figures 5.5 & 5.6.	108
5.2	Precision and IOU values for each model trained on VDD- \bar{C}	110
5.3	Frames per second for inference.	110
6.1	Inference Time on a Jetson TX2.	126

7.1	The Pseudo Distance (PD) metric based on distance d between a diver and robot.	138
8.1	The dataset configurations used for training DeepLabCut models.	155
10.1	The possible values for the context variables. Distance is the multi-bone pseudoistance described in Section 9.4.3, angle is a centrality ratio for the center point of the diver, and priority and symbol tags are passed with the communication request.	213
10.2	Vector suitability heuristics for the Heuristic and Heuristic_Combos policies. Each context value is multiplied with the others, and the highest overall score is selected. Therefore, a 0 value blocks the vector from being selected, and the higher a context value is, the higher the final suitability score will be.	214
10.3	Combination rules for the Heuristic_Combos policy. These values are added to the combination probabilities calculated for each vector, then clamped to [0, 1.0]. A positive value (blue) increases the likelihood of use, a negative value (red) decreases the likelihood, and N/A (gray) indicates that the vector cannot be combined with itself.	216
10.4	Results of the Oracle Search task per attempt.	229
10.5	Results of the NASA TLX surveys administered after Oracle Search scenarios	234
A1.1	LoCO AUV Specifications.	288
A3.1	Detection metrics in mAP, IoU, and AP	315
A3.2	Performance metrics in frames per second	315
A3.3	Performance metrics in frames per second	320
A3.4	Overall metrics for each combination of the TrashCan dataset and model	320
A4.1	Selected existing localization algorithms	326
A4.2	Lake Bathymetry Information	336
A4.3	Model parameters	337
A4.4	Localization performance evaluation for an AUV.	340

List of Figures

1.1	Divers performing archaeological and scientific surveys underwater. <i>Images courtesy of NOAA and Mael Ballanad via Unsplash.</i>	8
1.2	A sampling of AUVs from the first up to the present day. <i>Images courtesy of copyright holders.</i>	10
1.3	The Aqua2 AUV “exploded” with all of its system annotated. Image courtesy of the Mobile Robotics Laboratory [1].	12
1.4	A CAD drawing of LoCO-AUV with its primary systems annotated.	13
1.5	Divers and AUVs in a variety of field, pool, and simulated environments.	14
1.6	Divers using fiducial flashcards (left) and gestures [2] (right) to control AUVs. <i>Left image courtesy of McGill Mobile Robotics Laboratory.</i>	16
1.7	A diver attempts to read the Aqua AUV’s display, with a closer view of the display on the right (<i>Courtesy of McGill Mobile Robotics Laboratory</i>).	18
2.1	The Aqua AUV in the Caribbean, indicating a K_{No} by shaking its “head”.	22
2.2	The kinemes of the RCVM system demonstrated on Aqua. Note that these kinemes are a newer version, not the version tested in the described studies.	26
2.3	The Aqua AUV simulation using Unreal Engine used for Study I.	27
2.4	The baseline systems used for comparison to RCVM in Studys I and III.	29
2.5	Accuracy of each kineme and luceme from Study I, averaged across all education levels.	33
2.6	Accuracy of the kineme and light-based communication systems compared across education levels. Statistically significant improvements are noted at Edu1 and Edu2.	34

2.7	Operational accuracy of the kineme and light-based communication systems compared across education levels. Statistically significant improvements are noted at Edu1.	35
2.8	Time to answer of the kineme and light-based communication systems compared across education levels. The kineme system is significantly slower at Edu0 and Edu1, but not at Edu2.	36
2.9	A flow chart of Study II's interaction loop, as described in Section 2.6.1.	38
2.10	A graph of the per-kineme accuracy results of Study II, for easier visualization.	41
2.11	The viewpoints captured for this study, illustrated in three examples(a-c) and a diagram(d) showing the location of each viewpoint relative to the AUV.	44
2.12	Accuracy results arranged by viewpoint and system. The y-axis is square root scaled to better display differences between conditions. The dotted line at 7% represents the expected accuracy of a random guess.	48
3.1	The LoCO AUV informs a diver that it is ready to follow them, using the active luceme $L_{FollowYou}$	56
3.2	The evolution of AUV LED communication systems over time. (<i>Figure 3.2a credit: McGill MRL & Ioannis Rekleitis</i>)	58
3.3	HREyes in LoCO AUV, demonstrating $L_{FollowYou}$ out of the water, in a pool, and in a Minnesota lake.	60
3.4	Selected active lucemes, demonstrated in a laboratory.	62
3.5	Selected ocular lucemes, with two examples of directional gaze (angles in Cartesian coordinates).	63
3.6	Comparison between the HREye, OLED, and untrained HREye conditions in terms of all metrics.	67
3.7	Recognition accuracy and operational accuracy for participants trained to recognize active lucemes.	68
3.8	Confusion matrix of participant identifications of active lucemes.	69

3.9	Participant interpretation of gaze lucemes. Points are positioned radially by participant answer and colored by true gaze position. Distance from the center represents participant confidence, with a dashed circle at 5/10 confidence.	70
4.1	A depiction of the two types of sonemes that SIREN can produce, asking a diver for their attention.	76
4.2	Frequency response of the Dayton Audio DAEX25W-8, from the provided specification sheet.	78
4.3	Clustering of scuba diver sign language symbols, sourced from instructional scuba materials.	80
4.4	Selected sonemes, with both Tone and TTS versions. See the accompanying video for recordings of each soneme.	81
4.5	Experimental setup, featuring LoCO at the three possible distances, a participant (P), and the pool inlet (I).	84
4.6	Audiometry data, expressed as the decibel level required by each participant to hear the given frequency.	88
4.7	Internal validity tests for Study V.	89
4.8	Metrics for TTS-Sonemes and Tonal-Sonemes.	90
4.9	A comparison of per-soneme accuracy for SIREN at 1m.	91
4.10	A comparison of per-soneme accuracy for SIREN at 15m and 20m.	92
5.1	Divers operating the Aqua AUV in the ocean. A visual diver detection algorithm is responsible for determining the position of each diver within the image.	99
5.2	Distribution of VDD- \bar{C} (<i>a-c</i>) and DDD (<i>d-f</i>) data.	101
5.3	Distribution of bounding box centers.	103
5.4	An image from VDD- \bar{C} , with and without labels.	105
5.5	Models trained on both datasets, evaluated on DDD in terms of Average Precision at 50% IOU and IOU itself.	107
5.6	Models trained on both datasets, evaluated on VDD- \bar{C} in terms of Average Precision at 50% IOU and IOU itself. Both versions perform well.	108
5.7	The overall average precision at 50 and IOU results for VDD- \bar{C} trained networks on the VDD- \bar{C} test set.	109

5.8	The source of false negative errors in different models.	111
5.9	Measured stability errors for different models.	113
6.1	Stabilization of the bounding boxes over 50 frames for two divers using the dense optical flow based method. Bounding boxes are visualized in the frame of reference of the 50th frame.	121
6.2	Training Pipeline for all LSTM models. VDD- \bar{C} is divided into sequences of 121 frames, each of which is stabilized using the dense optical flow method. The first 120 frames form the training sequence, while the second to 121st frames form the target sequence.	122
6.3	Comparison of metrics over 50 frames of prediction for stabilized and unstabilized Vanilla LSTM and Social-LSTM.	125
6.4	Comparison of future motion predicted by the Social-LSTM and Vanilla-LSTM, both stabilized and unstabilized. LSTM outputs are shown in the frame of reference of the last observed frame. For each row, the last figure represents the true motion of the divers 50 frames in the future (15 frames for row 1).	127
7.1	Two examples of ADROC diver approaches: (a) a diver is in the AUV's field of view, so the AUV simply moves to the diver, (b) no diver is visible, so ADROC begins a search procedure, approaching once the diver is found.	131
7.2	Diagram showing the ADROC algorithm (<i>right</i>) with detail on the state machine (<i>left</i>). Different components of the algorithm (perception, states, approach controller, and conditions) are presented in consistent colors and shapes.	135
7.3	Diver-relative position (DRP) visualizations with a third person view, displayed in the Gazebo simulator and a pool scene. In the DRP visualization, the center of the circle is the target point while the radius represents pseudo distance.	137
7.4	Pool experiments setup: three distances (3m, 6m, 9m), three angles (0°, 45°, 90°). Circles represent diver positions for the experiments.	140
7.5	The success rates and average operation time of ADROC based on the trial, distances, and angles.	141

7.6	(a) LoCO was turned away from a diver at the beginning. Through the SEARCH and APPROACH states, LoCO detected the diver and placed itself at the designed distance from the diver, (b) LoCO was facing a diver and started with the APPROACH state. However, spikes in PD error (bounding box only DRP estimation) caused unstable control and failure.	143
7.7	Diver-relative position (DRP) visualizations for a scuba diver in a pool.	145
8.1	An overview of the POSH-G system, from pose estimation to data generation and tuning recognition systems.	148
8.2	Examples of RoboChatGest [2] and Caddian [3].	150
8.3	Examples of the training and evaluation data of the OceanPose dataset with labels. Note that the videos contained in the evaluation data are entirely separate from the videos in the training data, both in terms of the poses demonstrated and in terms of the divers shown.	153
8.4	Performance of DLC models trained on a variety of dataset configurations, evaluated on their own evaluation dataset. Some data is missing due to unresolved DLC dataset indexing errors.	156
8.5	Performance of DLC models trained on a variety of dataset configurations, evaluated on a common evaluation dataset. Some data is missing due to unresolved DLC dataset indexing errors. Y axis is in square root scale to better display low pixel error values.	157
8.6	Gestures 0-7 gestures of the Protean Gesture Language, demonstrated by a scuba diver. Note that there is also a $G_3Circle$, which is simply $G_1Circle$ three times.	160
8.7	Gestures 8-14 of the Protean Gesture Language.	161
8.8	Gestures 15-21 of the Protean Gesture Language.	162
8.9	The divers and environments present in the PGR dataset. Par-1 and Par-4 also have videos wearing scuba gear. Note that Par-3 is left-handed.	163
8.10	The generation process employed by POSH-G: finding inflection points in trajectories, sampling new points from a distribution around them, and fitting a spline to those newly sampled points.	165

8.11	The preprocessing layers which are attached to the beginning of every network.	169
8.12	An example of the multi-layer perceptron networks trained for gesture recognition, with the hyperparameters that can be changed by KerasTuner.	170
8.13	An example of the convolutional neural networks trained for gesture recognition, with the hyperparameters that can be changed by KerasTuner.	171
8.14	An example of the long short-term memory networks trained for gesture recognition, with the hyperparameters that can be changed by KerasTuner.	172
8.15	An example of the transformer networks trained for gesture recognition, with the hyperparameters that can be changed by KerasTuner.	173
8.16	Recognition results for MLP, CNN, LSTM, and Transformer networks trained on Set A or B, with four pose filter options.	174
8.17	Confusion matrices from the two best-performing network configurations of the POSH-G recognition networks.	175
9.1	The PROTEUS UHRI system. Boxes drawn with dashed lines are not integrated or implemented.	182
9.2	The body distances used to calculate multi-bone pseudodistance. The relevant keypoints are in pink, with the bust distance cyan, shoulder width yellow, trunk length green, and hip width red.	202
10.1	The Autonomous Communication Vector Selection system.	211
10.2	A diagram of the Underwater Telephone task for Study VII. Note the four different distances at which the diver may observe the robot.	217
10.3	A diagram of the Worf scenario of the Oracle Search task for Study VII. The black cylinders are empty containers, while the green cylinders have a target item within them. Uncontained items are not diagramed, as their location is set randomly.	218
10.4	Containers and uncontained objects used in the Oracle Search task. Note the	219
10.5	The AUV-diver communication protocol used for the Oracle Search task.	221
10.6	Depictions of the flashlight gestures used for human input in the Oracle search task.	222

10.7 Perception and understanding of symbols communicated to divers during the Underwater Telephone task.	228
10.8 Summary of the performance of ACVS communication policies in Oracle search in terms of perception, understanding, interaction duration, and average interaction quality.	230
10.9 A per-interaction breakdown of participant perception of communication during Oracle Search.	231
10.10A per-interaction breakdown of participant understanding during Oracle Search.	232
10.11The effects of communication policy on average interaction quality and task score during Oracle Search.	233
10.12Average number of correct containers and incorrect containers collected, along with missing items in oracle search.	233
A1.1 A sampling of early LoCO deployments.	284
A1.2 Original sketch and concepts for LoCO from Michael Fulton’s notebooks, drawn around September 2018. This design evolved into the abandoned submarine concept in Figure A3, but many elements that are core to LoCO were established here.	285
A1.3 The MiniEye stereo vision rig, a precursor to the LoCO AUV.	286
A1.4 CAD renderings showing the development of LoCO AUV.	287
A1.5 A free-body diagram of LoCO with IMU, camera, thruster, and robot frames.	289
A1.6 Power Systems Diagram.	290
A1.7 LoCO’s OLED display showing a menu.	294
A1.8 Gesture detection for a “0” gesture.	295
A1.9 LoCO-eye view of following a diver.	295
A1.10The LoCO AUV in Gazebo simulation.	296
A1.11The assembly process for LoCO AUV, from components to fully assembled (sans batteries). The process takes approximately 2 hours with one person and components pre-fabricated.	298
A3.1 Sample images showing underwater trash made of plastic and metal at different stages of shape and color deformation.	313

A3.2 Example detection results. In images A3.2a-A3.2d it can be seen that while all networks detect some plastic objects, only YOLOv2 (A3.2a) correctly identifies the fish in the scene as bio and Faster-RCNN (A3.2c) detects more individual plastic items than any other network. In images A3.2e-A3.2h, the same evidence of Faster-RCNN's (A3.2g) ability to detect more individual objects can be seen.	314
A3.3 Sampled images from segmentation and object detection evaluations.	317
A3.4 Data split between training (pink) and validation (blue) sets per object for the two versions of the TrashCan dataset.	318
A3.5 Results from Faster R-CNN (pink) and Mask R-CNN (blue) in terms of per-class average precision.	321
A4.1 Visual representation of AUV location in a body of water. The surface and bottom of the body are indicated by the dashed black lines. $L(p_{x,t}, p_{y,t})$: altitude, $p_{z,t}$: depth, $p_{z,a,t}$: range measurement, (Left): LoCO deployed in an open water environment, (Right): LoCO simulated in Gazebo.	324
A4.2 Visualization of raw bathymetry data in tagged interchange file format (TIFF) of Bde Maka Ska. Source: Minnesota Department of Natural Resources. Darker pixels represent locations with deeper depths.	335
A4.3 One example of simulated lakes (Bde Maka Ska) and LoCO AUV. From the top view, LoCO appears as an white dot in the left image due to the difference in actual scale between the lake and LoCO AUV. The right image shows a simulated LoCO AUV in Gazebo.	338
A4.4 Localization performance of the four algorithms for an AUV with an altimeter and depth sensor within Bde Maka Ska using bathymetry data (Start:○ End:✳️).	344
A4.5 Localization performance of the four algorithms for an AUV with linear motions (Best readability at 2x zoom).	345
A4.6 Localization performance of the four algorithms for an AUV with linear motions (Best readability at 2x zoom).	346
A4.7 Localization performance of the four algorithms for an AUV with mixed motions (Best readability at 2x zoom).	347

A4.8 Localization performance of the four algorithms for an AUV with mixed motions (Best readability at 2x zoom).	348
---	-----

Introduction

In the seventy years since autonomous underwater vehicles (AUVs) began to swim in the depths of the ocean, scientists and engineers have been working to improve their capabilities, affordability, and reliability. Due to their tireless efforts and advancements such as improvements in battery technology and the miniaturization of computing hardware, AUVs have evolved into a wide variety of shapes, sizes, and specializations. Small, human-portable AUVs which work alongside human divers are one such variety. While their deployment in real-world work is rare today, the time is not far off in which these AUVs will help divers to inspect and repair underwater infrastructure, conduct biological surveys and sample collections, complete water quality and safety inspections, and much more. A diver assisted by an AUV could potentially achieve far more than they could on their own, provided that the AUV can easily be collaborated with underwater, in real time. By taking advantage of the robot's superior sensing capabilities (sonar, for example), higher swimming speeds, longer endurance, and relative disposability, the diver can task dangerous, difficult, and time-consuming tasks to the robot, allowing them to use their time more effectively. Such collaborative AUVs will be invaluable partners to divers, improving diver safety, efficiency, and precision – provided they can work together with humans.

To work collaboratively, a robot and a human must be able to communicate with one another, understand each other's position, movements, and actions, operate in relation to one another, and adapt to new tasks, environments, and partners. These capabilities are no less important than the other abilities an AUV needs for underwater work: general perception, localization, motion planning, manipulation, etc. In fact, interaction can sometimes help to bridge gaps in those abilities. For example, a robot can follow a human instead of navigating through a map and a human can direct a robot

to objects of interest that it cannot easily perceive. However, unlike the non-interaction capabilities of an AUV, underwater human-robot interaction/collaboration (HRI/HRC) is a severely under-researched topic. This is due in no small part to the inherent difficulties of underwater interaction: wireless radio communications are heavily attenuated, production of human-comprehensible sound is difficult, and visual means of communication are limited by water quality (which is highly variable). Additionally, access to AUVs suited to collaborative work with humans is limited by high prices and a lack of mass-production/open source designs of AUVs. This second barrier is beginning to fade, as the availability of AUVs to agencies and individuals with lower funding is rapidly increasing. As this shift continues, the number of AUVs deployed alongside divers will increase, further underscoring the need for robust methods of interaction supporting collaborative work.

Research on AUV-diver HRI until the present day has laid a basis for our effort to enable underwater human-robot collaborative work through interaction. This work can largely be summarized by following two main threads of research: the efforts of the Mobile Robotics Lab at McGill University [1, 4–6] and its graduates [7], and the work of European researchers associated with the CADDY project [8]. These research groups each introduced an important, HRI-capable AUV (Aqua [1] and BUDDY [9]), utilized hand gestures for human-to-robot communication [2, 10, 11] and digital displays [1, 8] for robot-to-human communication, made use of diver relative positioning [12–14], and operated in an imperative command structure between robot and diver. While the world of AUV research extends far beyond this, and there are researchers outside of these two research genealogies [15] with disparate approaches [16, 17], this is the general form of AUV/diver interaction. This interaction paradigm has become the standard for underwater HRI. It has allowed the development of AUV capabilities and provided a useful base for the further development of collaborative AUVs. However, this interaction paradigm cannot be adapted to the context of interaction (water clarity/visibility, interactant distance, etc.) and thus leads to an operational mode that is far too rigid and inflexible for actual underwater work.

In the current state of underwater human-robot interaction (UHRI), interactions between a robot and a diver consist of hand gestures from the diver answered by a response

on the AUV’s integrated display. This limits the effective range of communication between the two, allowing little to no flexibility in the environments and contexts of their interactions. Additionally, methods for the perception of humans are plagued by low accuracy, poor temporal reliability, and minimal information returned (typically only a 2-D bounding box location), making further inference of diver position, attention, and action difficult. This leads to a status quo for UHRI where divers are limited to starting and stopping pre-programmed missions by gesture or direct control, where all interaction must occur within a very small range of the AUV. In Parts I and II of this thesis, we will address the dearth of robot-to-human communication options and insufficient diver perception capabilities. These problems can all be considered “first-order” HRI research, required to perform research on topics of greater complexity such as achieving effective collaboration through communication, dialogue management, object referencing and disambiguation, etc. Having addressed these issues, in Part III we present a multi-modal communication system for AUVs which dynamically chooses the appropriate method to communicate with a diver, one of the first pieces of “second-order” UHRI research.

Document Organization

Background

Prior to the description of the novel research contained in this thesis, some background is provided, explaining the robot platforms and environments used in the research covered later. An overview of human-robot interaction is also provided with commentary on research outcomes and methodologies, to further demonstrate the ways in which our research advances upon existing approaches.

Part I: Methods for AUV-To-Human Communication

Part I of this thesis is comprised of three chapters, each covering a novel method of AUV-to-diver communication. Firstly, Chapter 2 introduces Robot Communication Via Motion (RCVM), along with three studies exploring the viability of RCVM for use underwater. RCVM is similar to human body language or gestural communication, moving the entirety of the robot to represent phrases or ideas. Chapter 3 discusses light-based

communication for AUVs, beginning with several systems designed as comparison points for RCVM, and culminating in the HREye, a device for communicating with emitted light. HREyes are circular light arrays that can be used to communicate phrases or ideas (like RCVM) but can also be used to mimic eyes, providing gaze cue information. Gaze indication has not previously been studied for AUVs, but Chapter 3 provides a human study of the use of HREye both as an active communication device and for indicating gaze direction. Lastly, Chapter 4 introduces SIREN, a device for sound-based communication. While RCVM and HREye greatly expand the range of interaction for AUVs, both literally and figuratively, both must be visually observed, making their use in vision-deprived environments difficult. SIREN closes that gap by providing audible communication in two forms: Text-to-speech and musical tones. Both forms of audible communication are evaluated in a human study. The human studies in Part I, while small, are the largest evaluations of human-robot interaction conducted in underwater environments (Study II: 8 participants, Study IV: 14 participants, Study V: 12 participants). Taken as a whole, the systems presented in Part I provide a robust set of multi-modal communication methods for AUVs, expanding the options for communication as well as the contexts in which an AUV can effectively communicate.

Part II: AUV Perception of Humans For Interaction

The second part of this thesis contains 4 chapters, each of which covers a different topic in human perception for UHRI. Chapter 5 discusses diver detection – the task of determining a diver’s location in an image – and the contributions our work has made to the state of the art in that field. The next chapter introduces an entirely new perception capability for AUVs, predicting the future position of divers. Our work on that topic adapts methods previously used in terrestrial environments to the unique challenges of underwater robotics. Chapter 7 introduces a method that allows AUVs to autonomously approach a diver, using only monocular visual input. This method, named ADROC, leverages our previous work on diver detection as well as human body pose estimation, combining the resultant information to roughly estimate the distance between the AUV and the diver. Lastly, in Chapter 8 we discuss a one-shot gestural control system called OSG. OSG learns gestures from a small set of examples, based on output from a freshly trained body pose estimation method using DeepLabCut [18]. A

number of publicly available datasets are introduced in Part II, including VDD- \bar{C} , the OceanPose dataset, and the PALG dataset. Considered together, the methods and data presented in Part II represent a significant improvement in perception, diver-relative navigation, and gestural control methods in the underwater HRI field.

Part III: Underwater Human-Robot Interaction

Part III of this thesis presents one piece of second-order UHRI research, along with a software system for UHRI. First, in Chapter 9 we present PROTEUS-HRI, a software infrastructure for human-robot interaction underwater. PROTEUS is comprised of implementations of the methods described in Parts I and II, as well as ACVS. PROTEUS represents the fruits of 6 years of research on this topic and is designed to be highly extensible. While no direct scientific aim is evaluated for PROTEUS-HRI, the system is described in full to provide a baseline for future UHRI researchers. Additionally, we present PROTEUS-HRI in the hope that others interested in this topic will make contributions to further its aims: natural, robust, and adaptive interaction at depth. Finally, in Chapter 10, we present ACVS, or Autonomous Communication Vector Selection: a multi-modal communication system that allows autonomous selection or combination of communication methods. We also present a human case study evaluating the performance of ACVS as compared to a baseline of random vector selection. ACVS is a new capability for AUVs, taking the context of an interaction into account when communicating with a diver and adapting to new situations autonomously. It is also a culmination of the work presented in this thesis, combining all three of the communication vectors introduced in Part I, utilizing perception methods presented in Part II, and all built within PROTEUS.

Appendices

Finally, after the conclusion of the thesis proper, five appendices are provided with extended information. Appendix I covers the design and development of the LoCO-AUV, an open-source, low-cost autonomous underwater vehicle that was used for much of the research presented in this thesis. The following appendix provides recommendations on

human-robot interaction research practices, human study design, protocols, and statistical analysis methods, provided for the benefit of further UHRI researchers. The next two appendices (III and IV) provide brief descriptions of research projects not appropriate for full inclusion in the thesis: detection of marine debris and Bayesian methods for AUV localization using bathymetric data. Finally, Appendix V provides information on the availability of the various datasets presented in this paper, along with commentary on their development and construction.

Collaborations In This Thesis

Due to the multidisciplinary and collaborative nature of robotics research, this thesis naturally contains research that a number of authors contributed to. All research presented in this thesis was conceived of and either led or significantly contributed to by the author. The Acknowledgements at the beginning of this document contain a complete list of undergraduate, graduate, and professional collaborators, but the following significant contributors must be acknowledged for their invaluable efforts on certain chapters.

- Karin de Langis, for Chapter 5.
- Tanmay Agarwal, for Chapter 6.
- Jungseok Hong, for Chapter 7, Appendix III, and Appendix IV.
- Elsa Forberger, for Chapter 8.

Chapter 1

AUVs, Underwater Environments, and UHRI

The field of underwater human-robot interaction is complex and multi-disciplinary, despite its relatively small size. Due to the still-emerging nature of the robots involved, research on small, human-portable collaborative autonomous underwater vehicles (co-AUVs) is sparse. However, in the years since the introduction of the first companion AUVs, a wide variety of threads of research have been established. In this chapter, we will discuss underwater robots, with a heavy emphasis on the emergence of collaborative AUVs, and describe in depth the AUVs used in the research contained in this thesis. Following this, we will briefly discuss the aspects of the relevant environments for deployment (pools, lakes, oceans, etc.) and briefly summarize the field of underwater human-robot interaction, focusing on the few groups whose work defines the field, with a few exceptions. This chapter should provide the reader with a background within which to place the research contained in this thesis.

1.1 The Advent of the Co-AUV

Water is fundamental to the existence of every human on this planet. It is the habitat of the organisms whose photosynthetic processes create the oxygen we breathe [19], we must drink water to survive, and everything we eat requires water to grow. Waterways were the principal commercial pathway for goods and people to move around the globe



Figure 1.1: Divers performing archaeological and scientific surveys underwater.

Images courtesy of NOAA and Mael Ballanad via Unsplash.

for much of human history, allowing the spread and growth of human society. Even now, our internet traffic flits through a massive network of undersea cables, powered by electricity derived by burning petroleum and natural gases extracted underwater and piped to power plants around the world. Our commerce and industry, our society, and our continued existence as a species all depend on water. Despite this, water resources on the Earth are under-explored and under-protected, and the work that is done in them requires enormous human effort and risk. The types of work conducted underwater are myriad, but a brief sampling would include underwater construction (installation, inspection, and repair) [20], search and rescue, archaeology [21], marine debris cleanup [22], pollution remediation, oceanographic survey [23], marine biological study [24], and water resource monitoring. These jobs support human life around the globe by utilizing oceanic resources to enable energy production and communication, providing aid in environmental efforts, expanding our knowledge about the subsea world, and maintaining safety in local recreational and potable water resources around the globe.

These critical tasks are conducted almost entirely by human divers working at depth, sometimes with the assistance of remotely operated vehicles (ROVs) piloted by surface workers. ROVs expand the reach of human divers by diving to greater depths for longer

times, carrying heavier loads, and operating in areas of extreme danger. The operating nature of ROVs requires human input from the surface, which introduces logistical issues of cabling, additional required equipment and infrastructure, communication line delay, and so on. Because work done underwater is difficult and dangerous due to the environment in which it takes place, it is an obvious candidate for robotic assistance. However, autonomous underwater vehicles are not utilized for many types of underwater work, seeing the most use in oceanographic surveys. While large, torpedo-like AUVs have been roaming the oceans for decades, their size and construction makes them unhelpful for the kinds of work done by humans underwater. What then, is the answer to this opportunity for robots to aide humans in critical work in dangerous environments? If ROVs introduce further infrastructure and require human input, and traditional AUVs are ill-suited for the tasks, what can be done to help automate and assist the work of underwater divers?

In the early 2000's, a new class of human-scale AUVs capable of operating alongside humans and assisting them in tasks began to emerge [1, 25–29]. These AUVs (often called mini or micro AUVs) are smaller, more highly maneuverable, and more diverse in their construction than their ocean-going cousins. While it was not always their intended purpose to be collaborators with divers, the miniaturization and new designs of these AUV has made them better suited to working alongside divers. This, along with a growing interest in underwater human-robot interaction research, has begun to create a new class of AUV, the *collaborative AUV* (co-AUV). Though few AUVs are actually being deployed as collaborative diver partners to humans in real-world applications, these smaller AUVs' capabilities are growing by leaps and bounds and the day is not far off when co-AUVs will be able to join humans in underwater work.

1.2 A Brief History of AUVs

The world of AUVs is vast and any summary claiming to be complete is likely to be inaccurate. However, to provide context for the work in this thesis, it is necessary to provide a brief overview of how these robots came to be. AUV development can be considered to have begun [30] in 1957 with the development of the Special Purpose Underwater Research Vehicle (SPURV), at the University of Washington, funded by the

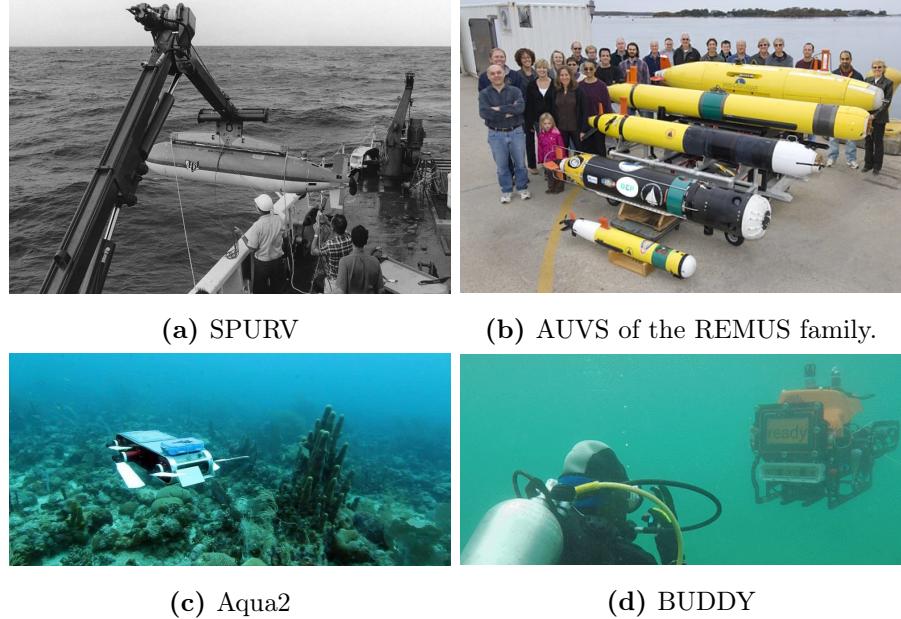


Figure 1.2: A sampling of AUVs from the first up to the present day.

Images courtesy of copyright holders.

United States Office of Naval Research [31]. SPURV was used until 1979, followed by SPURV II [32], which improved the hydrodynamic design of its predecessor and increased the number of sensors on board. Soon after, the REMUS AUV [33] was developed by Woods Hole Oceanographic Institute, in an attempt to develop a smaller, lower-cost AUV. Later, REMUS 600 [34] improved on the design of REMUS, with a side-looking synthetic aperture sonar and improved endurance and payload flexibility. By this time, gliders such as Seaglider [35] and Deepglider [36] were setting the standard for a new type of AUV: the long endurance glider, with mission times measuring in weeks and months rather than hours.

While traditional torpedo-shaped AUVs and gliders still dominate AUV design, innovation in the realm of mini-AUVs (mass of 20-100kg) and micro-AUVs (mass of 20kg or less) began to spring up, with the flipper-driven Aqua [1] being a prominent example of the variation now existent in the field. Variation in micro-AUV design has continued to grow with the development of HippoCampus [37] and SEMBIO [26], micro-AUVs for swarm applications, and AUVs with less typical drive designs, such as a momentum-drive

single actuated robot, which rotates its inner body to move the exterior passive flaps and produce swimming motion [38]. Other AUVs such as SHAD [38], HOBALIN [39], and Sparus II [28] focused on hovering motion for seabed inspection and observation tasks. Lastly, the development of general-purpose micro-AUVs is still going strong, with Bluefin Sandshark [27], a docking AUV [40], and other similar AUVs appearing in the last few years, including the LoCO AUV [29], which is used for much of the research in this thesis.

We have discussed the evolution of AUVs from the first submersibles to the dominance of torpedo-shaped gliders, to the newer wave of mini and micro-AUVs, but what of the co-AUV? While the early micro-AUV Aqua [1] was intended to work as a “dive buddy” with a diver, its design is not focused on that goal, instead being concentrated on amphibiousness and bio-inspired control. One of the first micro-AUVs prominently designed as collaborative was the BUDDY AUV from the CADDY FP7 project [9]. Described as a “cognitive diving buddy”, this AUV was designed for interactivity and collaboration from the beginning, with a large, removable tablet for interaction. While other AUVs have been used for studies of human-robot interaction (most notably Aqua and LoCO), BUDDY remains one of the few AUVs that have been designed for collaboration. In the current state of the field, the term “co-AUV” largely refers to the potential use of micro-AUVs as collaborative partners for divers rather than a distinct hardware class. It is likely, however, that as the field grows in size, more distinct hardware will become available, expanding a new branch of the family tree, beginning with the Aqua and BUDDY AUVs.

1.3 Co-AUVs Used In This Thesis

While many AUVs and co-AUVs have similar characteristics, each robot has unique capabilities and poses different challenges. All of the research contained in this work could conceivably be applied on any AUV, with differing degrees of difficulty. For this reason, we now describe the AUVs which were used for this thesis: Aqua and LoCO.

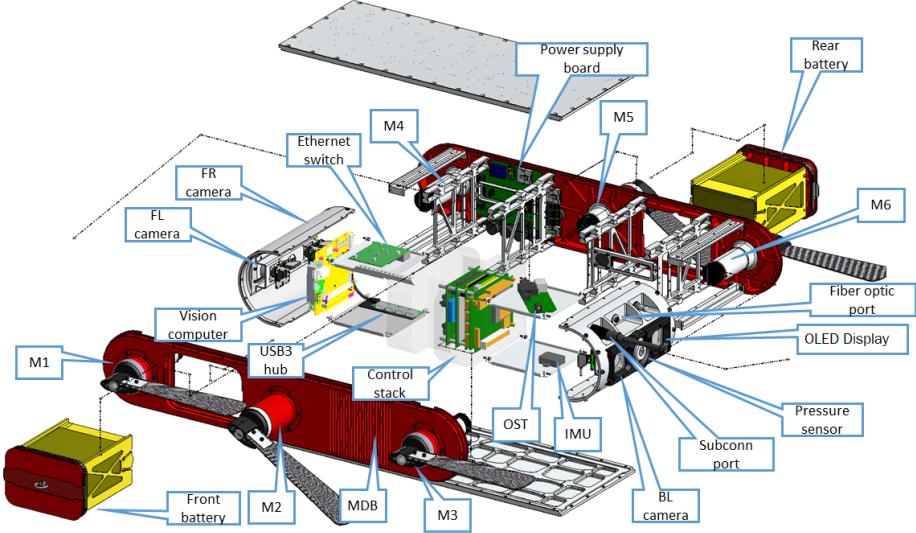


Figure 1.3: The Aqua2 AUV “exploded” with all of its system annotated. Image courtesy of the Mobile Robotics Laboratory [1].

1.3.1 Aqua

The Aqua AUV [1] is a six-legged, amphibious, autonomous underwater vehicle (AUV) developed principally at McGill and York Universities. Equipped with three cameras (a stereo pair in the front and a third in the back), a depth sensor, and an inertial measurement unit (IMU), Aqua is capable of tether-less operations for over three hours. For human-robot interaction, the AUV uses its back camera and a small OLED-type display. While Aqua’s flipper-based drive system gives it a lower top speed than some AUVs, its motion is highly dynamic and expressive, with independent translation in the x axis (*surge*, forward and back) and z axis (*heave*, up and down) and rotation about the x axis (roll), y axis (pitch), and z axis (yaw). It is a reliable AUV, but is quite expensive and difficult to modify due to its solid aluminum shell and custom electronic components.

1.3.2 LoCO

LoCO-AUV is a **Low Cost, Open** autonomous underwater vehicle. Developed by the Interactive Robotics and Vision Laboratory at the University of Minnesota, LoCO is

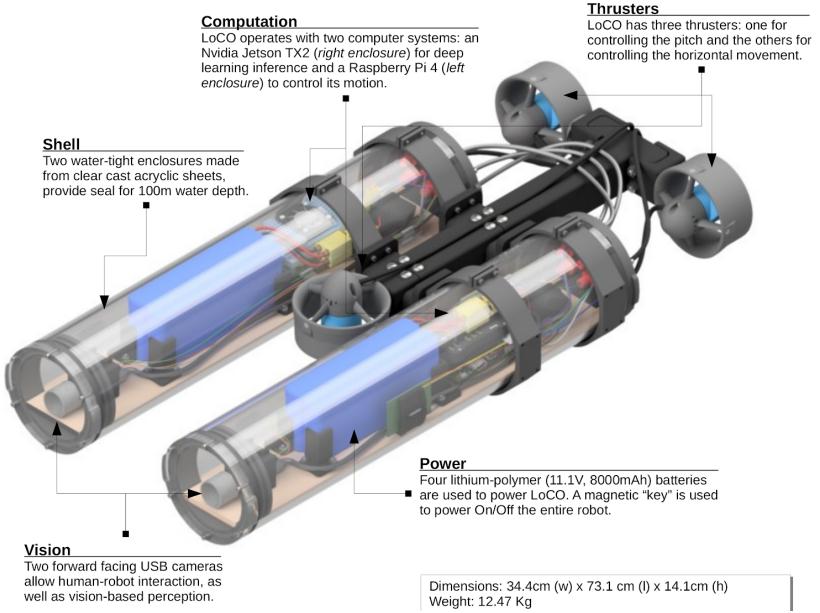


Figure 1.4: A CAD drawing of LoCO-AUV with its primary systems annotated.

built of less than 4,000 USD worth of off-the-shelf and 3D-printed parts. It is driven by three thrusters, which allows the AUV independent translation only in the x axis (*surge*, forward and back), and rotation about the y axis (*pitch*) and z axis (*yaw*). While its motion is not as complex or dynamic as Aqua's, LoCO accelerates much faster and has a higher top speed. Equipped with two monocular cameras, a depth sensor, and an IMU as its base sensor configuration, LoCO is a highly extensible platform. Its structure and design make it a relatively simple matter to swap out sensors or other components. For HRI, LoCO is equipped with a 2.7 inch OLED, and has subsequently been fitted with custom-built LED display devices (Chapter 3) and an audio output device (Chapter 4).

1.4 Underwater Research Environments

Underwater work environments introduce a number of considerations when developing robots that operate in them. Among these issues are:

- Severely limited range (< 50 cm) [41] of moderate to high bandwidth radio-frequency communications.

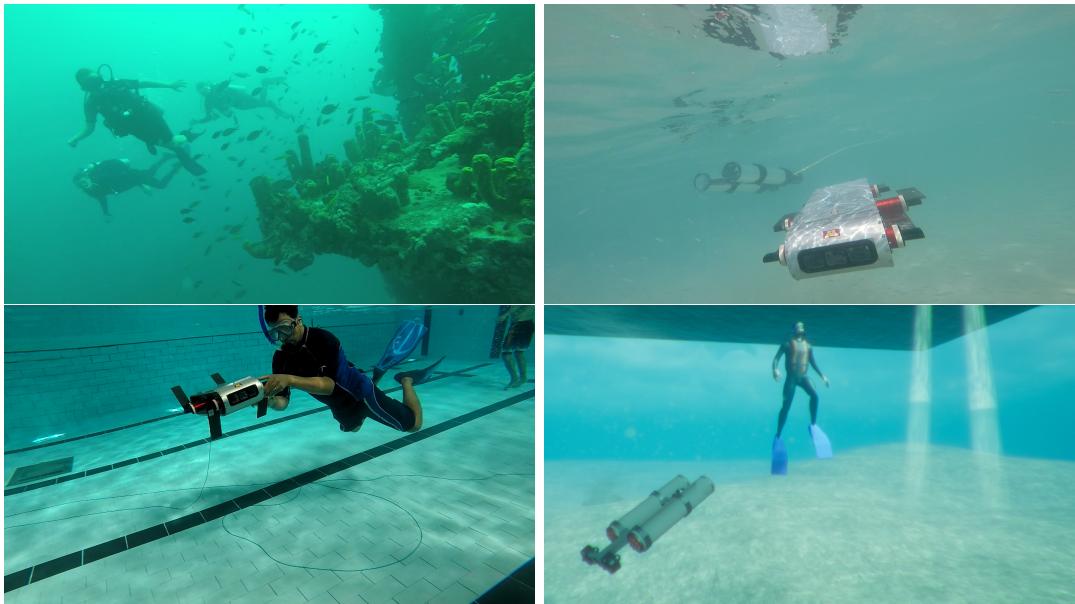


Figure 1.5: Divers and AUVs in a variety of field, pool, and simulated environments.

- No cheap global localization sensors with accuracy comparable to GPS.
- No active sensors such as LIDAR or infrared depth cameras.
- Limited visibility and degraded image quality (backscattering and particulates)

While every underwater environment has these issues to a certain degree, different types of water environments have different properties. The majority of our research is conducted in simulated and pool environments, but field environments are always the target environment.

Simulation of underwater environments is difficult. The highly dynamic visual conditions, complex hydrodynamics, and under-explored areas that are common in field environments make the problem of creating a simulated environment that is sufficiently similar to a field environment a nearly impossible task. For the Aqua and LoCO AUVs, we mostly utilize simulators based on the popular robot simulator Gazebo, as well as simulators based on Unreal Engine and Unity. While these simulators are helpful for prototyping algorithms, the simulation of AUV motion is quite inaccurate, making it difficult to use them for final development and evaluations. Pool and field environments

are where the majority of development and evaluation can happen for AUV algorithms. Pool environments have the benefit of being more tightly controlled and safe, but do not have the realism of field environments. Field environments, on the other hand, are highly unpredictable and riskier, but much more difficult to operate in, making them a poor choice for algorithms that are not mature and robust. Due to these considerations, the typical development cycle for underwater algorithms is prototyping in simulation, followed by pool evaluations once a working algorithm has been created, and finally field evaluations to determine if a method is truly ready for field deployment. However, due to the time and money required to perform field evaluations, not every method gets evaluated in the field.

1.5 The Current State Underwater HRI

Currently, human-scale AUVs are not being deployed widely around the world, with co-AUVs being even more uncommon. Nevertheless, an interaction paradigm has emerged in the work of researchers designing and developing such AUVs. We will briefly touch on the standard methods of AUV-directed and diver-directed communication that have become common over the last several decades. This is not an exhaustive review, though the majority of AUV research focused on interaction does generally follow this paradigm [42]. Put simply, co-AUVs are typically controlled via external devices, fiducial flashcards, or gestures and communicate information to humans using internal digital displays, external devices, or (rarely) indicator lights. AUVs operated in this manner are typically controlled very directly, with divers specifying task parameters in gestural programs and keeping the robot close by.

1.5.1 AUV-Directed Communication

Communication between a robot and an AUV has two very different entities participating in it. The unique capabilities of both humans and AUVs affect what forms of communication work well for them. This creates a heterogeneous communication landscape where the methods used by AUVs to communicate to divers differ from those used by divers to communicate to AUVs. In the following sections, we discuss the primary methods used for human-to-AUV communication: external devices, fiducial flashcards,

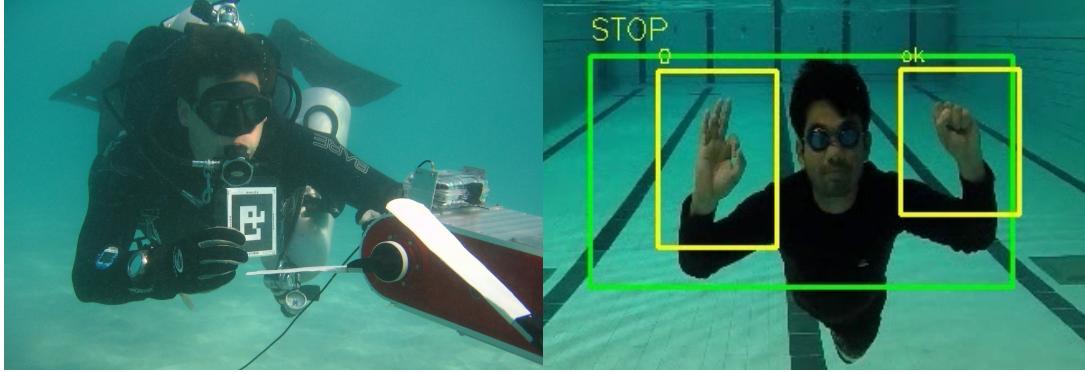


Figure 1.6: Divers using fiducial flashcards (left) and gestures [2] (right) to control AUVs. *Left image courtesy of McGill Mobile Robotics Laboratory.*

and hand gestures.

External Devices

In the slow transition from using ROVs for underwater work to using AUVs for the same task, controlling those AUVs using a dedicated control device underwater is a natural step. Early attempts at AUV control via external devices include Verzijlenberg *et al.*, [17] and their use of a wired tablet inside a water-tight enclosure. The BUDDY AUV [9] created for the CADDY project uses a tablet in waterproof casing as a method of control, similar to Verzijlenberg *et al.*, . The controller [43] used by MIT’s SoFI soft fish AUV is another example of two-way controllers used as interaction devices, though the SoFI controller utilizes acoustic communication rather than a wired interface. These devices allow specific, direct control, but introduce equipment burdens for the divers using them, and create another possible point of equipment failure. Remote control devices have their place, allowing direct control when necessary, but they are not without their weaknesses and are fundamentally not devices for interaction with an autonomous vehicle. Rather, they are a way of reducing the distance between a remotely operated vehicle (ROV) and operator.

Fiducial Flashcards

The use of fiducial markers as flashcards (Figure 1.6, left) [4, 10, 44] was the first device-free method of AUV control that took hold for collaborative AUVs. Fiducial markers are

often used in augmented reality and robotics to act as a point of reference. The markers in question are typically a patterned square of white and black boxes, optimized for easy detection using simple computer vision and image processing methods. Their use in AUV control was born out of necessity: early AUVs needed some signaling device that was easy to detect and fiducials such as the ARToolKit and ARTag markers are well suited for this purpose. Assigning program execution hooks to each tag within a predefined set allows divers to control an AUV working with them by simply displaying a fiducial to the AUV. The main drawbacks of this method are the fact that a set of tags can easily be dropped or lost and that the markers themselves have no semantic meaning for humans.

Hand Gestures

To further develop the fiducial control system, some early research explored the use of gestures made with fiducial markers as a method of AUV control [45]. This method was successful, but did not receive wide adoption. It was not until the improvement of deep neural networks for object detection and gesture recognition that gestures began to be considered for AUV-directed communication, this time in the form of hand gestures (as in Figure 1.6, right). Often based on diving hand signal languages and formalized similarly to a programming language, these more recent hand gesture recognition methods [2, 11] are easily the dominant method of AUV-directed communication. However, these languages are relatively strict in their interpretation, typically only use static gestures (hand positions with no motion), and still suffer from difficulties in recognition.

1.5.2 Diver-Directed Communication

Having discussed diver-to-AUV communication, we now turn to AUV-to-diver communication, which is dominated by the use of digital displays and external devices, but less frequently uses indicator lights.

Digital Displays

Digital displays are by far the most common vector of human-directed communication in UHRI. Many AUVs contain a digital display of some kind, though the size of the display



Figure 1.7: A diver attempts to read the Aqua AUV’s display, with a closer view of the display on the right (*Courtesy of McGill Mobile Robotics Laboratory*).

can vary significantly [1, 11, 15]. Displays are easily reconfigurable, capable of displaying complex information, and use an established method of information transfer (written language). However, the displays embedded in AUVs can be quite small (as seen in Figure 1.7), making them difficult to read at any distance, and are also difficult to read at an angle (particularly at angles greater than 90°, reading becomes infeasible) [46]. Further, digital displays are not particularly helpful if the intended recipient of the information is not already looking at the display. This greatly limits the effective range of communication, making it impossible to attract a diver’s attention or communicate information to them without requiring their full attention. For these reasons, while displays are useful for communicating complex/high density information, they should be used in combination with other methods of AUV-diver communication.

External Devices

Less common than digital displays, remote control devices are still a frequent vector of diver-directed communication, though they typically function as two-way controllers. We previously described external control devices in terms of their AUV-directed communication, but these control devices also typically allow information to be transmitted from the AUV (over cable or audio modem) to be displayed on the controller. External devices have the same benefits of digital displays: complex/high density information is straightforward to communicate, but the drawback is the additional equipment load and risk of equipment failure which is intrinsic to carrying a digital device underwater.

Indicator Lights

Lastly, a small number works have suggested the use of an AUV’s functional work lights for communication, briefly mentioned as a feedback method in Verzijlenberg et al. [17] and further proposed by Demarco et al. [16]. Many AUVs have lights for the purpose of illuminating scenes when in or around the aphotic zone (deep enough underwater that light levels are very low). These lights can be flashed on and off or have their intensity varied in order to create a Morse code-like system. While this new method was evaluated in an in-pool case study in [16], it has yet to be explored further.

1.5.3 The Current Paradigm

Considering these methods of communication and control, we can begin to see the current paradigm of co-AUV operation. Observant readers will notice that the majority of the communication vectors described require close proximity between the AUV and its interactants. This often leads to a “phantom tether” between the diver and an AUV, where the diver rarely lets the AUV out of arm’s length. Additionally, collaborative AUVs are frequently programmed to execute a pre-defined mission or are dynamically programmed (rather laboriously) to complete simple tasks. This imperative command structure is effective, but does not sufficiently support the dynamic nature of underwater work: if a task for the AUV becomes available, the diver should be able to easily assign the AUV to it, without having to pre-program the mission or write programs with gestures underwater. Additionally, the common methods of communication (AUV and human-directed) are generally *unintuitive and technological* rather than *natural and human*. Taken as a whole, the current paradigm of AUV interaction is the way one controls a computer, not the way one would collaborate with a partner. While AUVs are obviously computerized devices, it is our position that researchers should strive for natural, intuitive, human-like communication in service of creating a true collaborative AUV, capable of dynamic and flexible deployments. The research presented in the following chapters is our contribution toward that goal: three novel methods of natural AUV-to-Human communication, advancements in the state of the art of AUV perception of humans, and the first attempts at making interaction underwater adaptive to the context of the interaction. We begin with Part I: natural AUV-to-human communication.

Part I: Natural Methods for AUV-To-Human Communication

Robot-to-human communication is a task of critical importance for any interactive robot, including AUVs. Without the ability to provide a human partner with information, ask for commands and clarification, or provide instructions to a human, an AUV is restricted to the role of a silent worker, greatly reducing its utility. However, communication in underwater spaces is difficult due to the medium of interaction: water. Water diffuses and distorts audio communication, makes digital displays difficult to read, eliminates the use of handheld wireless controllers, and visibility can vary significantly with distance. Despite these challenges, robots must be able to communicate with divers to facilitate collaboration underwater. As established in Chapter 1’s summary of UHRI, the standard paradigm of AUV-to-human communication uses digital displays integrated into AUVs and dedicated control devices. While these methods allow for complex information to be communicated simply, displays are nearly impossible to read at a distance or an angle, and separate control devices add additional equipment which a diver must be responsible for. The research in the three following chapters is a departure from the established paradigm, using motion, light, and sound as communicative signals. These methods can be understood from a distance, do not require the diver to carry additional devices, and yet provide robust, intuitive ways to communicate. All three are groundbreaking methodologies, with Chapters 2 and 4 introducing novel vectors of motion and sound for underwater AUV-to-human communication and Chapter 3 significantly breaking from established uses of light. Together, the research presented in Part I has broadened the field of UHRI, expanding the ways in which an AUV can communicate to a diver.

Chapter 2

Motion for AUV-To-Human Communication

Motion is a vast and powerful method of human interaction, from gestural languages such as ASL, head nodding and shaking, to body language, humans are constantly processing motion-based communication. Robots have used this form of communication since the early days, with their facial expressions and hand gestures growing ever more realistic and expressive. Due to their relative recency and appearance constraints, however, AUVs have yet to utilize motion as communication. Motion has obvious advantages: it is easily viewed from a moderate distance or angle, many simple concepts or phrases have natural gestural equivalents, and humans are predisposed to recognizing motion-based communication in one another. However, it is difficult to create communicative motion for AUVs. Their appearances are typically non-humanoid and purely functional, meaning that humanoid gestures such as facial expressions, pointing, waving, etc, cannot be easily applied. In this chapter, we cover our work creating motion-based communication for AUVs, including the design process and three human studies covering the viability of motion communication in simulation, in implementation, and in differing interaction contexts when compared to other options.

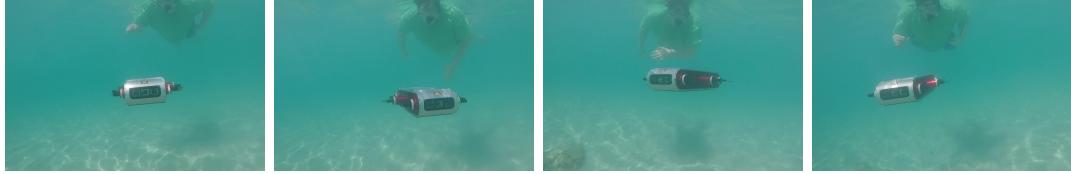


Figure 2.1: The Aqua AUV in the Caribbean, indicating a K_{No} by shaking its “head”.

Definition: Kineme

Our work on motion-based communication centers around the production of *kinemes*, a sequence of robotic motion with an associated semantic meaning. The word kineme comes from the Greek *kinetikos*, meaning “putting in motion” combined with the Greek *eme*, a unit of linguistic structure. In the categories of gestures defined by Mark L. Knapp [47], kinemes can be considered emblems, a category of gesture to which the signs of American Sign Language (and other localized sign languages) belong. In this document, a kineme will be designated by the symbol $K_{meaning}$, where “meaning” is parameterized as a simple description of the meaning. Our system for the production of kinemes is referred to as Robot Communication Via Motion or RCVM.

2.1 Background: Motion As Communication

Before discussing RCVM, we describe existing methods for communication via motion for robotics, excluding underwater communication (which is covered in Chapter 1). The use of motion as a vector of communication is an old concept, dating at least back to early attempts to create mechanical automatons mimicking the motion of humans and later research on creating androids and humanoid robots. Human communication uses motions (conscious and unconscious) as modalities of communication secondary only to spoken language. As such, it is natural for motion to be considered as an option for human-robot communication. However, the majority of motion-based communication has been explored in non-field environments, and for affective communication (communication of feelings) rather than transfer of information, which is our focus. The largest amount of motion-based information communication in previous work can be found in the terrestrial domain. The vast majority of motion-based communication research, however, use humanoid robots, typically attempting to replicate human gestures [48–50],

facial expressions [51–53], and body language [54,55] on these platforms. While fascinating, this type of research has little benefit for us, as nearly no field robots are humanoid in form.

In terms of non-humanoid motion interaction, Cindy Bethel’s seminal thesis [56] is one of the most important works in the field, using the angle and motion of non-humanoid search and rescue robots to communicate affect from the robots to the humans they are helping. This work [56] was influential to ours as it introduced the idea of the variation in communication vector effectiveness at different distances (but not at different angles) and the concept of using multiple vectors of communication, either in concert or separately. Not having access to the same robots or communication systems, we are unable to reproduce Bethel’s results directly (the content being communicated also differs), but our results on the effect of viewpoint (Section 2.9.3) on communication echo Bethel’s work throughout. Bethel has studied and summarized non-facial, non-verbal affective communication quite extensively, providing useful surveys as well as her own contributions to the topic [57,58]. This type of motion-base affective communication has also been applied to zoomorphic robots such as a canine robot [59] and learned over time by an agent given the feedback of a human [60]. Non-affective motion communication has also been explored, from the use of a pan-tilt camera to simulate head nodding and other gestures [61] to using a digitally displayed virtual “head” to generate gaze cues in order to manage navigational conflicts with humans [62].

A large body of work by Dragan et al. has covered the topics of legible pointing [63], nonverbal communication for feedback in teaching by demonstration [64], the expressiveness of timing in manipulation motion [65], and the effect of different types of robot motion on human-robot collaboration outcomes [66] for terrestrial robots, largely in the realm of manipulators. Similar research on communication for aerial robots by modifying flight paths to indicate direction or intent has been conducted by Szafir et al. [67] and Duncan et al. [68]. These kinds of flight path “communicative overlays” aide human partners in working with these drones. Flight path modification has also been used by Cauchard et al. [69] to encode emotional states in drone flight. However, the Daedelus [70] social unmanned aerial vehicle (s-UAV) is more closely related to our research. Daedelus used a motorized ‘head’ with colored ‘eyes’ to express emotions to interactants, capable of head tilting and nodding movements. These works all have some

#	Interaction Phrase	Meaning	Study I	Study II	Study III	Phrase Type
0	$K_{Affirmative}$	Yes	<i>Animated</i>	<i>Implemented</i>	<i>Implemented</i>	Conversational
1	$K_{Negative}$	No	<i>Animated</i>	<i>Implemented</i>	<i>Implemented</i>	Conversational
2	$K_{Possible}$	Maybe	<i>Animated</i>	Eliminated	Eliminated	Conversational
3	K_{Ascend}	Move up	<i>Animated</i>	Replaced with #5	Replaced with #5	Spatial
4	$K_{Descend}$	Move down	<i>Animated</i>	Replaced with #5	Replaced with #5	Spatial
5	$K_{Directional}$	Move in [direction]	Not created	<i>Implemented</i>	<i>Implemented</i>	Spatial
6	K_{Stay}	Remain where you are	<i>Animated</i>	<i>Implemented</i>	<i>Implemented</i>	Spatial
7	$K_{Attention}$	Look at me	<i>Animated</i>	Unused	<i>Implemented</i>	Conversational
8	K_{Danger}	Danger nearby	<i>Animated</i>	<i>Implemented</i>	<i>Implemented</i>	Status
9	$K_{FollowMe}$	Follow me	<i>Animated</i>	<i>Implemented</i>	<i>Implemented</i>	Status
10	$K_{Malfunction}$	Something is wrong	<i>Animated</i>	Unused	<i>Implemented</i>	Status
11	$K_{RepeatPrevious}$	Repeat last instruction	<i>Animated</i>	<i>Implemented</i>	<i>Implemented</i>	Conversational
12	$K_{ObjectIndication}$	Object here...[direction]	<i>Animated</i>	<i>Implemented</i>	<i>Implemented</i>	Spatial
13	$K_{BatteryIndicator}$	Battery level is...[battery]	<i>Animated</i>	<i>Implemented</i>	<i>Implemented</i>	Status
14	K_{Lost}	The robot is lost	<i>Animated</i>	Unused	<i>Implemented</i>	Status

Table 2.1: Development of the RCVM language over time.

similarity to our study of RCVM, though we focus on informative communication (unlike Bethel [56]) via explicit motion gestures (unlike Dragan [71]), and use the base motion of a non-humanoid robot instead of adding human features (unlike Hart [62]).

2.2 Kineme Design

The task of creating a system for communicating meaning with motions is a difficult one, particularly when dealing with non-humanoid robots. When designing motion for humanoid robots, one can turn to the extensive literature on human body language, gestures, and facial expressions for inspiration and understanding. Similarly, the design of interfaces on digital displays, or interfaces using light and sound is an art that has been well-established and studied. Creating meaningful motion for a non-humanoid form requires inspiration from further afield, and is a task that few have attempted.

2.2.1 Creating Kineme Content

The first phase in developing RCVM was to select a common library of phrases for communication. These phrases and their development over time can be seen in Table 2.1. To develop the interaction phrases, we generated a comprehensive list of ideas in response to the prompt "What information do you want an AUV companion to be able to

tell you?". Through an informal iterative coding process, these phrases were distilled into their raw conceptual forms. For instance, multiple answers such as "rough water ahead", "shark nearby", and "small, confined space ahead", were distilled into the concept of danger being nearby, which became K_{Danger} . While most of these kinemes remained unchanged in their meaning throughout our research, K_{Ascend} and $K_{Descend}$ were made to be more general by replacing them with $K_{Directional}$, which indicates movement in any direction in three-dimensional space. Additionally, $K_{Possible}$, was removed after Study I (Section 2.4), based on discussions with divers on the usefulness of each phrase. The kinemes in Table 2.2 are a further updated version (the same set of meanings used for the HREye and SIREN systems described in Sections 3.3.1 and 4.3.2). All of the kinemes contained in the multi-dimensional study are in Table 2.2, with the exception of $K_{Indicate_Object}$.

2.2.2 Designing Kineme Motions

The next task after creating the list of kinemes was to design the actual motion for each. We looked to human body language for inspiration, emulating common human gestures whenever the shape and motion capabilities of the robot allowed it. For instance, by yawing the AUV back and forth around the vertical axis, an approximation of a head-shake can be achieved. This presents a question: what is a common gesture? The example provided, nodding and shaking of the head, might seem universal at first glance, depending on the readers' background, but this is not the case. For example, in some Balkan countries including Bulgaria [72], the meaning of head nodding and shaking are reversed from the typical meanings. While our experimental populations thus far have been almost entirely comprised of residents of the United States, these cultural perceptions of motions should be considered during motion design.

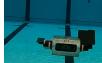
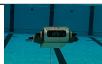
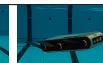
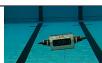
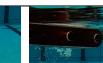
Kineme	Meaning	Visuals
$K_{Affirmative}$	Yes, Okay.	  
$K_{Negative}$	No.	   
K_{Danger}	Danger in the area.	   
$K_{Attention}$	Pay attention to the robot.	   
$K_{Malfunction}$	Internal malfunction.	   
$K_{WaitCMD}$	Waiting for instructions.	   
$K_{Indicate_Direction}$	Go to direction.	   
$K_{WhichWay}$	Which way should we go?	   
K_{Stay}	Stay where you are.	   
$K_{ComeHere}$	Come to the AUV.	   
$K_{FollowMe}$	Diver should follow AUV.	   
$K_{FollowYou}$	AUV will follow diver.	   
$K_{BatteryLevel}$	Battery level is	   

Figure 2.2: The kinemes of the RCVM system demonstrated on Aqua. Note that these kinemes are a newer version, not the version tested in the described studies.

When we could not copy human gestures directly, we sought to distill the interaction phrase into directional information (such as turning towards an object that is meant to be indicated) or considered the associated emotions (such as fear for K_{Danger} , leading to the selection of quick, jerky motions). This was a much more artistic process, which we began with a review of media containing nonverbal characters (the Magic Carpet in



Figure 2.3: The Aqua AUV simulation using Unreal Engine used for Study I.

Disney’s Aladdin, R2-D2 in Star Wars, and the entirety of WALL-E) as well as guidelines from the field of animation. These sources provided help and inspiration. For all kinemes, we sought to keep their performance time relatively low, so as to not interrupt other necessary motions the robot might need to make, such as station-keeping or navigation to a new area.

2.3 RCVM Implementations

With the motions for each kineme selected, the next step is implementing these kinemes so that the appropriate motion can be produced on a robot. While the focus of our work is communication for AUVs, we have implemented RCVM on two other robots, which will be briefly described in this section (but not discussed in detail). For more information about the multi-robot implementation of RCVM, please see our article in ACM Transactions on Human-Robot Interaction [46].

2.3.1 Aqua AUV

As previously mentioned, the Aqua AUV is a highly dynamic robot, capable of independent rotation on all three axes. However, any motion taken can lead to drift, as there is significantly less friction acting on the robot than on a terrestrial vehicle. This can lead to issues with the vehicle’s motion control software, which is comprised of a set of Proportional-Integral-Derivative (PID) controllers [73, Chapter 9.3] which attempt to

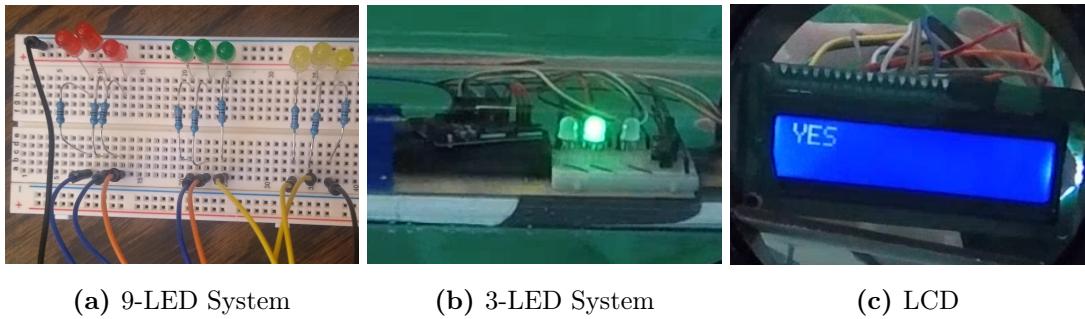
move the robot to target angles and positions. For example, if a water current pushes the robot in a direction, it may over-correct and overshoot its target position, requiring another correction to reach the target and continue with the motion plan. The Aqua family of robots are biomimetic in design, having a biologically-inspired shape, with flippers that remind many observers of fish or cockroaches and two cameras in the front of its shell which are often referred to as eyes by onlookers. This serves it well in the development of kinemes, as kinemes based on human head gestures can be imitated with relative success.

2.3.2 LOCO AUV

Due to the lack of a proper motion controller for the LoCO AUV, the implementation of RCVM on the LoCO AUV is quite simple. For each kineme, a timed set of control inputs has been derived using iterative, in-water testing. This method produces acceptable kinemes in relatively calm water, but would fail to do so in rough water, or with interference. To overcome this issue, a reactive motion controller capable of responding to changes in the environment would need to be implemented for LoCO, an engineering task which is relatively straightforward, but out of the scope of this thesis. Beyond this technological shortcoming, the primary difficulty of designing kinemes for LoCO is that unlike the Aqua AUV, LoCO has no independent roll control. Kineme translation from Aqua to LoCO therefore involves removing the roll components of kinemes, often replacing them with motion in other axes.

2.3.3 Matrice 100 and Turtlebot2

Along with our AUV implementations, we also implemented RCVM on two non-aquatic robots: the Matrice 100 and the Turtlebot2. The Matrice 100 drone is an aerial vehicle capable of a variety of motions. However, for the development of kinemes on the vehicle, the motion being utilized is not that of the vehicle itself, but rather of its camera, mounted on a Z3 gimbal. This gimbal is capable of a wide range of motion, with 320 degrees of pan, 120 degrees of tilt, and 120 degrees of roll. Because of its highly expressive motion, it is an excellent analog for the human head, making the implementation of many kinemes fairly easy. However, due to the robot's size and the distances at which



(a) 9-LED System

(b) 3-LED System

(c) LCD

Figure 2.4: The baseline systems used for comparison to RCVM in Studys I and III.

operators may be located, its effective "transmission range" may be limited.

Designed as a robot for teaching and use in office environments, the Turtlebot2 is kinematically and dynamically most related to warehouse robots (operating on a differential drive which allows free movement on the X axis and rotation around the Z axis only). The Turtlebot was selected as an example of a terrestrial robot (not a field robot itself), as it was the most cost-effective platform available for development, and shares motion characteristics with actual terrestrial field robots (X-axis translation and Z axis rotation). In order to work within this limitation certain features, such as wobbling after an abrupt stop, were employed to gain motion outside the 2D plane.

2.3.4 Baseline Systems

For Study I and III, a variety of baseline communication systems were created. These systems are not meant to represent the best possible communication system of their type, but rather an average system using the medium in question (lights, digital displays, and sound). More advanced communication systems using light and sound are presented in Chapters 3 and 4, respectively.

9-LED System

This system (Figure 2.4a) was created as a comparison system for RCVM in Study I. It was designed to create an encoding from light color and flash rates to semantic meaning, building on the suggestions of Demarco [16] and Verzijlenberg [17], who both created communication systems utilizing a single-color light's flashes. In this case, the

system was designed to use a combination of green, yellow, or red LEDs to communicate concepts equivalent to the kinemes tested in Study I.

3-LED System

The 3-LED system (Figure 2.4b) was an improvement on the 9-LED version, focused on reducing system size, increasing color options, and improving long-distance visibility. Three larger RGB LEDs were used instead of 9 smaller single-color LEDs, enabling the use of a wide variety of colors. This system was used in Study III as a comparison and was part of the inspiration that led to the HREye system described in Chapter 3.

LCD System

The LCD communication system (Figure 2.4c) is an analog to the various digital displays used in many AUVs. In this case ,for each interaction phrase, a short text which represents the meaning is displayed for five seconds on a two-line, sixteen character-per-line backlit liquid crystal display, driven by an Arduino. The screen displays white text on a blue background, which makes it particularly good for reading underwater compared to other display. However,it has poor viewing angles and is still difficult to see from any distance.

Audio System

The audio baseline communication system was used in Study III, and is relatively similar to the TTS-sonemes mode of the SIREN system described in Chapter 4. Unlike SIREN, this audio system was created by simply immersing a waterproof speaker in water, and controlling it via Bluetooth connection. For this reason, the speaker was kept at the surface, to allow the connection to its host device to remain stable. As the speaker was not designed for producing sound underwater, the speech produced by the Google Text-to-Speech API was somewhat but still audible.

2.4 Study I: Kineme Viability and Learning

In order to determine if motion-based communication was a viable option for robot-to-human communication, we preformed an evaluation on a simulated underwater robot for

a small, three-group study of 24 participants. Since no previous work evaluating motion for communication underwater had been conducted, we chose to begin our work with a pilot study based on a simulated version of the robot in Unreal Engine to limit overhead costs. In order to have a point of comparison to work against, we tested not only our first, simulated version of kinemes for an underwater robot, but also a baseline communication system based on colored LED codes called lucemes (definition in Section 3). This baseline system is the 9-LED system described in Section 3.1. The list of interaction phrases evaluated in this study can be found in Table 2.1. The structure of the study was as follows: users were randomly assigned one of three preparatory training levels, then were asked to test the kineme and LED communication systems by watching randomly ordered videos of interaction phrases (kinemes or lucemes) and answering several questions. This study was submitted to the University of Minnesota’s Institutional Review Board and determined to be Not Human Research (reference number: 00004182).

2.4.1 Experimental Design

The hypotheses we wished to test in this study were as follows:

- **Kineme_Accuracy:** Participants will interpret kinemes more accurately than lucemes.
- **Kineme_Operational:** Operational accuracy (accuracy for answers with a confidence ≥ 3 on a scale of 1 to 5) for kinemes will be higher than kineme accuracy and higher than luceme operational accuracy.
- **LED_Time:** Time taken by participants to answer will be longer for kinemes.

In order to test these hypothesis, we recruited 24 participants from student email lists for our study. Once they had been enrolled in the study and answered a demographic survey, each participant was provided with the education based on the education group they were in:

- **EDU0:** Participants were told which communication system they would be testing next (motion or LEDs).
- **EDU1:** Participants were told the communication system, as well as a list of possible phrases.

- **EDU2:** Participants were told the communication system and shown videos of each kineme or luceme while being told the meaning.

Education was offered directly before testing each system and was not repeated upon participant request. Once a participant had been educated, they were shown videos of the kinemes or lucemes in random order. Additionally, the order of the system to be shown was randomized, further reducing the chances of order effects and producing more independent measures of each system and interaction phrase. For each video shown to a participant, three pieces of information were recorded. Firstly, the participant reported the meaning they perceived from the kineme or luceme in the form of a free answer. The method used to evaluate the correctness of each answer is discussed further in Section 2.4.2. The time, taken from the beginning of each video until the beginning of the answer, was recorded, along with the participant's confidence in their answer rated on a scale from 1-5.

2.4.2 Analysis

In the study, each participant provided an answer to the question: "What is the robot trying to say in this video" as a free answer, not restricted to a number of words or a specific set of choices. In order to determine correctness, a rubric composed of five categories was defined: 1) Entirely wrong. 2) Slight conceptual relation. 3) Partially correct. 4) Correct understanding, but worded differently. 5) Verbatim correct. Four independent raters analyzed the responses of participants according to this rubric. The inter-rater reliability measure of Krippendorff's alpha coefficient [74] is used to determine the level of reliability in the scoring of the raters. Krippendorff's alpha coefficient may be seen as a measurement of how the rater's agreement compares to the agreement of random selections, with $\alpha = 1$ indicating far greater agreement than random (perfect reliability), $\alpha = 0$ indicating the agreement expected of random (absence of reliability), and $\alpha < 0$ indicating a greater amount of disagreement than expected of random selection (systemic disagreement). The raters' scores were found to have $\alpha = 0.9059$, indicating strong evidence of reliability. Therefore, the correctness scores of all four raters were averaged without weighting and used for all further analysis.

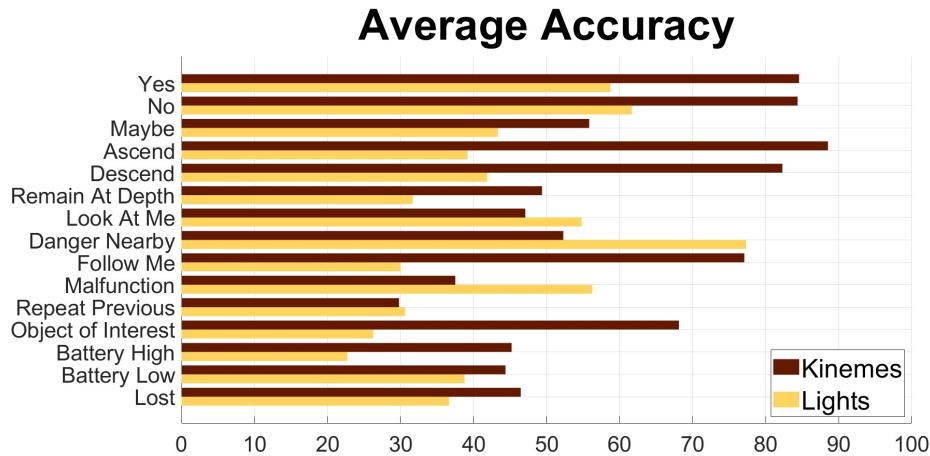


Figure 2.5: Accuracy of each kineme and luceme from Study I, averaged across all education levels.

2.5 Results of Study I

The kineme and light system are compared on the basis of the following criteria:

- **Accuracy** – The accuracy of a participant’s understanding of a kineme or luceme, rated from 1 to 5 in order of increasing accuracy.
- **Confidence** – The confidence a participant has in their understanding of a kineme or luceme, rated from 1 to 5, in order of increasing confidence.
- **Operational Accuracy** – The same metric as accuracy, but only taking answers rated at a confidence level of 3 or higher, representing the answers that participants would be likely to act on.
- **Time To Answer** – The time it takes a participant to give the meaning of a kineme or luceme, measured in seconds from the beginning of the signal to the beginning of their answer.

Results for each kineme and luceme in terms of accuracy can be found in Figure 2.5. These results show some kinemes which work better than their related lucemes and vice versa. In order to evaluate the hypotheses outlined in Section 2.4.1, we use

a Mann-Whitney [75] test with Bonferroni correction [76]. The Mann-Whitney test is ideal for measuring the statistical effects of using the different systems in our trials, as it does not require normally-distributed data.

2.5.1 Study Population

The population ($N=24$) used in this study was mostly composed of students (16 male, 8 female) from the University of Minnesota-Twin Cities with a mean age of 22 ($\sigma = 3.3$). In order to ensure that our population would be representative of non-expert users, participants were asked to rate their experience level with robots on a scale from one to five, with one being low ($\mu = 1.54$, $\sigma = 0.5$). While this group was limited in size and age range, the average level of robotics experience was low. Because of this, while their experiences with the kineme system might generalize to the public at large, and is certainly useful for guiding the development of the system, further testing to verify a lack of effect from age or other factors would be beneficial. Participants were randomly put into one of three groups (eight members each), designated EDU0, EDU1, and EDU2 ordered by the amount of preparatory training they received for the communication task they were tested on.

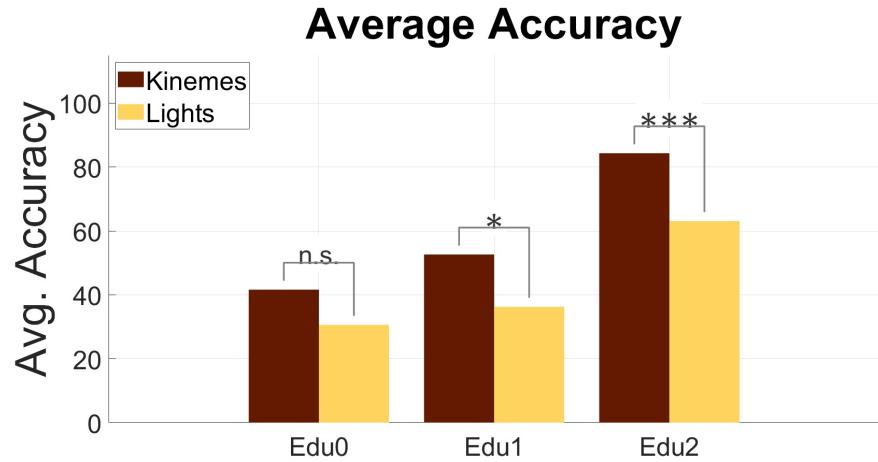


Figure 2.6: Accuracy of the kineme and light-based communication systems compared across education levels. Statistically significant improvements are noted at Edu1 and Edu2.

2.5.2 Effects of System on Accuracy

We find that participants perceive kinemes more accurately at all education levels as seen in Figure 2.6. We additionally find statistically significant differences between systems at EDU1 ($p = 0.002$, $z = 2.774$) and EDU2 ($p < 0.001$, $z = 3.824$), but not EDU0. This implies that while kinemes may not be accurately understood if they were being viewed with no prior training or context, they are more accurately understood by users with a small amount of either. Therefore we find our hypothesis **Kineme_Accuracy** is supported for higher education levels, but not statistically supported for situations in which participants have no training for the systems.

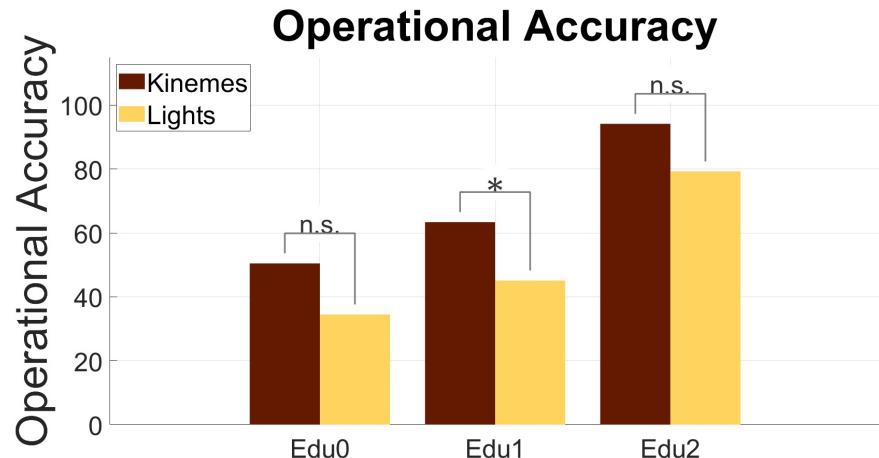


Figure 2.7: Operational accuracy of the kineme and light-based communication systems compared across education levels. Statistically significant improvements are noted at Edu1.

2.5.3 Effects of System on Operational Accuracy

Similar to the results of **Kineme_Accuracy**, we find that kinemes have greater operational accuracy at all education levels. However, the only statistically significant result is found at EDU1 ($p = 0.001$, $z = 3.0716$). This is unsurprising, as the confidence requirements of operational accuracy confine the answers used in this analysis to only the best. At EDU2, the answers which participants are confident in are typically correct

for both RCVM and the LED baseline, which lowers the difference between the two systems below the level of statistical significance. Nonetheless, kinemes retain a superiority to LEDs at all education levels (seen in Figure 2.7), supporting the hypothesis with statistically significant support at EDU1.

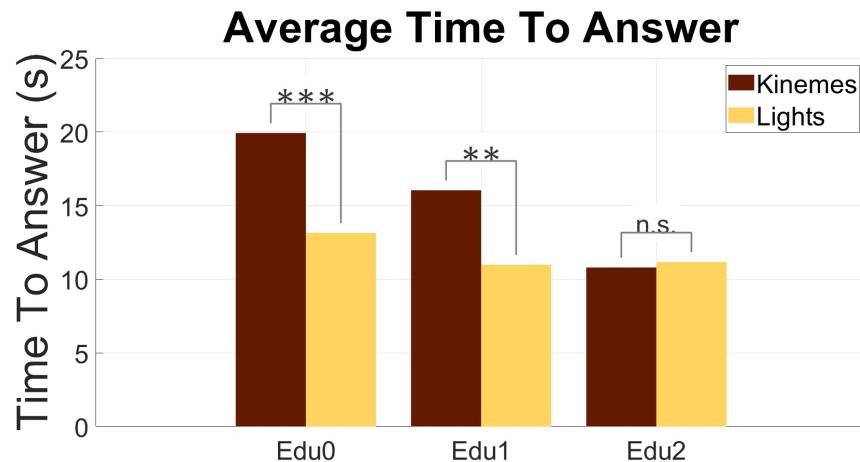


Figure 2.8: Time to answer of the kineme and light-based communication systems compared across education levels. The kineme system is significantly slower at Edu0 and Edu1, but not at Edu2.

2.5.4 Effects of System On Time To Answer

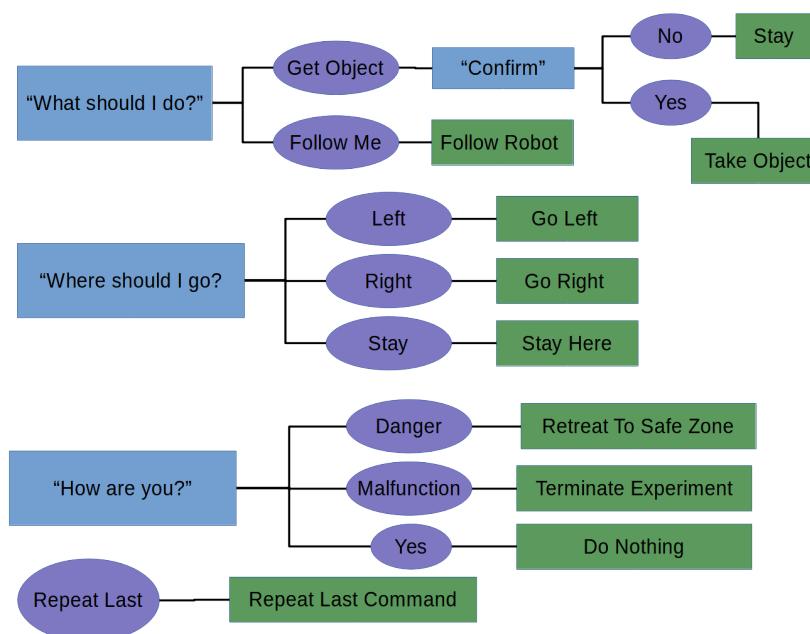
Lastly, in the case of **LED_Time**, we find statistically significant support for the hypothesis at EDU0 ($p < 0.001$, $z = -5.031$) and EDU1 ($p < 0.001$, $z = -3.398$), but not at EDU2. Simply looking at the values, we can see that understanding kinemes takes an average of almost 10 seconds longer than lucemes at EDU0, 5 seconds longer at EDU1, and is approximately even at EDU2. This can be seen in Figure 2.8. This gives support for our hypothesis that LEDs will take less time to understand, with the exception that this does not hold true at high levels of education.

2.5.5 Discussion

The performance of kinemes in this study, competitive with and even outperforming the luceme system (which is similar to previously used underwater communication systems), showed that kinemes could work well enough for underwater communication to be worth further investigation. While the study employed simulated data and a small sample size, clear benefits to the kineme system could be seen. Participant accuracy for kineme identification is greater than for luceme identification overall, at every education level, with variation in this trend for individual kinemes. For instance, kinemes which mimicked human gestures towards directions (K_{Ascend} , $K_{Descend}$, $K_{FollowMe}$) outperformed their equivalent lucemes, while lucemes which matched common uses of light (such as red lights for K_{Danger} or $K_{Malfunction}$) outperformed the kinemes. These differences in success based on the type of information being communicated formed the basis for our later multidimensional study, presented in Section 2.8. The results of this study formed a strong groundwork to build on and provided some initial understanding of the complex factors which comprise human perception of motion as meaning. While the results did not establish a clear superiority of kinemes over lucemes, kinemes performed as well or better than lucemes in most instances. This legitimized the approach for further study with a physical implementation on robots.

2.6 Study II: RCVM Implementation

After validating that RCVM worked in simulation, the natural next step was to evaluate our physical implementation. To this end, we performed a small pilot study with 8 participants, training them in the use of the RCVM system and then testing their ability to complete full-loop interactions. Given the small population size and complexity of the setup, this study was not intended as a definitive quantitative measurement of kineme effectiveness for communication. Rather, it serves as a confirmation that RCVM still works when translated to a physical robot, as well as when embedded in a full communication loop. This study was determined to be Not Human Research, exempting it from IRB oversight (reference numbers: 00004699, 00004700).



(a) Flow chart depicting the interaction process.



(b) Key: Boxes are gestures (blue) or actions (green) of the human, while the purple ovals are kinemes generated by the robot.

Figure 2.9: A flow chart of Study II's interaction loop, as described in Section 2.6.1.

2.6.1 Experimental Design

All participants were provided with educational material before the study began, to be completed at their own pace. This education was similar to education level Edu2 in the previous study (Section 2.4.1). The educational material familiarized them with the study layout and flow (using the image in Fig. 2.9), the gestures they would use for input, and videos of the kinemes on a simulated AUV (generated with the Gazebo simulator using our implementation). Participants were told to ask the Aqua AUV one of three questions, pay attention to the robot’s response delivered via kineme, and take the appropriate action. The question asked was communicated using a set of gestures based on relevant American Sign Language signs for ‘What should I do?’, ‘Where should I go?’, and ‘How are you?’, with another sign for ‘Confirm’. The users were told that Aqua would observe their gesture and automatically select a kineme to display. However, the kinemes were actually manually selected pseudo-randomly in secret by study staff. This selection was done manually in order to easily balance the number of each kineme shown throughout the study, and because gesture recognition in underwater environment is a challenging problem on its own. Therefore, the study is a Wizard-of-Oz study, where the study staff operate some aspect of the robot’s function without the knowledge of the user. Once the robot displayed a kineme, the participant verbally identified the action they would take, appropriate to the response they received. If their selected action was correct, the interaction was marked as completed correctly. If the participant forgot the correct action, leading to them taking an incorrect action or simply refusing to take an action, the interaction was marked as a failure. Therefore, the recorded “Accuracy” of each kineme is more appropriately considered the success rate of interactions including that kineme. After each interaction loop, users were asked for a confidence between one and ten (ten being high) on how accurately they had conducted their interaction. This confidence was recorded, along with the total time of the interaction loop and the sequence of question to kineme to action. For each robot, participants completed an interaction session composed of between ten and fifteen interactions. Two kinemes (K_{Lost} and $K_{ReportBattery}$) were not tested in this study, as they were under consideration for elimination at the time of the study.

Kineme	Accuracy	Avg. Time	Avg. Conf.
$K_{Affirmative}$	100.00%	27.00	8.00
$K_{Negative}$	50.00%	28.50	6.00
$K_{FollowMe}$	33.33%	40.22	5.34
$K_{MoveLeft}$	100.00%	29.71	7.14
$K_{MoveRight}$	60.00%	31.20	8.00
$K_{ObjectLeft}$	50.00%	35.50	5.00
$K_{ObjectjRight}$	71.43%	37.71	6.28
K_{Stay}	71.43%	30.57	7.72
K_{Danger}	44.44%	40.00	4.44
$K_{Malfunction}$	80.00%	41.20	4.40
$K_{RepeatLast}$	28.57%	36.57	4.86
Total Avg.	60.00%	34.38	6.1
<i>Avg. Study I @ Edu2</i>	<i>85.00%</i>	<i>11.00</i>	<i>8.00</i>

Table 2.2: Per-kineme results from Study II, along with the total average compared to education level 2 in Study I (Section 2.4.1).

2.6.2 Study Population and Education

The study population (N=8) was comprised of participants who were mostly age 21-34, 75% male and 25% female, 55.66% Asian and 44.44% White or Caucasian. Participants took an average of eighteen minutes to complete their education (max=33 minutes, min=9 minutes) and completed the education an average of fifteen hours before their session (max=52 hours, min=1 hour). Participants self reported on their experience with marine robots on an ordinal scale from zero to one hundred, rating themselves at an average of 49.20. Users were shown Figure 2.9 briefly before their first interaction to refresh their memory. The participants of this study do not represent expert users, but ones with some base knowledge of the system.

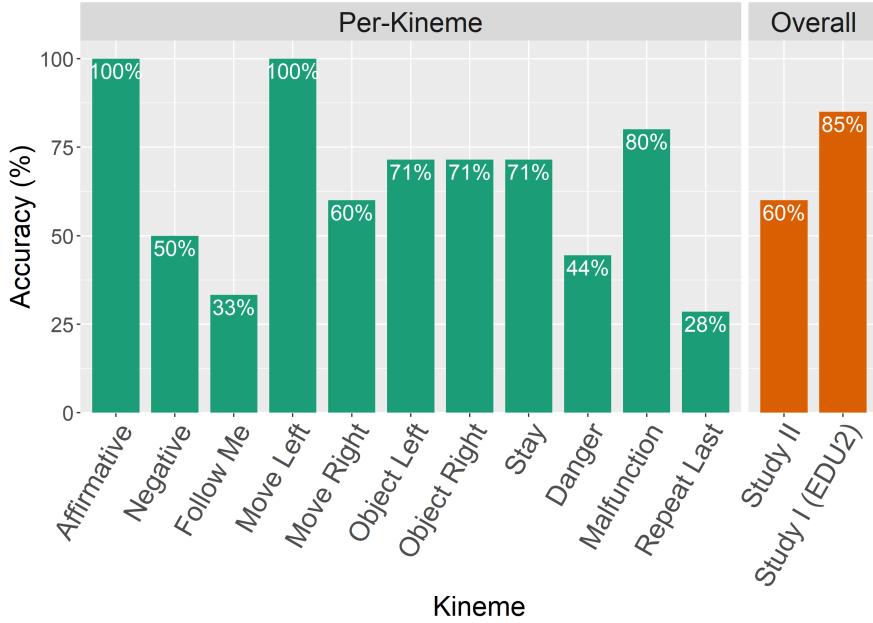


Figure 2.10: A graph of the per-kineme accuracy results of Study II, for easier visualization.

2.7 Results of Study II

The results from our first study (Table 2.2) show that RCVM continues to function relatively well when implemented on a physical AUV and placed in a full-loop context. While our overall accuracy is less than that of the best-trained participants Study I, we see several high accuracies on certain kinemes, particularly $K_{Affirmative}$, $K_{Negative}$, K_{Stay} , and $K_{IndicateMotion}$. We note that average times increased, but since our measurement of duration included the entire interaction loop, this is of less significance. Given the small sample size, it is difficult to draw any statistically significant results from this data, but it serves to validate the in-person performance of our kineme system: kinemes implemented for the Aqua robot achieved similar levels of accuracy to the original simulated kineme system that we are implementing. In addition, RCVM operated effectively as a part of a full interaction loop. Further evaluation of RCVM in a full communication loop, including analysis of the cognitive load placed on interactants, would be interesting to explore. However, such an evaluation would be premature, without an understanding

of how RCVM operates in comparison to other communication modalities, purely in terms of recognition accuracy.

2.8 Study III: Viewpoint and Content Effects on RCVM

With pilot studies completed, validating the baseline performance of physically implemented AUV kinemes, we turn our attention to the task comparing RCVM to other viable underwater communication systems. Understanding the comparative performance of RCVM to other robot-to-human communication options is critical in determining how to utilize RCVM in actual interaction loop designs, for both further research and field applications. To achieve this understanding of RCVM’s performance compared to other options, we designed a study comparing RCVM to the baseline systems described in Section 2.3.4 in terms of their efficacy from a variety of viewpoints and when communicating different types of information. Our participants, recruited using Amazon Mechanical Turk, were trained to use a randomly selected communication system, then tested on that system from a randomly selected viewpoint. This study design was submitted to the relevant Institutional Review Board for review and was determined to be Not Human Research, exempting it from IRB oversight (reference number: 00004695).

2.8.1 Viewpoints

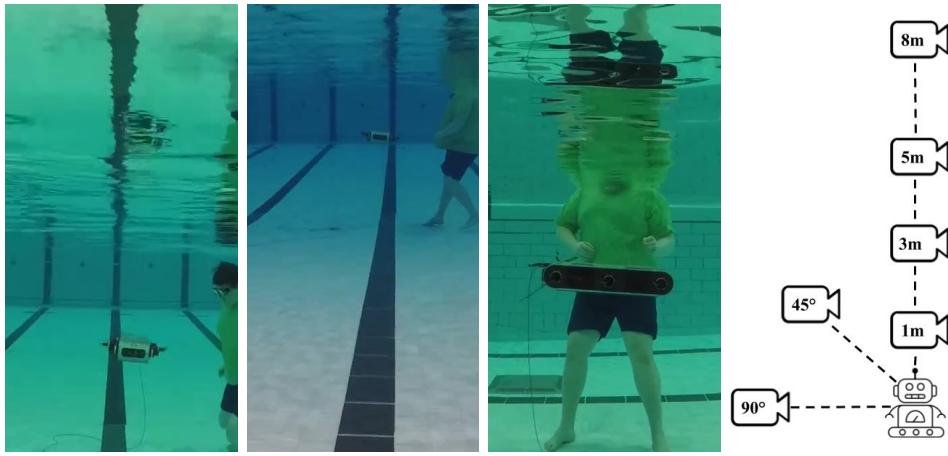
To prepare videos for our survey, each communication system was recorded from a variety of viewpoints. These viewpoints (illustrated in Fig. 2.11) were 3 meters, 8 meters, 3 meters at an angle of 45 °, and 3 meters at an angle of 90 °. In addition, an ideal viewpoint for each communication system was recorded to be used for the training, as well as being a possible viewpoint condition for testing. **This viewpoint, referred to as the EDU viewpoint, is defined as the closest distance to the robot at which all portions of every interaction phrase are visible, minimum of 1 meter.** Therefore, the EDU viewpoint is a distance of 1 meter for all of the baseline systems, but a 5 meter distance for the kineme system, because some kinemes must be viewed from a slight distance in order to see all of the movement. As each viewpoint is not viable for all systems, some were not tested; *e.g.*, the LCD screen cannot be seen at all from a 90-° angle for instance, so accuracy should be close to 7%, as all

attempts to identify the fourteen interaction phrases will essentially be random guesses. Additionally, the TTS systems should return similar results at the 3m viewpoints as the 45° and 90° viewpoints, as those viewpoints are also at a three meter distance and audio propagates evenly regardless of angle, given the position of our speaker. The viewpoint and robot/domain combinations which we tested can be seen in Table 2.3 along with the number of participants in each condition.

The viewpoints selected for this study represent a good sample of the possible distances and orientations viable for visual communication between an AUV and a diver. The orientations were selected by the assumption that divers and AUVs attempting to communicate with one another will be at least within a 90 degree orientation from one another. Distances were selected by considering the possible ranges of communication in the field. Visibility varies by levels of particulate matter, algae blooms, ambient light level, and depth. While visibility distances in the field range both higher and lower than the 1m-8m range tested in this study, this set of distances allows us to observe the effect of distance on communication, at distances that are realistic for a deployment (assuming that visibility is greater than 8m). Lower visibility distances than 8m would obviously reduce the effectiveness of all but the TTS system to nearly nothing past the distance of visibility, but diver-to-diver communication via hand signals would experience the same effect.

2.8.2 Video Recording

To capture videos of the systems at every viewpoint, several recording sessions were completed, using two GoPro™ HERO5 Black cameras, using a 1080p resolution in the linear aspect ratio at 24 frames per second. Due to scheduling constraints, the TTS system had to be recorded in a different pool than the rest of the Aqua systems. However, visual clips from the original pool were layered over the audio from the other pool to maintain a visual similarity. Due to this, the speaker is not visible in the video clips depicting the TTS system, but as the TTS system's communication is done over audio, this is not expected to impact results.



(a) Five meters. (b) Eight meters. (c) Ninety degrees. (d) Diagram

Figure 2.11: The viewpoints captured for this study, illustrated in three examples(a-c) and a diagram(d) showing the location of each viewpoint relative to the AUV.

2.8.3 Population Recruitment and Statistics

The participants recruited on Mechanical Turk were required to be in the United States, have never taken the survey before, and have completed more than 5000 Human Intelligence Tasks (HITs) with an approval rating of 97% or greater. The HIT posted on Mechanical Turk paid participants \$1.50 and was estimated to take an average of 12-15 minutes, meaning that users would be paid approximately minimum wage. Submissions were typically approved within 24-48 hours of completion. The only criteria for accepting Mechanical Turk work and paying the worker was that users spend an amount of time on the educational page equal to at least 25% of the duration of the education video. Users who spent less than a quarter of the video's duration before continuing were considered to have made a bad-faith effort and were rejected. They were, however, paid \$0.75 (half-pay) for their time. The line for inclusion in the dataset was set higher: only users who spent at least 75% of the video duration on the page were included in the analyses.

The resultant population was relatively diverse. We surveyed 130 participants, with 9 of them being excluded from all analysis due to short education video watch times. The final population of 121 participants was 62% male, 38% female, middle-aged ($M = 37$, $SD = 21$), had a variety of education levels (42.1% had a bachelor's degree), came from

System	Viewpoint					Total
	EDU	3m	8m	45°	90 °	
KINEME	7	8	7	7	7	36
TTS	8	10	10	0	0	28
LCD	7	8	0	6	0	21
LED	7	8	7	7	7	36
TOTAL	29	34	24	20	14	121

Table 2.3: Study conditions marked by whether (green) or not (red) they are tested, with participant numbers for each.

35 of the 50 states of the US, and were mostly employed (93.4%).

This study population is a decent sample of the US population, which was the target sample. Participants were not explicitly asked if they had diving experience. While divers are most likely to use the RCV system and evaluation on a sample drawn specifically from experienced divers would be useful, all of the communication systems evaluated in this study are designed to be easily interpreted and understood with little training. Thus, studying the performance of our communication systems on a sample from the general population helps us to evaluate the performance of these systems with minimal effects of prior knowledge of diving and underwater communication on that performance.

2.8.4 Education and Distraction Procedure

Once participants entered the survey and passed a bot check, they were randomly assigned a condition and directed to an education procedure. The education procedure for each condition consisted of a video composed of the fourteen interaction phrases in a set order from the EDU (ideal) viewpoint of the robot and system of the participant’s condition. Participants were asked to watch the entire video without skipping around and warned that their payment would depend on watching the entire video; however, they were permitted to leave the page at any time. As previously mentioned, only participants who spent at least 75% of the duration of their educational video on the page

System	Accuracy		Op. Accuracy		Confidence		Time (s)		
	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
OVERALL	KINEME	28.4	22.0	43.7	33.1	4.5	1.9	39.9	15.7
	TTS	21.2	24.8	29.4	40.2	4.4	2.6	20.6	9.3
	LCD	29.6	40.4	32.4	44.0	4.9	3.5	25.0	11.2
	LED	35.3	27.7	34.7	32.5	4.7	2.4	25.6	11.1
AT EDU	KINEME	41.8	13.3	77.7	18.9	4.5	1.7	44.7	10.0
	TTS	51.8	25.3	77.1	34.4	6.1	1.8	20.6	9.0
	LCD	82.7	21.8	91.6	15.2	7.4	1.9	19.3	2.6
	LED	70.4	23.8	61.6	36.8	6.9	2.7	30.8	15.4

Table 2.4: Mean and standard deviation of communication system metrics, averaged over all viewpoints and at the ideal viewpoint for each system. Bold values are the best (min/max respectively) mean for metric in group (overall or at EDU).

were included in the analysis. This education level falls somewhere between the two highest education levels described in Section 2.4.1(only meanings shown and meanings shown along with videos of kinemes), as the participant is shown both the communication system displaying an interaction phrase as well as the meaning of the phrase. However, since the participant was not being taught directly by study staff, the education provided in this study likely falls below the accuracy at the highest education level from Section 2.5. Following the education procedure, participants were asked to solve five ninth-grade level mathematics questions as a distraction procedure. Distraction procedures are a common method in psychology research used to induce forgetfulness in subjects. Some examples can be found in the 1950’s memory research of Brown [77] and Peterson [], or other more recent work on working memory by Waris et al. [78]. In our work, a distraction procedure is used to separate the training and testing phases of the study so that participants are less likely to be able to hold the entirety of the training they just completed in their short-term memory.

2.8.5 Testing and Evaluation

Each participant was shown videos of all 14 interaction phrases in a random order, using the communication system from the viewpoint indicated in their condition. Some

received the EDU viewpoint as their testing viewpoint, so they had the same viewpoint for education and testing. For each video, the participant was shown a video hosted on YouTube™ and asked to select its meaning from a drop-down menu. Participants were also given the option to select “Unable to select meaning”. If a meaning was selected, they were then asked what their confidence in their choice was, on an ordinal scale from 1 to 10 (10 being the most confident). Otherwise, the confidence question was not presented, and they were instead asked what made them unable to make a selection: forgetting the meaning, being unable to see the interaction phrase, the survey not displaying the video, or some other issue. The time from when the participant entered the webpage to when they left it was recorded, though participants were not told that it would be. Once participants had completed all of their videos, they were debriefed and given information on how to submit their responses for certification.

2.9 Results of Study III

We use the same metrics to measure system efficacy that we used in Section 2.4.2: **accuracy**, **operational accuracy**, **confidence**, and **time to answer**. Accuracy is the correctness of a participant’s answer, ranging from 0 to 100. Operational accuracy is the same metric but only considering answers also rated a 5 or higher in confidence, to simulate the answers that a user would be likely to act upon. Confidence and time to answer are simply the values recorded from the confidence question (0-10) and the time participants took to select a meaning for a video in seconds. Time to answer data was processed to remove outliers by discarding values greater than 150 seconds. This was set as the cutoff because for all interaction phrases, 95% of responses had a time to answer lower than 150 seconds (mean 95th percentile was 73.28 seconds). Durations of these outlier answers greatly exceeded 150 seconds (*e.g.*, 500 seconds or greater), which suggests that the webpage was left open while the participant briefly did something else.

The two metrics upon which we will perform statistical analysis are accuracy and operational accuracy. Shapiro-Wilk [79] tests were performed for accuracy $W = 0.84$, $p < .001$ and operational accuracy $W = 0.84$, $p < .001$, both finding evidence that data was not normally distributed, and direct observation of the data confirmed this. Due to this finding, we will perform the following hypothesis testing using the Kruskal-Wallis

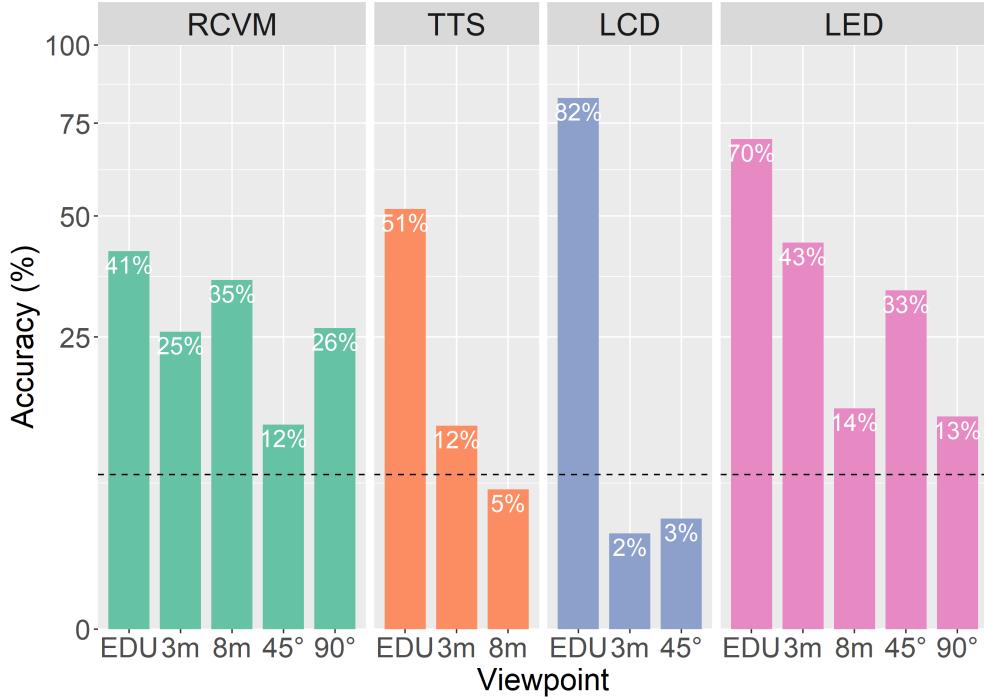


Figure 2.12: Accuracy results arranged by viewpoint and system. The y-axis is square root scaled to better display differences between conditions. The dotted line at 7% represents the expected accuracy of a random guess.

H-test [80]. Kruskal-Wallis is also referred to as one-way ANOVA on ranks and is a non-parametric equivalent to one-way ANOVA which does not assume a normal distribution of data. Tests were run at a confidence level of 99%, a significance of $\alpha = 0.01$.

2.9.1 Internal Validity

The Kruskal-Wallis H-test found no significant relationship between the percentage of their education video that a participant watched and their average accuracy in the testing phase $H(5) = 2.89, p = .717$. Further, we found no significant relationship between accuracy and gender $H(1) = 0.10, p = .750$. A correlation test using Spearman's method detected no significant correlation between accuracy and age $r(119) = -0.145, p = .112$. No threats to internal validity were detected, but participant recognition accuracy was considerably lower for all systems than expected.

The kinemes in Study II achieved 60% accuracy, while the highest RCVM accuracy achieved in this study was 41.8%, at the ideal viewpoint. This may be due to the fact that the study was conducted online via video, or due to low education absorption. However, the performance of the kinemes and LED systems at the EDU viewpoint is generally between the accuracy level reported at education levels 1 and 2 in Section 2.4.1, which is consistent with our education procedure’s expected success (as mentioned in Section 2.8.4). Higher accuracy at all viewpoints could likely be achieved by administering an education procedure which would train users until a certain competency level is reached. However, we believe the general trend of results to be correct, and statistically large effects should persist in in-person testing, as Study II demonstrates similar accuracy levels to Section 2.5 for in-person testing.

2.9.2 Overall Results

When considering differences between the four communication systems we tested, we are most interested in the effects on accuracy and operational accuracy. When considered over all viewpoints, none of the systems tested have statistically significant differences in accuracy $H(3) = 7.60, p = .055$ or operational accuracy $H(3) = 4.27, p = .234$. Due to the challenging nature of the underwater environment and the viewpoints at which they were tested, none of these systems have achieved high accuracy overall. However, when we consider the accuracy of these systems at different viewpoints, we see significant differences, despite the overall low accuracy, which indicate how RCVM performs compared to other communication options underwater (and often outperforms them).

2.9.3 Viewpoint Comparisons

The effect that viewpoint has on the accuracy of tested systems can be found in Table 2.5. While we see RCVM accuracy is the lowest of any system at the EDU viewpoint, once we move to the more challenging viewpoints, the TTS and LCD systems become entirely non-competitive, with accuracies near to that of a random guess (7%).

Kruskal-Wallis tests show that viewpoint has a statistically significant effect on every communication system, **with the exception of the kineme system** $H(4) = 8.94$,

Viewpoint	Accuracy		Op. Accuracy		Confidence		Time (s)		
	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	
KINEME	EDU	41.8	13.3	77.7	18.9	4.5	1.7	44.7	10.0
	3m	25.9	25.3	38.4	38.0	5.4	2.4	45.8	16.6
	8m	35.7	26.4	44.4	31.9	4.3	2.3	41.8	8.1
	45°	12.2	7.9	20.3	25.7	4.1	1.3	34.4	16.5
	90°	26.5	23.2	38.3	25.1	3.7	1.4	32.0	22.2
TTS	EDU	51.8	25.3	77.1	34.4	6.1	1.8	20.6	9.0
	3m	12.1	10.7	8.5	10.5	4.6	2.8	19.9	10.4
	8m	5.7	6.6	12.1	31.2	2.9	2.2	21.4	9.4
LCD	EDU	82.7	21.8	91.6	15.2	7.4	1.9	19.3	2.6
	3m	2.7	3.7	3.0	6.1	3.0	3.1	33.0	10.8
	45°	3.6	6.0	2.4	5.8	4.6	3.9	21.1	12.8
LED	EDU	70.4	23.8	61.6	36.8	6.9	2.7	30.8	15.4
	3m	43.8	18.9	41.1	30.6	4.6	2.0	22.8	10.0
	8m	14.3	17.0	14.9	19.5	4.3	2.1	27.3	10.0
	45°	33.7	20.9	40.0	26.6	5.4	1.7	27.8	10.1
	90°	13.3	12.7	15.0	27.8	2.4	1.7	19.9	8.8

Table 2.5: Mean and standard deviation of communication system metrics, for all evaluated viewpoints. Bold values are the best (min/max respectively) mean for metric in system group.

$p = .063$. The effect is most significant with the TTS $H(2) = 14.38$, $p < .001$ and LCD $H(2) = 14.71$, $p < .001$ systems, but is also present for the LED system $H(4) = 20.99$, $p < .001$. Considering the values shown in Table 2.5, it is apparent that non-EDU viewpoints reduce accuracy significantly for each of these systems. We also test operational accuracy with Kruskal-Wallis tests, which show that viewpoint affects operational accuracy for the LCD system $H(2) = 15.87$, $p < .001$ and the TTS system $H(2) = 10.75$, $p = .005$. However, there is no statistically significant difference in operational accuracy by viewpoint for the kineme system $H(4) = 10.78$, $p = .029$ or the LED system $H(4) = 10.33$, $p = .035$.

To summarize, through statistical testing we find that TTS and LCD communication begin to fail quickly at any challenging interaction viewpoint, while **LED** and

	Phrase Content	Accuracy		Op_Acc.		Confidence		Time (s)	
		Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)	Mean (μ)	SD (σ)
KINEME	Conversational	35.0	28.8	45.5	38.8	4.8	2.2	37.5	18.2
	Directional	23.3	27.3	29.3	36.3	4.3	2.2	41.7	20.7
	Status	23.9	21.3	42.2	41.3	4.1	2.1	42.0	17.2
TTS	Conversational	17.9	27.4	26.2	42.2	4.2	2.6	20.3	12.3
	Directional	23.6	27.2	30.5	41.7	4.6	2.9	19.7	11.0
	Status	22.9	30.2	29.4	43.1	4.5	2.7	21.7	11.9
LCD	Conversational	28.6	42.7	31.0	45.3	5.0	3.5	22.2	10.4
	Directional	32.4	38.7	34.5	42.8	5.0	3.6	28.0	14.6
	Status	28.6	42.7	32.4	47.1	4.7	3.5	25.5	14.1
LED	Conversational	40.6	31.9	38.8	39.3	4.8	2.6	28.7	16.1
	Directional	40.6	33.6	39.8	38.0	5.0	2.7	24.0	13.2
	Status	25.0	26.3	23.0	33.1	4.5	2.6	24.6	11.6

Table 2.6: Mean and standard deviation of communication system metrics, separated by phrase content. Bold values are the best (min/max respectively) mean for metric in system group.

Kineme communication are more viewpoint-invariant, particularly kinemes. Since the accuracy of the kineme system is above that of a random guess (7% accuracy) at challenging viewpoints, we suggest that this shows that kinemes are more viewpoint invariant than other systems, though the LED system is a strong competitor.

2.9.4 Content Comparisons

We chose to study the effect of message content because message content is known *a priori*, meaning that if a communication system shows an affinity for communicating certain types of messages, we can autonomously switch to using that system to get the message across most effectively. For this experiment, we consider three categories of our interaction phrases (see Section 2.2 and Table 2.1): Conversational, Directional, and Status phrases. In Table 2.6, we can see that kinemes and LEDs have similar accuracies for Conversational and Status interaction phrases, but the accuracy of LEDs is higher for Directional phrases. This is unexpected given the spatial nature of kinemes, but analysis of kineme identifications suggests that participants may have been confused

as to whether “left” referred to their left or the robot’s left, and vice versa for right. A Kruskal-Wallis test does not show a significant effect on the accuracy of Directional phrases based on which communication system is used $H(3) = 6.40$, $p = .094$, however. Simply observing the higher accuracy of the LED system suggests that LED codes may be more effective in expressing directional information than kinemes, but this difference is not statistically significant.

No communication system is found to have statistically higher accuracy than others when considering the accuracy of Status phrases $H(3) = 1.48$, $p = .686$, but system type does have a significant effect on the accuracy Conversational phrases $H(3) = 11.61$, $p = .009$. Post-hoc analysis with Dunn tests using a Bonferroni-adjusted alpha level of 0.0017 (0.01/6) was used to compare pairs of systems. No comparisons were significant after Bonferroni adjustment (all $ps > .012$), but the TTS system under-performs both the kineme and LED systems on Conversational phrases. This may be due to the fact that Conversational phrases are typically short for the TTS system, meaning that they might be entirely missed or easily misidentified at challenging viewpoints, while the longer phrases of the kineme and LED systems provide more opportunity to understand the phrase.

To summarize, while we had hoped that Directional accuracy of the kineme system would be high, no statistically significant effect is detected for Directional accuracy based on system type. Not effect is detected for Status phrases either, but Conversational accuracy is affected by system type (though no system is shown to be better than others in post-hoc analysis).

2.10 Conclusion and Future Directions

In this chapter, we presented our research on motion-based AUV communication with RCVM. After a discussion of the design and implementation of kinemes, we presented an initial study evaluating the fitness of RCVM for underwater communication by comparing simulated kinemes to a 9-LED system. This was followed by a physical implementation of RCVM for the Aqua AUV, which was evaluated in a closed-water setting, demonstrating reasonable accuracy for use in the field. Finally, our third study compared RCVM to a 3-LED, LCD, and Audio communication system, over a large population

of online participants. This study suffered from some issues with its administration online rather than in person but demonstrated that RCVM outperforms other systems in difficult and distant viewpoints. Overall, the three studies presented in this chapter demonstrate that RCVM is an effective method of natural robot-to-human communication, robust to interactant viewpoint change, and can outperform light and sound-based communication methods. While our later research on light and sound-based communication has produced much more successful methods of communication using those vectors, the demonstration of RCVM’s effectiveness in the studies presented in this chapter remains valid. It does not, however, provide an indication of how RCVM might perform in comparison to HREye and SIREN.

Possible Areas of Future Exploration

At the end of most chapters of this thesis, we will briefly provide a few suggestions for future directions of research on this topic, based on our experience and the results of the studies presented within the chapter.

Motion Overlays for Passive Communication

A strong thread of research in motion communication “overlaid” on practical motion exists for manipulator robots [71] and aerial robots [68]. The benefits are obvious: instead of dedicating motion to communication, the normal swimming of an AUV could be rendered communicative by modifying it in some way. The likely information density of these trajectories is low, but perhaps simple information such as battery level, important object detections, or affective information could be communicated. This is likely to be more effective on biomimetic AUVs such as the Aqua AUV, though propeller-driven AUVs such as LoCO have the advantage of the ability to make quick changes to heading or velocity.

Atomic Kinemes for Dynamic Communication

The set of meanings implemented for RCVM is a small, research language, which may be missing symbols that deployments may require. Additionally, new concepts may come to

light during deployments that require new kinemes and having the ability to communicate these without a pre-defined kineme could be useful. A number of approaches could be applied to this problem, but the most likely to be successful is to create a more atomic kineme, where each kineme is a smaller concept rather than a phrase. For instance, with a K_{Safety} , $K_{Battery}$, $K_{Confidence}$ and kinemes that represent quantity, dynamic motions could be created on the fly to communicate as many values of safety, battery level, and confidence as the valued kineme could represent, without having separate kinemes for each. This has already been explored to some extent with the existing $K_{Directional}$ and $K_{BatteryRemaining}$, but could be expanded into a more fully featured language of combined atomic kinemes.

Cross-Form Gesture Mimicry

Another possible modification to kineme design comes from the idea of motion mimicry. While our design process for kinemes involved mimicking human motions such as nodding and shaking the head, this was only at the level of using these general motions as an inspiration. Kinemes could also be sourced from actual human gestural motion, using either motion capture or pose estimation systems. After recording and processing gesture trajectories, a mapping from body parts to robot motion would have to be established, followed by solving the inverse kinematics for the trajectory. This would allow exact mimicry of human gestures, but the utility would be limited by the quality of the human-to-robot motion mapping functions.

Acquiring Understanding of Motion

An outstanding question from our study of motion-based communication for AUVs is how best to teach practitioners to understand the robot. While the intuitiveness of kinemes is always a design goal, it is unlikely that a diver will be able to understand each kineme without training, especially as the number of symbols in a language increases. Exploring the process of teaching kinemes to divers, particularly in a longitudinal manner, would yield useful insights into the most effective ways not only to design kinemes, but to ensure that users can effectively learn to understand them.

Chapter 3

Emitted Light For AUV-To-Human Communication

Light is a frequently used method of information display in digital devices, from power or status lights to simple dot-matrix displays, all the way to high-definition displays. It is particularly useful in nonverbal communication contexts, where the color, brightness, and flash rate of light can be mapped to complex concepts. This, combined with the fact that it can be perceived from long distances (assuming adequately bright sources and good visibility), makes it an excellent option for AUV-to-human communication. The fact that it is one of the few methods that appear in the literature besides the use of digital displays reinforces this and was the reason why it was selected as a comparison point for RCVM in the previous chapter. Neither the methods present in the literature nor our simple baseline systems have explored the full potential of light for underwater communication. They are all extremely simplistic and have limited expressiveness, particularly when compared to LED-based communication systems seen in terrestrial and aerial robots.

In this chapter, we present a new method for AUV-to-human communication using emitted light: the expressive light devices called HREye(s). HREyes allow for communication at a further distance than digital displays, are more intuitive and expressive than previously proposed light-based communication, and provide a new capability for AUVs: gaze indication. In the subsequent sections, we first discuss the background of this work



Figure 3.1: The LoCO AUV informs a diver that it is ready to follow them, using the active luceme $L_{FollowYou}$.

with further discussion of existing UHRI methods, brief summaries of research on using light for HRI, and different methods of communicating gaze in terrestrial robots. Following this, we present the design of the HREye devices, their evolution from earlier light-based methods we have proposed, and the software used to control them. The HREye devices communicate through **luceme(s)**, sequences of colored light with semantic meaning. We next present a sixteen-symbol luceme language with content based on the gestures commonly used for diver-to-diver communication. Finally, to evaluate the effectiveness of the HREye devices and the lucemes we have defined for them, we present Study IV, a human study with fourteen participants evaluating the ability of trained participants to recognize active lucemes and the perceived gaze direction of ocular lucemes. This study demonstrates high levels of accuracy in participant recognition of active lucemes (83%, 92% with high confidence) and reasonable success (21° avg. error) in the communication of gaze direction using ocular lucemes.

Definition: Luceme

In either active or ocular mode, the HREye device is designed to produce animated sequences of colored light called **lucemes**. These sequences include the color, order, duration, and in some cases the intensity of illumination. The word luceme comes from the Latin *luc*, meaning “light” combined with the Greek *eme*, a unit of linguistic structure. A luceme is designated by the symbol $L_{Meaning}$, where “Meaning” is a simple description of the meaning. Active and ocular lucmemes are displayed in Figures 3.4 and 3.5 respectively.

3.1 Background: Light-Based Communication

3.1.1 Emitted Light For HRI

Power and status indicator lights have been integrated into the design of robots since the creation of the first robots. Modern humanoids such as Pepper [81] and Nao [82] often use colored lights around their cameras/eyes to indicate the robot’s state, be it power, error, or emotion. However, the use of light for HRI extends past simple indicator lights. Light was used as a mediating signal to improve speech interactions by Funakoshi *et al.*, [83] and used to express emotion by Kim *et al.*, [84]. More recently, Szafir *et al.*, [85] used light to communicate directionality for aerial robots, while Baraka *et al.*, [86, 87] developed a variety of ways to use light to communicate state and movement direction for mobile service robots. Song *et al.*, [88, 89] applied similar approaches with designs inspired by bioluminescence.

3.1.2 Gaze Cues

Gaze is one of the most important implicit vectors of communication between humans [90–92], used intuitively by people of all ages across all cultures. This highlights the importance of gaze for human-robot interaction, particularly in coordinating collaborative work. The use of gaze as a vector of implicit communication in human-robot interaction has been well studied [93] for use in establishing shared attention [94, 95], managing handovers [96, 97], coordinating task roles [48, 98, 99], creating persuasive robots [100], and smoothing social interaction [101]. Some robot designers have even explored the

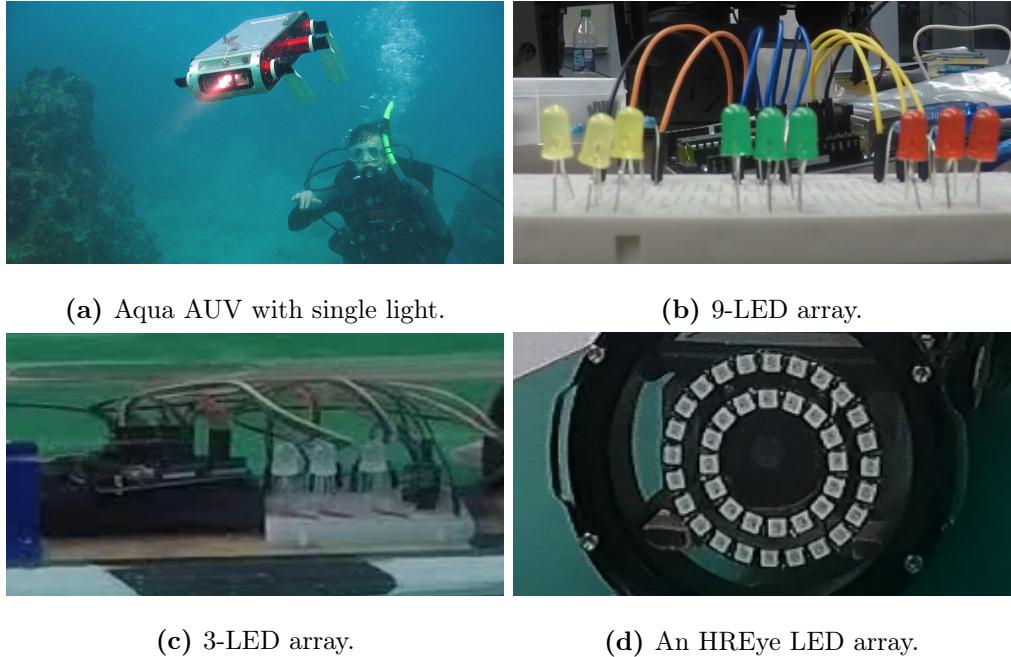


Figure 3.2: The evolution of AUV LED communication systems over time.
(Figure 3.2a credit: McGill MRL & Ioannis Rekleitis)

development of camera-eye hybrids [102, 103] to enable gaze interaction jointly with perception. Despite the depth of this field, we are unaware of any attempts to provide gaze cues from an AUV, making the ability of the HREye to communicate gaze direction a first for underwater HRI.

3.1.3 AUV Light Communication

In the past two decades, the majority of underwater human-robot interaction (UHRI) publications have focused on enabling human-to-AUV communication with fiducial markers [10], hand gestures [2, 45, 104], and remote control [17]. The few works that have explored the inverse question of AUV-to-human communication have primarily focused on the use of digital displays [10, 15], remote control devices [17], and low-complexity light systems [16, 17]. Our work in Chapter 2 introduced motion-based communication for AUVs [105], as well as a variety of light-based communication methods (briefly discussed in Section 2.3.4, which we now briefly survey).

Single-Light Signals

The first mentions of designing emitted light communication for AUVs use one functional light (comprised of multiple LEDs, but controlled together) as a flashing signal. An example is shown in Figure 3.2a. In Verzijlenberg *et al.*, [17], the light is mentioned as an option for confirming that the AUV has received a command, while Demarco *et al.*, [16] design a 4-symbol language similar to Morse code. Neither approach was quantitatively evaluated.

9-LED Array

Building on these suggestions, we proposed a communication device based on an array of 9 LEDs (Figure 3.2b) [105]. The goal was to add color and position as aspects of encoding, on top of the flash rates suggested in previous work. When observed by well-trained participants, the 9-LED system achieved a recognition accuracy of 60% and an average response time of 10s.

3-LED Array

The idea of our 9-LED system was further refined in a 3-LED version (Figure 3.2c) [106] which uses full RGB diodes, enabling those three lights to produce not only the red, yellow, and green lights of the 9-LED system, but also blues, purples, and many other colors. With well-trained participants, the 3-LED system achieved an accuracy of 70.4% with an average response time of 28.3 seconds (inflated by the nature of online administered evaluation).

3.2 HREyes: Biomimetic Light-Based Communication

Our light-based communication systems built upon the suggestions of earlier researchers and demonstrated that light-based communication was possible for AUVs, while providing a comparison for RCVM. However, our previous systems had only achieved 70% accuracy at most, were not intuitive at all, and were difficult to create light displays for, given their limited state space. To provide collaborative AUVs (co-AUVs) with a highly communicative and more intuitive light-based communication method, we present

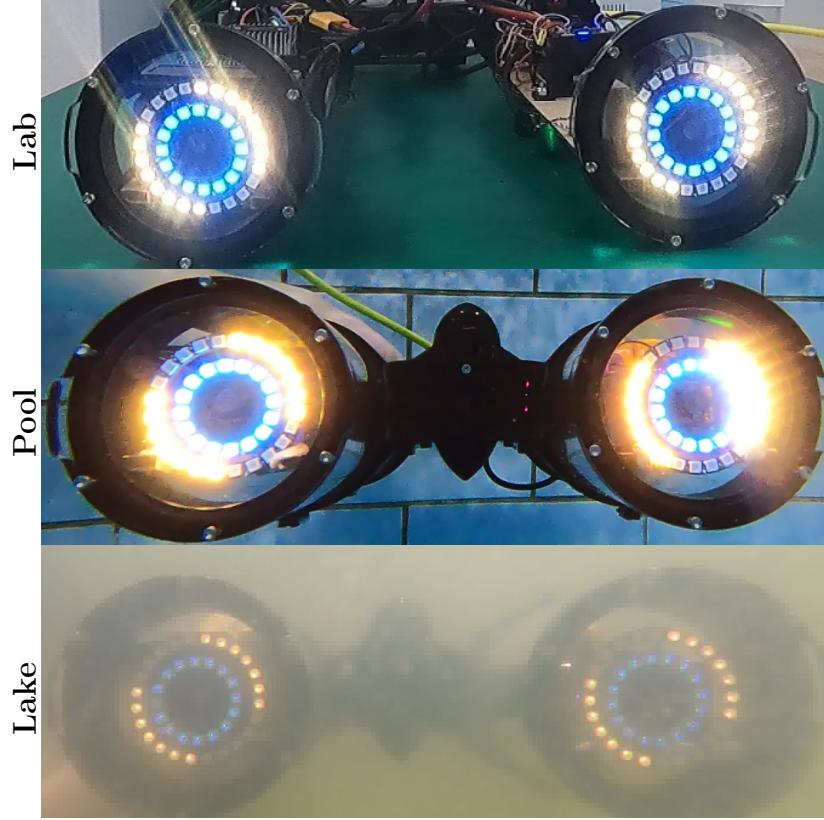


Figure 3.3: HREyes in LoCO AUV, demonstrating $L_{FollowYou}$ out of the water, in a pool, and in a Minnesota lake.

the **HREye**, a biomimetic LED device for AUV-to-human communication. HREyes are comprised of 40 individually addressable RGB light-emitting diodes arranged in two concentric circles. Built by daisy-chaining two Adafruit NeoPixel rings, one with 24 LEDs and one with 16 LEDs, an HREye can be used on its own or synchronously with another. In our implementation, the HREyes are controlled using the Robot Operating System (ROS) and a microcontroller that drives the NeoPixel rings. Two HREyes integrated into the LoCO AUV [29] can be seen in a variety of environments in Figure 3.3.

3.2.1 Design Inspiration

When designing the HREye, the goal was to create a light-based communication device capable of displaying complex light displays, producing an amount of light useful for

illuminating the AUV’s environment and indicating direction in an intuitive manner. To enable the goal of complex displays, we chose to increase the number of LEDs from previous systems and arrange them in a structure which enables the creation of complex shapes. Next, to illuminate the scene for an AUV’s cameras, the lights were placed near the AUV’s camera. Many participants in previous studies have noted that the cameras of AUVs such as LoCO and Aqua seem to be the “eyes” of the AUV. This gave rise to the idea for HREyes: arrange a large number of LEDs around the AUV’s camera and mimic the appearance of the human eye. This helps to fulfill the final design goal by allowing the HREye to mimic human gaze cues.

3.2.2 Control Modalities

An HREye’s state can be represented by a set of 40 tuples with 4 values (red, green, blue, alpha) across time. Each tuple maps to a specific LED on the two rings of the HREye. Each HREye is controlled by a ROS node (the *hreye_driver*) running on a microcontroller that receives messages containing an array of 40 tuples, which represent the instantaneous state of HREye. The microcontroller processes these messages and applies the appropriate states to the LEDs. A secondary ROS node (the *hreye_controller*) is responsible for creating these state vectors. The *hreye_controller* has three modes: active, ocular, and functional. The lucemes which comprise active and ocular modes will be discussed in the following section. In functional mode, the HREye device simply acts as a functional light, providing artificial lighting for the AUV at a variety of intensities and color grades. This mode is not discussed further, as its uses are purely functional.

3.3 Design of Lucemes

3.3.1 Active Lucemes

In active mode, the HREye device produces lucemes with specific semantic meaning. The current luceme versions are shown in Figure 3.4. The symbols of this language were selected by extracting a set of common communication phrases from diver gestural languages through a process of coding and clustering. A luceme was designed for each of the concepts extracted from these languages. These lucemes utilize a color and structure

Luceme\Meaning	Visuals
$L_{Affirmative}$ Yes, Okay.	
$L_{Negative}$ No.	
L_{Danger} Danger in the area.	
$L_{Attention}$ Pay attention to AUV.	
$L_{Malfunction}$ Internal malfunction.	
$L_{WaitCMD}$ Waiting for instructions.	
L_{GoLeft} Go left/AUV going left.	
$L_{GoRight}$ Go right/AUV going right.	
L_{GoUp} Go up/AUV going up.	
L_{GoDown} Go down/AUV going down.	
$L_{WhichWay}$ Asking for directions.	
L_{Stay} Stay where you are.	
$L_{ComeHere}$ Come to the AUV.	
$L_{FollowMe}$ Diver can follow AUV.	
$L_{FollowYou}$ AUV will follow diver.	
$L_{BatteryLevel}$ Battery level is...	

Figure 3.4: Selected active lucemes, demonstrated in a laboratory.

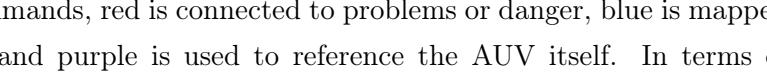
Luceme \ Meaning	Visuals				
$L_{Ocular-Blink}$ Mimicking a blink.					
$L_{Ocular-Squint}$ Squinting or focusing.					
$L_{Ocular-EyesWide}$ Eyes widen.					
$L_{Ocular-90}$ Gaze cues at 90°					
$L_{Ocular-135}$ Gaze cues at 135°					

Figure 3.5: Selected ocular lucemes, with two examples of directional gaze (angles in Cartesian coordinates).

mapping protocol intended to improve recognition accuracy. First, for color: yellow relates to directional commands, red is connected to problems or danger, blue is mapped to general information, and purple is used to reference the AUV itself. In terms of structure, there is a shared symbol used for L_{Danger} and $L_{Attention}$ to indicate time-critical information, a pulsing illumination used for $L_{WaitCMD}$ and $L_{Malfunction}$ to indicate a state which requires diver intervention, and a circular animation with one segment of light following another to connect the $L_{FollowMe}$ and $L_{FollowYou}$ lucemes. Together, these color, shape, and motion mappings result in lucemes with related meanings having similar appearances, which allows interactants to more easily memorize them.

3.3.2 Ocular Lucemes

Ocular lucemes are much more uniform than their active counterparts. All ocular lucemes follow the same structure: the inner ring is illuminated in pink (recalling the iris), with the outer ring illuminated in white (recalling the sclera). The color pink was selected as it had not been used in many lucemes, was similar to the color used to reference the robot, and would be quite noticeable in the water. The ocular lucemes created (depicted in Figure 3.5) include a steady state luceme, lucemes for blinking, squinting,

and widening eyes, and a set of gaze cues with 30° granularity. Only the gaze direction ocular lucemes have been evaluated in human studies.

3.4 Study IV: Communication and Gaze Indication with HREye

In the following sections, we describe a study that tests the recognition accuracy of active and ocular lucemes by evaluating the recognition of human interactants observing the lucemes in a pool. This study was approved as human research by the University of Minnesota's Institutional Review Board (reference number: 00015327).

3.4.1 Study Design

The study of HREye performance is based on human recognition of HREye lucemes in a pool environment. To achieve this, we recruited participants, trained them to a point of competency in the use of the HREye system, then asked them to identify active lucemes as produced on the LoCO AUV's HREyes in a pool. The results from this evaluation serve to demonstrate the performance of the active lucemes of HREyes when being shown to a trained interactant. Following this, we also asked each participant to identify the gaze direction of the ocular luceme gaze cues, which they had not been shown previously. To provide a point of comparison to the HREye-trained participants, we also collected data on human recognition of information passing from the AUV's OLED display. The participants for this condition were also asked to identify the HREye active and ocular lucemes, providing more data on ocular luceme comprehension along with insights into the ability of untrained participants to recognize active lucemes.

3.4.2 Administration Procedures

Training

A total of 14 participants were recruited for the study via student email lists. After collecting participant consent and administering a short demographic survey, participants were trained to use either the HREye communication device or an OLED display. This

was done by displaying a video of 4-5 lucemes (or OLED phrases with equivalent meanings) along with their meanings on screen, then testing the participants on the content of the video. Once this had been done for all lucemes, a final competency test was administered, asking participants to identify each luceme/OLED phrase. Participants who correctly recognized at least 12 out of the 16 phrases shown passed the competency test, and were scheduled for a pool evaluation, which was completed approximately 7 days after training.

Pool Sessions – HREye Condition

In the pool evaluation portion of the survey, participants were given a quick refresher of the list of luceme meanings. Following this, participants entered the pool approximately 2 meters from the LoCO AUV and shown the lucemes of the HREye system in a randomized order. For each luceme, the participant dove under the water, observed the luceme display, then surfaced and verbally reported the luceme's meaning. The time from the beginning of the luceme until the beginning of the participant's answer was recorded, along with the participant's confidence in their answer (0 - 10). Once all active lucemes had been tested, participants were asked to continue the same process while being shown ocular lucemes for gaze indication (which they had not been shown during training).

Pool Sessions – OLED Condition

Participants in the OLED condition followed a similar procedure. They were first asked to swim the 2 meter distance to LoCO three times, after which they were asked to observe the OLED displays from LoCO at a closer distance. This allows the participant's average swim time to be added to each response, simulating how long it would take the participant to determine the OLED's content if they had to swim to it, as the OLED is unreadable at 2m. Most OLED condition participants were then asked to perform the same procedures as the luceme condition participants.

Debrief Process

Following their completion of the pool evaluation, participants were asked to complete a quick debrief survey which consisted of a modified Godspeed [107, 108] questionnaire

and a set of NASA Task-Load Index (TLX) [109, 110] questionnaires, one for active luceme/OLED recognition and one for ocular recognition. The Godspeed questionnaire measures participant opinion on the AUV they were shown, while the NASA TLX questionnaires measure the amount of mental, physical and temporal demand that the luceme recognition placed on participants. With the debrief survey completed, participants were given a gift card with a value of 10 USD.

3.4.3 Analysis Procedures

Because participants had answered questions about luceme meanings freely, their responses had to be transformed into a quantifiable format for analysis. For active lucemes, three raters were given a list of active lucemes shown to participants, matched with the meanings participants reported. The three raters then scored each pair with a correctness score between 0 and 100. An inter-rater reliability analysis using Fleiss' Kappa [111] was performed to evaluate consistency between raters. The inter-rater reliability was found to be $\kappa = 0.74$, which is generally taken to mean that there is substantial agreement between raters [112]. Since rater consistency was high, the rater's scores were averaged to the final recognition accuracy score. The raters also transformed ocular luceme responses from various types of input (*e.g.*, "To the upper left") into an angle of reported gaze ($\kappa = 0.87$). These translations were also averaged to create a final reported angle. To complete the analysis, we utilize common metrics in AUV-to-human communication research [46]: accuracy (0-100 recognition rate), operational accuracy (0-100, recognition rate of answers with a confidence ≥ 6), confidence (0-10, participant reported), and time to answer (time between luceme beginning and participant answer). In the case of OLED participants, the average time it took them to swim to LoCO was added to the time to answer.

3.5 Results of Study IV

3.5.1 Demographics

Our study population was comprised of 14 people. Ten participants were trained in the HREye condition, the remaining four were trained on the OLED device, with three of

those four also evaluating the HREye system’s active lucemes without training. Ten of these were students, the rest were working or looking for work. When asked to self-identify their gender, nine identified as male, three as female, and one identified as non-binary. The majority of participants were between the ages of 18 and 24 and had lived in the US for at least 3 years. Eight participants indicated that they had experience with robots, four indicated some level of scuba diving experience, and five indicated a familiarity with some kind of sign language. Nine participants self-reported nearsightedness and none reported colorblindness, which was confirmed to be accurate by a self-administered Ishihara test [113].

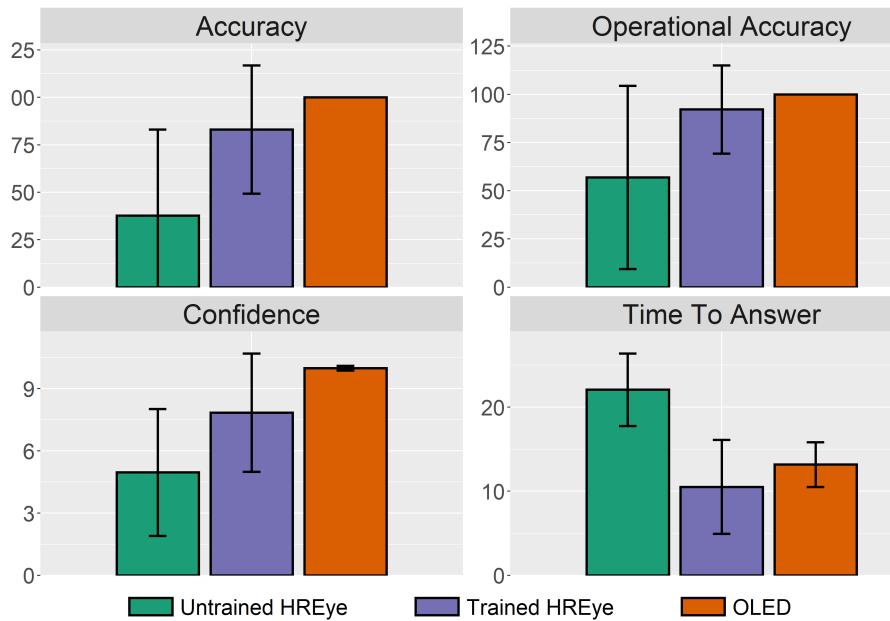


Figure 3.6: Comparison between the HREye, OLED, and untrained HREye conditions in terms of all metrics.

3.5.2 Comparing Across Conditions

As shown in Figure 3.6, the trained HREye participants identified lucemes with a reasonable 83% accuracy. This is lower but comparable to the accuracy of OLED trained participants (100%). The lower accuracy is to be expected as the use of the OLED device requires no learning or memorization, simply the ability to read. However, the

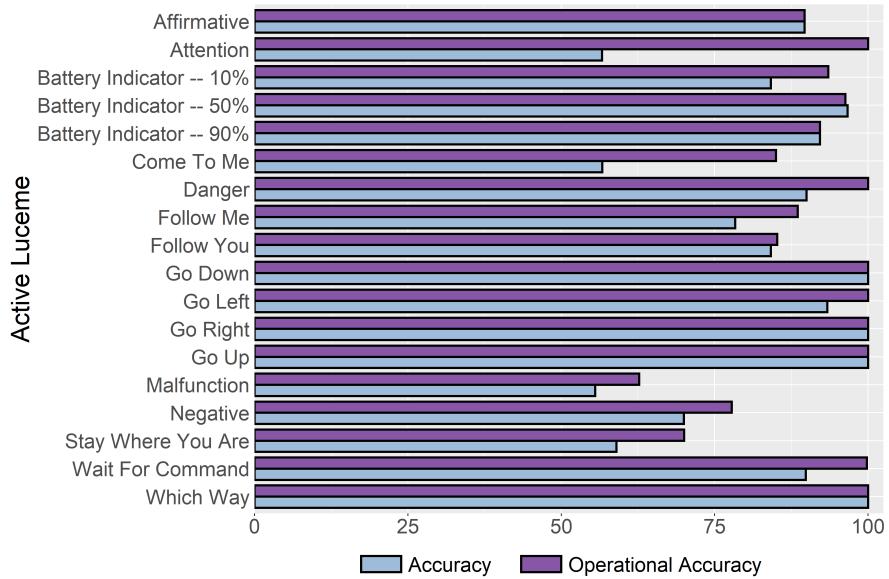


Figure 3.7: Recognition accuracy and operational accuracy for participants trained to recognize active lucemes.

recognition accuracy of active lucemes, higher than any previous light-based communication system and comparable to that OLED display phrases, supports the idea that the HREye devices could be effectively used in actual deployments. When considering the accuracy of untrained HREye participants, we see higher accuracy than might perhaps be expected, especially considering previous results on the untrained use of light-based communication systems in [105]. The accuracy (37%) and operational accuracy (57%) of untrained recognition indicate that HREye lucemes are intuitive to understand. Further analysis of the performance of specific lucemes indicates that the lucemes most correctly identified by untrained participants were directional lucemes (L_{GoLeft} , etc), $L_{BatteryLevel}$, and L_{Danger} .

3.5.3 Recognition of Active Lucemes

Trained participants identified active lucemes with an overall accuracy of 83%. As shown in Figure 3.7, the directional lucemes all achieve high accuracy. In terms of operational accuracy, also shown in Figure 3.7, participants achieve an even higher rate of 92%, with a number of lucemes even hitting 100% operational accuracy. Average

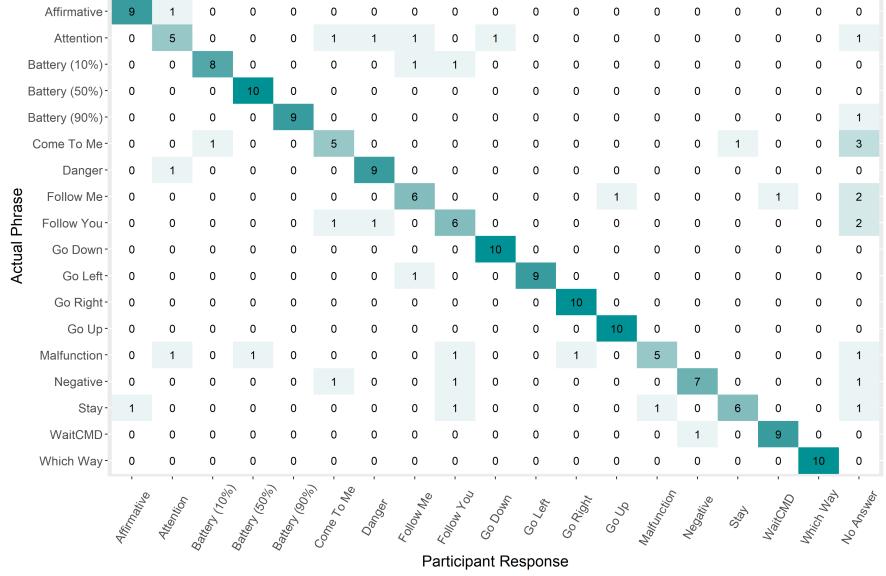


Figure 3.8: Confusion matrix of participant identifications of active lucemes.

participant time-to-answer is 10.5 seconds, with the $L_{Attention}$ and $L_{Malfunction}$ being two of the longer response times. The level of accuracy (and particularly operational accuracy) demonstrated in this study is sufficient to expect that similar lucemes could be used effectively in the field. This is consistent with the relatively lower accuracy of these lucemes, indicating some confusion on those lucemes, which can be more clearly seen in Figure 3.8. Note that in the confusion matrix, it is obvious that participants are confusing $L_{FollowMe}$ and $L_{FollowYou}$, which have similar animations with the only difference being the color of the inner ring. Similarly, lucemes which heavily feature the color red, such as $L_{Negative}$, L_{Danger} , and $L_{Malfunction}$ are confused for one another. This suggests that further separating lucemes in color space and ensuring that lucemes with similar animations have large differences in color may improve reduce confusion.

3.5.4 Ocular Lucemes

Participants correctly identified gaze cues with high accuracy, despite being untrained in this aspect of the system. While some participants had confusion about the gaze cues initially, all participants realized the key idea: the purple center was the pupil and the motion of gaze lucemes was meant to indicate the motion of the pupil in a

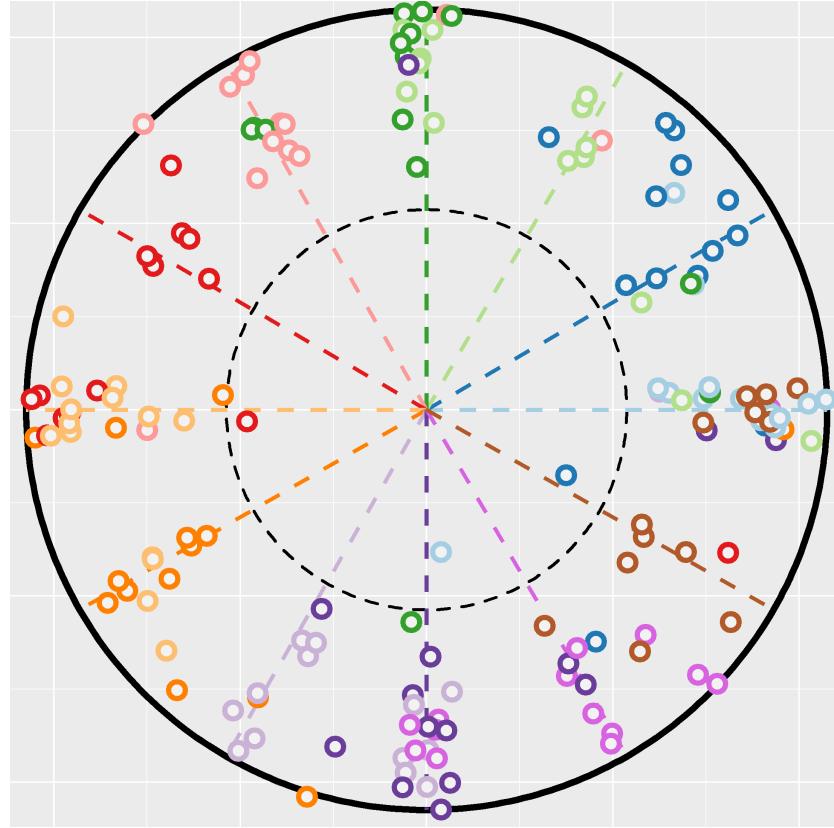


Figure 3.9: Participant interpretation of gaze lucemes. Points are positioned radially by participant answer and colored by true gaze position. Distance from the center represents participant confidence, with a dashed circle at 5/10 confidence.

direction. Overall, participants interpreted ocular lucemes with an average error of 21° , as visualized in Figure 3.9. Some of this angle error may have been due to the fact that in one session, the HREye in the LoCO AUV’s right enclosure began to slip and rotate, rotating almost 45° before the issue was detected. Remaining participants in that session were only shown gaze cues on the left HREye, and the issue was rectified with proper mounting before the following session. The actual error is likely lower. This is due to the fact that in one pool session, one of the HREyes came loose and rotated by almost 45° before the issue was detected and fixed. Nonetheless, even with this confounding issue (which affected 3 participants), interactants seem to be able to intuitively understand the gaze direction of an AUV using the HREye ocular lucemes. Post-study interviews

with some participants suggested that participants did not feel the robot was actually looking in the directions it was indicating, however. Therefore, it remains to be seen how well ocular lucemes achieve grounding for tasks in underwater environments.

3.5.5 Participant Opinion

In terms of results of the Godspeed and NASA TLX surveys, which ten participants completed, the LoCO AUV was rated as neither particularly anthropomorphic (1.9-2.7 on average sub-scores out of 5) or animated (1.8-3.0), but it was considered likable (3.6-4.0) and intelligent (3.6-4.0). The active lucemes were not considered demanding in terms of physical effort (12.0 out of 100) or time (12.9), but somewhat mentally taxing (35.1). The ocular lucemes were rated at similar levels of physical effort (9.2) and time requirements (14.3), but lower in terms of mental demand (19.3).

3.6 Conclusion and Future Directions

In this chapter, we have presented the HREye, a novel, biomimetic, light-based AUV communication device. We designed and created a sixteen symbol active luceme language based on diver gestural languages as well as a set of ocular lucemes for communicating gaze direction, and evaluated both types of lucemes in a pool study with fourteen participants. This study demonstrated high recognition accuracy for active lucemes and intuitive understanding of directional lucemes as well as L_{Danger} and $L_{BatteryLevel}$. Additionally, the ocular lucemes were demonstrated to communicate gaze direction with an average error of 21°, a reasonable level of accuracy for the first attempt at gaze indication for AUVs. These results demonstrate that the HREye device and its active and ocular lucemes are suitable for use in human-robot collaboration, greatly expanding the variety of AUV-to-human communication methods with a novel, robust, and intuitive form of communication.

Possible Areas of Future Exploration

As with previous chapters, we now provide a few suggestions for future directions of research on this topic, based on our experience and the results of the studies presented within the chapter.

Passive Information Displays

The active and ocular lucemes of the HREye system provide useful communication primitives for divers but are directed and focused on small portions of time. The hardware could easily be used for passive data display, which may be useful for human-robot collaboration. We have developed some early versions of these passive data displays for utilitarian uses, prominently for displaying the results of object detection algorithms. However, the impacts of these passive data displays on collaborative work underwater have not yet been explored. This would also necessitate an in-depth design process of these displays, which differ from active or ocular lucemes in the type of information being displayed.

Further Investigation of HREye Gaze

Our initial study on ocular lucemes has demonstrated that divers can identify the gaze direction of the HREyes within an average 21° error. Future studies should focus on the ability of a robot to effectively indicate and differentiate between objects of interest using gaze. Additionally, it would be helpful to evaluate different eye constructions and animations to determine if greater success can be achieved. For instance, perhaps a different set of colors should be used, or the directional ocular lucemes should use a tighter angle of display. Further, no evaluation has been completed to explore the use of non-directional ocular lucemes such as blinking, squinting, and widening eyes on both gaze indication and affective display. While such animations may seem frivolous, an increased standard of realism and anthropomorphism may lead to improved results in human identification of robot gaze.

Non-Eye Structures

While the eye-like structure of the HREye devices has enabled a number of exciting new developments such as the creation of ocular lucemes, there is no evidence that these eye-like structures are the best way to express information. One of their major weaknesses is their limited viewing angles, making them nearly imperceptible from the side. It would be worthwhile for future explorations of emitted lights for UHRI to consider different morphologies, perhaps spread across the body of the AUV. These would lose the ability

to mimic gaze, but could potentially be used in concert with HREyes to provide a more universally observable light display.

Chapter 4

Sound For AUV-To-Human Communication

In previous chapters, we have described the current state of underwater robot-to-human communication (digital displays and dedicated devices) and introduced two new methods for robot-to-human communication using motion and emitted light. However, these new methods share a critical weakness: they all require visual observation, meaning that they cannot be used in low-visibility environments and depend on already having the diver's visual attention. An audible vector of communication would solve both issues, but aside from power/status-indicating tones, audio communication has not been significantly explored for underwater robots. Sound travels well through water, but producing and comprehending it is challenging [114]. Most commercially available speakers are not designed for vibrating water rather than air, while underwater-compatible speakers tend to be quite expensive, and incompatible with small AUVs. Additionally, human auditory processing is not well suited for comprehending sound underwater, leading to confusion and garbling of complex signals such as speech. Due to these confounding challenges, audible communication from robots to humans underwater is largely unexplored.

We present in this chapter a novel audio-based system for underwater robots, named **SIREN** (**S**ound **I**ndicators via **R**esonance **E**citors **u**Nderwater). In the following sections, we first discuss the background of this work, exploring the types of robot-to-human communication which have been developed thus far. Next, we present the hardware and

software used to create the audible communication indicators of SIREN, which we refer to as **sonemes**. We expand upon the design of these sonemes in the next section, defining two types of sonemes: synthetic speech (TTS-sonemes) and musical tone indicators (Tone-sonemes). To evaluate these sonemes for use in communication, we perform a substantial in-person human study with a total of 12 participants, in which swimmers identified sonemes at a variety of distances. While a study population of twelve is not considered large in many fields, this is the second largest UHRI study conducted in an underwater environment, due to the logistical challenges of performing such studies. The results from this study reveal the effectiveness of audible communication underwater at close range, with some viability at distance.

Defintion: Soneme

A soneme is a sound intentionally produced by a robot for interaction with a human. In the field of human-computer interaction, these sounds would be referred to as *earcons* [115]. However, to provide a distinction between the types of information communicated via sound by computers and robots, and to continue parity with our previous work on motion and light-based communications (*kinemes* and *lucemes*), we refer to robotic audio communication phrases as *sonemes*. The word soneme is derived from the Latin *sonus* (meaning sound) and the suffix *eme* used for phonemes and cheremes, fundamental parts of audible and gestural languages.

4.1 Background: Sound-Based Communication

SIREN is the first dedicated system for AUV-to-human communication. As such, the background of this chapter is split into two parts: previously developed approaches for AUV-to-human communication and previous sound-based robot communication methodologies.

4.1.1 Underwater Human-Robot Interaction

Underwater human-robot interaction is a relatively new field, with the first research actually focused on interaction published in the early 2000s. Nonetheless, the field has

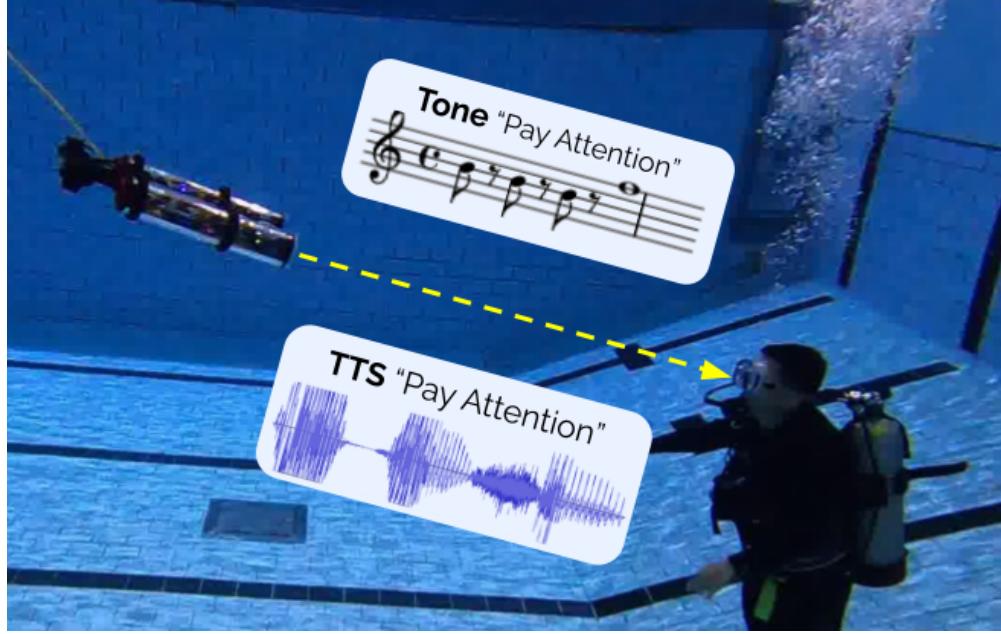


Figure 4.1: A depiction of the two types of sonemes that SIREN can produce, asking a diver for their attention.

already diversified significantly in these short twenty years. Human-to-robot communication in underwater environments is dominated by gestural control systems [2, 104], with lesser use of fiducial markers as cue cards [10] and self-contained remote control devices [17]. In the inverse problem of robot-to-human communication, the dominant method has long been displaying text on a screen [9, 15, 17]. The size of these screens varies, but their performance is relatively similar: complex and high-density information can be easily communicated, but viewing angles and distances are poor. In recent years this fact has led to the development of novel methods of AUV-to-human communication. We have previously proposed communication via motion [46, 105, 106], which is much more resistant to distance and orientation changes. Beyond motion, we also have expanded upon early, simple use of emitted light for communication [16] into a complex system of active communication and gaze direction indication [116]. Both of these systems have expanded the effective range at which AUVs can communicate with divers, but a problem still remains: what if no one is looking at the robot or the visibility is negligible? In those cases, any kind of communication based on the visibility of

the robot will fail. For this reason, we turn our attention to sound, which passes well through water, can be omnidirectional, and eliminates the need for good visibility.

4.1.2 Sound-Based Robot Communication

Sound is a common vector for robot-to-human communication, exploiting what Andrea Bonarini [117] calls the "Hearing Channel". The study of sound-based communication is broad, with a wide variety of applications of sound being investigated. A great deal of work has focused on synthetic speech for robots [118–120], particularly the ability of a robotic voice to convey emotion [121, 122]. Another aspect of robotic speech that has been well-studied is the effect of different types of voices on robotic acceptance and social behavior [123]. However, not all robotic sound is speech. A number of works have explored the effect that consequential sound – sound that the robot makes as a consequence of its operation – has on the perception of robotic operations and interactions [124, 125]. Further investigation has also considered the addition of artificial sound to consequential sound and the way that it affects perceptions of safety and capability [126]. Sound has also been used as a nonverbal signal to improve human perception of a robot's location [127]. While the use of nonverbal sound is less common in the field than speech, nonverbal sound has been applied to topics such as emotion and intention expression [128]. Similar works have ventured into generative methodologies, where properties of a robot's internal state or emotional cues are directly input into a sound synthesis engine to control different parameters [46, 129]. The design of these types of nonverbal sound communication provided design inspiration for the tonal sonemes we describe later in this chapter, though our indicators are pre-defined.

4.2 SIREN: A Low-Cost AUV Audio System

SIREN is the first dedicated system for sound-based AUV-to-human communication. Its purpose is straightforward: producing sonemes that divers can accurately perceive.

4.2.1 Hardware Design

Two pieces of hardware are required for SIREN: a transducer/exciter to vibrate against the frame of an AUV and an amplifier to drive the said transducer. The amplifier

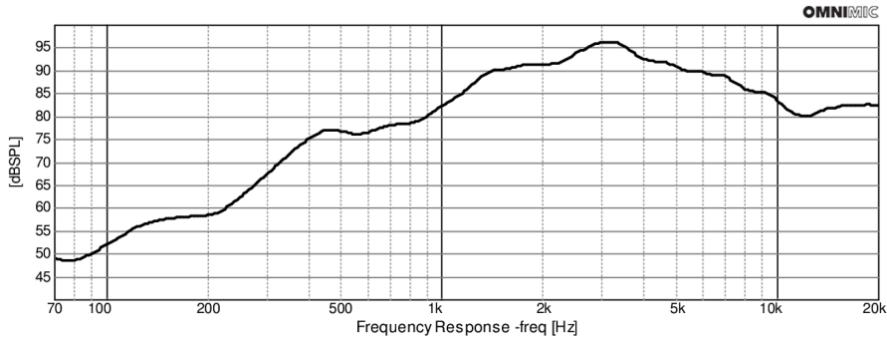


Figure 4.2: Frequency response of the Dayton Audio DAEX25W-8, from the provided specification sheet.

also requires a source of audio input, but as AUVs have onboard computers, this is not considered part of the required hardware. SIREN utilizes the DAEX25W-8 [130] waterproofed surface transducer produced by Dayton Audio, which is IP67 rated, having undergone one hour of immersion in one meter of water. Our transducer has been immersed to two meters for extended periods of time, but if a deeper depth rating is required, many other excitors can be acquired and affixed inside an AUV’s shell, circumventing the need for greater waterproofing. The exciter has a relatively strong frequency response at all frequencies, though it is most capable between 1k-10k Hz, as seen in Figure 4.2. However, the frequency response of our system is likely to be different, given that the material the exciter is attached to will have the greatest bearing on its ability to produce certain frequencies. We power our exciter with a small dual 10W amplifier from Parts Express, but any amplifier which fits in the available space and adheres to the power requirements of the exciter used would be appropriate. The cost for both parts totals 35 USD, making SIREN hardware an affordable addition to even the most economical of AUVs.

Why Not Earphones?

Commercial scuba divers often use open face helmets with built in microphones and earphones. These systems are extremely useful for surface communication and can be used for diver-to-diver communication – there is no reason why they cannot be applied to robot-to-diver communication. However, these systems are expensive, cumbersome,

and require a level of infrastructure that many divers do not have, even professional divers. In keeping with our general philosophy of avoiding the further instrumentation of the diver, we focus on a method of sound-based AUV-to-diver communication which require no additional equipment on the part of the diver.

4.2.2 Software Design: Reconfigurable Sound

The sonemes of the SIREN device are produced dynamically by a software module for PROTEUS, our UHRI software system for the Robot Operating System (ROS). PROTEUS loads XML definitions of sonemes and then builds ROS services to trigger them as needed. We wished to explore both synthetic speech and more abstract audio cues as a method of communication, so our software has two modes: **Tonal-Sonemes** and **TTS-Sonemes**.

Tonal-Sonemes

For tonal sonemes, PROTEUS expects an XML definition file that contains two things: a system configuration section defining the various waveforms to be used, and a set of soneme definitions. After parsing the definition file, the PROTEUS Tonal-Sonemes server uses a package called *tones* [131] to synthesize polyphonic music.

TTS-Sonemes

In the case of TTS-Sonemes, PROTEUS expects an XML definition file with a system configuration section specifying the voice, language, and volume to be used, as well as a set of soneme definitions. Once these definitions are parsed, a python package named *voxpopuli* [132] is used to interface with Espeak [133] and MBROLA [134]. Espeak is responsible for parsing the text into a list of phonemes, which MBROLA then synthesizes into audible speech. More high-quality text-to-speech software could be used, but we chose a minimal design that does not require an internet connection or GPU.

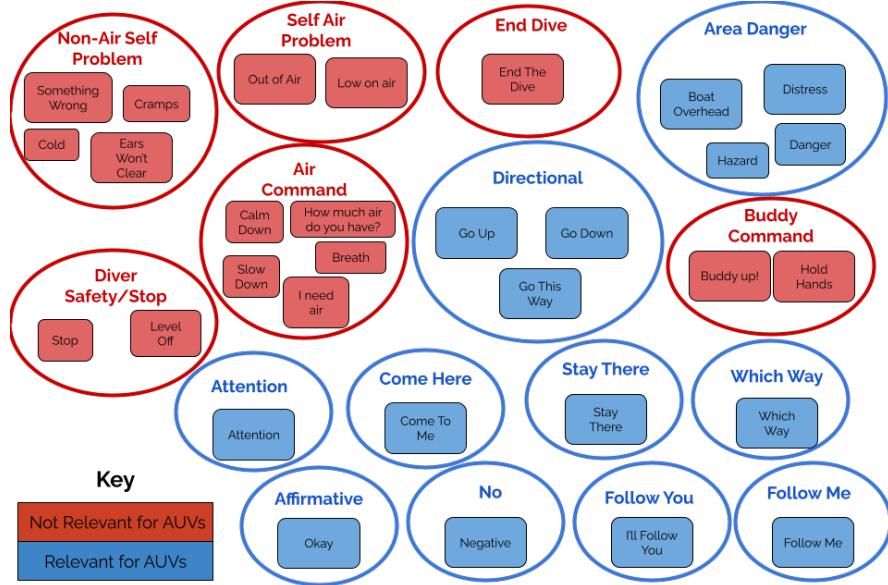


Figure 4.3: Clustering of scuba diver sign language symbols, sourced from instructional scuba materials.

4.3 Design of Sonemes

With the hardware and software for SIREN defined, we turn to specifying the sonemes of the system. Before defining sonemes however, we must answer a fundamental question: what might the robot need to say in an interaction?

4.3.1 Defining AUV Language Symbols

To create a language of robot communication phrases, we turned our attention to the sign languages in broad use among divers. There are many versions of diver sign language, and most divers have picked up further signs for specific situations they encounter in their dives. By considering instructional material for divers in training, we can find a common set of useful signs. The meanings of these signs are shown in Figure 4.3, along with our next step. After combining signs with the same meaning, we clustered these signs at two different levels. The first level, represented by the circles around signs in Figure 4.3, was common concepts, such as the cluster of signs which all refer to some kind of environmental danger. The second level, represented by the colors of circles and items, was the relevance of a concept to AUV communication. For instance, concepts

Soneme	Meaning	Tone Version	TTS Version
$S_{Affirmative}$	Yes, Okay.	Sqr.	“Yes.”
$S_{Negative}$	No.	Sqr.	“No.”
S_{Danger}	Danger in the area.	Sqr.	“Danger nearby!”
$S_{Attention}$	Pay attention to AUV.	Sqr.	“Pay attention!”
$S_{Malfunction}$	Internal malfunction.	Sqr.	“I’m experiencing an issue.”
$S_{WaitCMD}$	Waiting for instructions.	Sqr.	“I’m waiting for a command!”
S_{GoLeft}	Go left/AUV going left.	Tri.	“Go left.”
$S_{GoRight}$	Go right/AUV going right.	Tri.	“Go right.”
S_{GoUp}	Go up/AUV going up.	Tri.	“Go up.”
S_{GoDown}	Go down/AUV going down.	Tri.	“Go down.”
$S_{WhichWay}$	Asking for directions.	Tri.	“Which way are we going?”
S_{Stay}	Stay where you are.	Tri.	“Stay here.”
$S_{ComeHere}$	Come to the AUV.	Tri.	“Come to me.”
$S_{FollowMe}$	Diver can follow AUV.	Tri.	“Follow me.”
$S_{FollowYou}$	AUV will follow diver.	Tri.	“I’m going to follow you.”
$S_{BatteryLevel}$	Battery level is...	Sqr.	“N% battery remaining.”

Figure 4.4: Selected sonemes, with both Tone and TTS versions. See the accompanying video for recordings of each soneme.

relating to diver issues with air or bodily states such as trouble clearing ears are not relevant for an AUV to express. From this point, the selection of symbols was simple: any concept cluster relevant for an AUV to communicate was included in the language definition. Some symbols were added, such as $S_{WaitCMD}$ and others were adapted from clusters that were considered not relevant for AUVs, such as $S_{Malfunction}$ which maps to the Non-Air Self problem cluster.

4.3.2 Soneme Design

With the list of possible sonemes created in this manner, the next step was defining the audio for each soneme, which can be seen in Figure 4.4 and in the accompanying video.

Tonal Sonemes

For tonal sonemes, a variety of techniques were used. Firstly, any soneme with a negative implication (*e.g.*, S_{Danger} , $S_{Negative}$) was defined to use notes from a minor chord. Additionally, sonemes with positive meanings were designed to sound more cheery or energetic. An effort was made to begin most sonemes or groups of sonemes with unique notes, to reduce overlap between the initial notes of sonemes as much as possible. In addition, the sonemes which did have common start notes were designed to be as distinct as possible from one another other than their starting note, to further avoid confusion. Another design choice was the selection of waveforms used for various sonemes. Sonemes related to commands or information were rendered using a square wave-based tone generator, while directionally related sonemes used a triangle wave.

Text-To-Speech Sonemes

The design of text-to-speech sonemes was simpler, as the method was largely just writing out the meaning of the soneme. However, some phrases were intentionally lengthened to increase the amount of time that the sound would be audible, and others were modified to avoid confusion with other sonemes. These modifications focused on not starting TTS-Sonemes with similar words.

4.3.3 Version Selection Survey

With these design principles in mind, four versions of each soneme were created: two options for TTS-Sonemes and two options for Tonal-Sonemes. These versions were then demonstrated to a small internal focus group comprised of other AUV researchers and laypeople. While the various TTS phrases were all offered with the same voice, participants were also asked to select one of four voices available from MBROLA by listening to a sample sentence produced by each voice. Based on input from this survey, a final set of sonemes for each version of SIREN was selected by choosing the most popular option for each version, with some designer's discretion in the case of ties. The voice used for TTS-Sonemes was also selected using majority opinion.

4.4 Study V: How Do TTS & Tonal Sonemes Perform?

As the purpose of SIREN is to communicate with divers underwater, the only route to evaluating the effectiveness of the system is to conduct a human study with participants listening to SIREN underwater. Studies of this nature are challenging to administer, as finding and training participants can be time consuming, costly, and difficult due to low pool availability. The following sections describe the human study which we conducted to evaluate the effectiveness of our SIREN device and the two versions of the sonemes developed for it. This study was approved as human research by the University of Minnesota's Institutional Review Board (reference number: 00016705).

4.4.1 Study Design

The study of SIREN efficacy was fundamentally based on the success of trained participants at recognizing various sonemes underwater. After recruiting participants and training them to a pre-defined level of competence in recognizing lucemes, we asked them to identify those same sonemes in a pool environment. The initial distance of the AUV was one meter, but after all sonemes had been played, each participant was asked to identify sonemes at a distance of fifteen or twenty meters. Participants were randomly assigned to the TTS-Sonemes or Tonal-Sonemes conditions and were trained and tested only on that version using a between-subjects experimental design. Fourteen

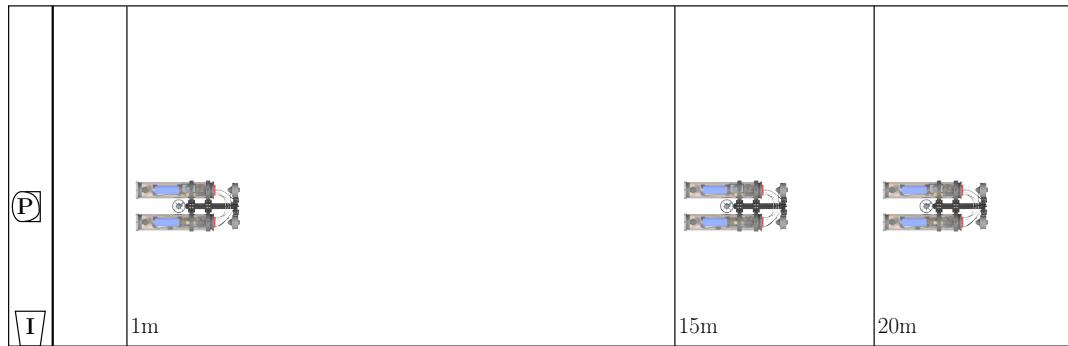


Figure 4.5: Experimental setup, featuring LoCO at the three possible distances, a participant (**P**), and the pool inlet (**I**).

people went through study procedures, but unfortunately, the data for two participants was contaminated due to technical issues, yielding twelve total participants' data being included in the analysis.

4.4.2 Study Procedures

The study is comprised of three steps: Participant recruitment and training, pool study sessions, and the debrief stage.

Participant Recruitment & Training

Participants were recruited from the University of Minnesota by emails, publicly posted fliers, in-classroom announcements, and word of mouth. Anyone who was interested was provided the link to a Qualtrics form, which began by collecting participant consent and demographic information. Participants who were over the age of 18, were not deaf or hard of hearing, able to swim independently, and had not taken part in previous studies from the IRV Lab were allowed to continue. In the second part of the survey, participants were shown videos with recordings of sonemes in their randomly selected SIREN version. They were taught 3 – 5 sonemes at a time, quizzed on their meanings, and then finally given a competency test after learning all sonemes. Only participants who correctly identified at least 12 of the 16 sonemes were able to continue to the end of the survey. At this point, participants were asked to complete an audiology test profiling their hearing ability, then schedule a time to complete their pool session.

Pool Study Sessions

Once participants had been trained in the use of SIREN, they scheduled time at a pool evaluation session. Three total sessions were administered to collect data from all 14 participants. Upon arriving at the pool, each participant was greeted and the pool session process was explained to them. Additionally, study staff read each participant the list of soneme meanings before beginning the session, as participants sometimes completed their education process as many as seven days before the pool sessions. The sounds for each soneme were not played to participants at this point. Once they had been prepared, participants entered the pool and were asked to identify sonemes played by the SIREN device attached to the LoCO AUV [29]. Participants submerged themselves, listened, then surfaced and reported the meaning of the soneme they had heard, along with their confidence in their answer on an ordinal scale from 0 to 10. Sonemes were demonstrated in a randomized order, first at a distance of one meter, then at a further distance of either fifteen or twenty meters. Other research activities were conducted at the same time, resulting in some ambient noise in the pool, which we believe improved the realism of the test. Additionally, a water inlet was located to the right of the participants, adding further background noise not dissimilar to the operation noise of scuba equipment.

Debrief Stage

After completing their pool session, each participant was asked to complete a small debrief survey, asking them their opinion on the SIREN system and the LoCO-AUV. This survey uses a modified Godspeed questionnaire [107] to measure attitudes about the AUV, and the NASA Task Load Index [109] survey to measure participant effort and stress during the soneme identification task. Participants were also allowed to input any comments on their sonemes they had into free-entry text boxes. Once a participant completed all of their study procedures, they were provided with a \$15 Amazon gift card. Additionally, once all participants had been enrolled, an additional \$50 gift card was given to a randomly selected participant.

4.4.3 Analysis of Data

Rating and Reliability

While performing pool sessions, participant answers on the meaning of sonemes and their confidence in said answers were recorded. Additionally the time from the beginning of a soneme to the beginning of a participant's answer was recorded. Two recordings were made, which were cross-checked with one another to ensure the accuracy of the data. Because participant identification of sonemes was given in their own words, further work was required to perform analysis on this data. Three independent raters from the study staff were asked to read through the recorded participant answers and rate the correctness of each answer from one to one hundred. To determine the level of agreement between raters, Fleiss's κ [111] was calculated to be $\kappa = 0.795$, which is typically understood to indicate a good level of agreement between raters. After determining this, raters' scores were averaged to create the final correctness score for each answer.

Metrics

When considering the effectiveness of SIREN, we utilize four metrics that have been beneficial in similar analyses in the past:

- **Accuracy:** The average of the rater's correctness scores, which indicates how accurately a participant has identified a soneme.
- **Confidence:** A value from 0 to 10, reported by participants, indicated their confidence in their answers.
- **Operational Accuracy:** The same values as accuracy, but only considering answers with a confidence ≥ 6 .
- **Time To Answer:** The time recorded from the beginning of a soneme to the beginning of a participant's answer.

Statistical Methods

The two metrics which will be analyzed for statistically significant effects are accuracy and operational accuracy. Prior to analyzing our data further, assumptions of statistical

tests must be considered. Most parametric statistical tests assume a normal distribution of data. Shapiro-Wilk [79] tests were performed for accuracy $W = 0.71, p < .001$ and operational accuracy $W = 0.51, p < .001$. Both results indicate that data is **not** normally distributed, and direct inspection of the data confirms this. Therefore, in all subsequent analyses, we use non-parametric tests, such as Spearman's correlation [135], Kruskal-Wallis H-tests [80], and Wilcoxon Rank Sum tests [75]. All tests are performed with a significance of $\alpha = 0.01$.

Removal of Two Participants

While fourteen people completed all study procedures, two participants had significant issues in their pool sessions. In one case, the wrong version of sonemes was played by accident, leading to significant confusion and removal from the final dataset. The other participant was shown the correct version of sonemes, but some data was lost due to a computer crash. Both participants were compensated equally to other participants, but their data is not included in our results.

4.5 Results of Study V

The following subsections describe the results of our human study in detail, with consideration given to population demographics, internal validity, the overall efficacy of the system, the difference between the TTS and Tonal versions, and the effect of distance on soneme communication.

4.5.1 Population

Our population is fairly small, with seven participants testing Tonal-Sonemes and five testing TTS-Sonemes. Approximately half the participants of each condition were tested at distances of 1m and 15m, with the rest tested at 1m and 20m. Eleven participants were between the ages of 18 and 24, with one participant between 35 and 44. Ten participants self-identified as male, with one identifying as female and the last identifying as non-binary/third gender. When asked if they had experience with robots, seven participants answered in the affirmative, and when asked if they had ever been scuba diving, four said they had. No participants self-identified as deaf or hard of hearing and audiology

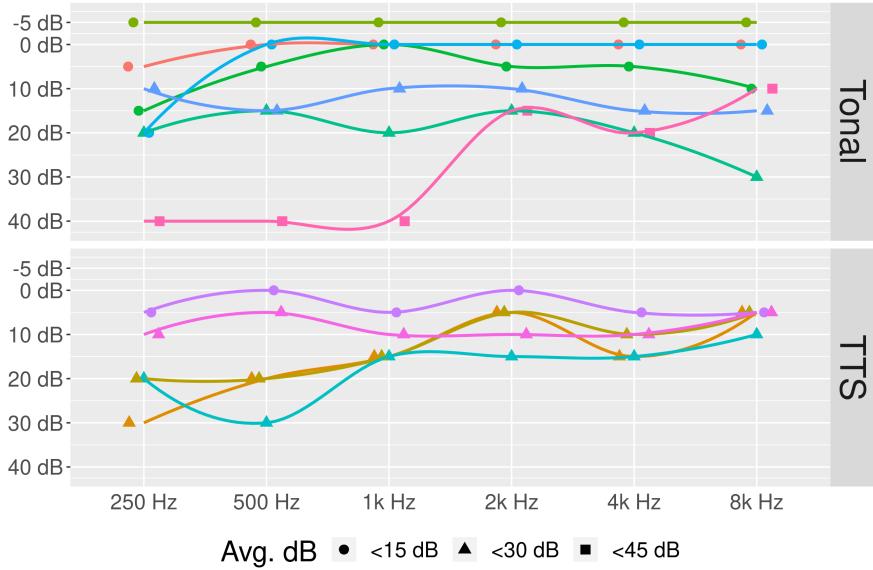


Figure 4.6: Audiometry data, expressed as the decibel level required by each participant to hear the given frequency.

data showed varying levels of hearing capability, with some falling into the level of mild hearing loss, as can be seen in Figure 4.6.

4.5.2 Internal Validity Tests

The primary situations which threaten the internal validity of this study are training-related. Does the duration of participant training, the level of competency after training, or the time between training and testing have a significant effect on accuracy? If so, a condition or distance's accuracy might be skewed, due to the training of participants in those sections. Because of the non-normal distribution of accuracy data, non-parametric tests are required, in this case, Spearman's rank correlation test [135] was used. No significant correlation is present between a participant's score on the training test and their accuracy, $r(10) = 0.25, p = 0.44$ (Figure 4.7a). Testing the correlation between the time taken during education and accuracy, no significant correlation was found, $r(10) = -0.15, p = 0.65$ (Figure 4.7b). Lastly, no significant correlation was found between the duration of a participant's training and pool session and the accuracy they achieved, $r(10) = -0.07, p = 0.82$ (Figure 4.7c). Finally, we consider the effect of

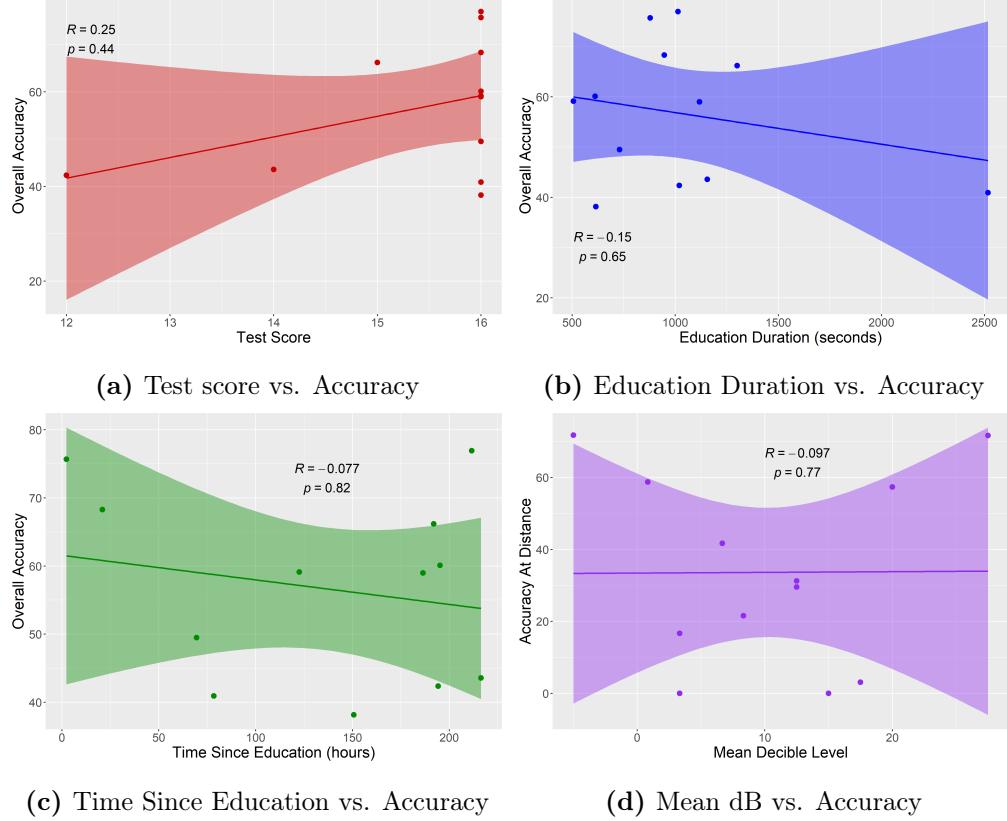


Figure 4.7: Internal validity tests for Study V.

a participant's hearing on their accuracy. We first average the decibel level required to hear all tested frequencies from the audiometry test, then assess correlation with participant accuracy (at distances greater than 1m) using Spearman's rank correlation. No significant correlation was found between a participant's hearing ability and accuracy, $r(10) = -0.09, p = 0.77$ (Figure 4.7d). With these tests completed, we can continue with our analysis assured that participant hearing capability and training procedures did not contaminate our results.

4.5.3 Overall SIREN Efficacy

In our pool study, SIREN was demonstrated to be an effective system for AUV-to-diver communication. The metrics previously discussed are presented in Figure 4.8, with separation between the TTS and Tonal versions, and every metric reported for

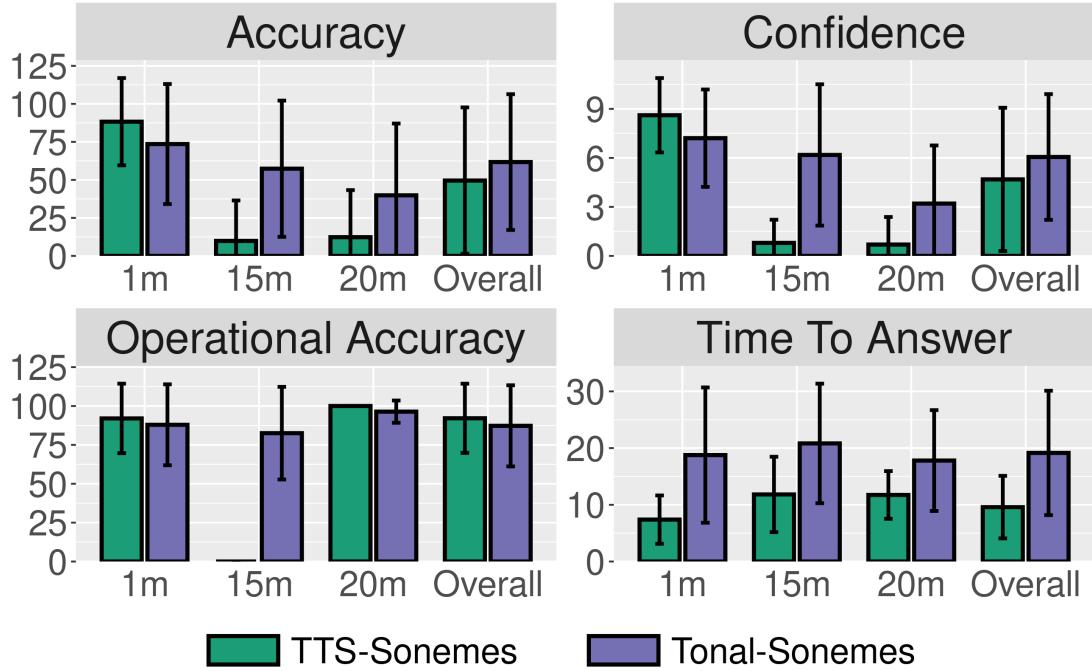


Figure 4.8: Metrics for TTS-Sonemes and Tonal-Sonemes.

the three test distances as well as overall. Both versions of sonemes achieved accuracy $\geq 50\%$ overall, with accuracy at 1m test distances $\geq 70\%$. While there is no accepted standard for considering an AUV-to-human communication system “field-viable”, these numbers, combined with the operational accuracy of both types of sonemes being $\geq 80\%$ overall, form a strong case that the system is effective at communicating. Additionally, the average time to answer values of 5-12 seconds indicates that the time required to understand the system is not out of line with other systems previously evaluated. In addition, we can see in Figure 4.9 that there are some outlier sonemes that are less frequently understood than others, particularly in the tonal condition. This indicates that a redesign of some sonemes could lead to greater accuracy if the aspects of the sonemes that are difficult to recognize or remember can be identified.

4.5.4 Should We Use TTS-Sonemes or Tonal-Sonemes?

A Kruskal-Wallis test [80] showed that the type of soneme (Tonal or TTS) had no effect on overall accuracy, $H(1) = 5.30, p = 0.02$. However, when we consider the

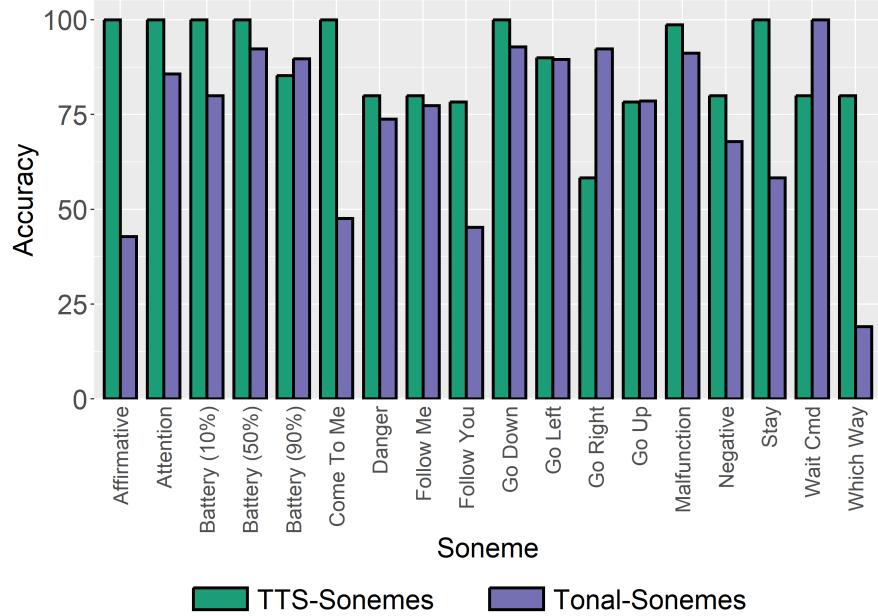


Figure 4.9: A comparison of per-soneme accuracy for SIREN at 1m.

accuracy of soneme identification at specific distances, we find a significant effect at 1m, $H(1) = 11.57, p < 0.01$, favoring the use of TTS-Sonemes. A Kruskal-Wallis test performed on accuracy at 15m and 20m combined shows a significant effect, $H(1) = 41.55, p < 0.01$, with Tonal-Sonemes leading in accuracy. These results explain the initial test: while there is no clear better version overall, at different distances of interaction, Tonal-Sonemes or TTS-Sonemes are significantly more effective.

Participant Effort and Stress

In the debrief stage, participants were asked to complete a NASA Task Load Index survey [109], which measures the mental, physical, and time demands of a task, as well as a person's estimation of their performance, the level of effort required, and the level of frustration induced by the task. Kruskal Wallis tests found no statistically significant differences between the answers of participants in the Tonal-Sonemes condition and participants in the TTS-Sonemes condition. In particular, no effect was found on the participant's reported effort, $H(1) = 0.077, p = 0.781$, or on their reported frustration, $H(1) = 0.697, p = 0.404$, two areas of major concern. A Spearman's rank correlation

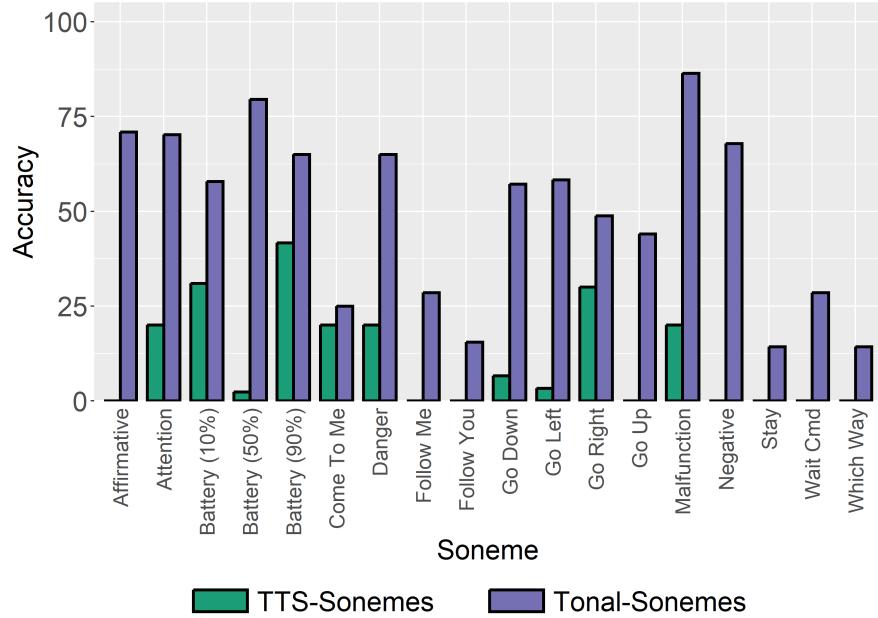


Figure 4.10: A comparison of per-soneme accuracy for SIREN at 15m and 20m.

test also found no significant correlation between a participant’s estimation of their performance and actual accuracy, $r(10) = 0.21, p = 0.52$.

4.5.5 How Does Distance Affect SIREN?

A Kruskal-Wallis test showed that the distance of interaction had a significant effect on accuracy, $H(2) = 107.75, p < 0.01$. Furthermore, performing pairwise comparisons using Wilcoxon rank sum testing [75] with Holm-Bonferroni p-value adjustment [136] reveals that there exist significant effects between the 1m distance and the others distances, but no significant effects are present between 15m and 20m. This much is readily apparent from an inspection of the accuracy values in Figure 4.8. By creating a categorical variable combining soneme type and distance (*i.e.*, tts_1m, tone_15m), we can consider the interactions between these variables. A Kruskal-Wallis test shows a statistically significant effect on accuracy from this condition-distance variable, $H(5) = 151.03, p < 0.01$. To further understand this effect, we perform a pairwise analysis using Wilcoxon rank sum tests with Holm-Bonferroni p-value adjustment, the results of which can be seen in Table 4.1. The pairwise results reinforce the finding of the tests on specific

	Tonal-1m	Tonal-15m	Tonal-20m	TTS-1m	TTS-15m
Tonal-15m	0.033	-	-	-	-
Tonal-20m	> 0.001	0.0535	-	-	-
TTS-1m	0.0034	> 0.001	> 0.001	-	-
TTS-15m	> 0.001	> 0.001	0.0019	> 0.001	-
TTS-20m	> 0.001	> 0.001	0.0179	> 0.001	0.7838

Table 4.1: The effect of combined condition/distance on soneme accuracy, calculated using pairwise Wilcoxon rank sum tests, adjusted using the Holm-Bonferroni method.

distance results in the previous section. Specifically, TTS-Sonemes outperform Tonal-Sonemes at close distances but fail at further distances and the further away from the participant SIREN is, the harder it is for a participant to recognize sonemes. Lastly, the recognition Tonal-Sonemes is not as negatively affected by distances as the recognition of TTS-Sonemes.

4.5.6 Why Are Some Sonemes Harder To Hear?

These results on the performance of both versions of SIREN at a distance, along with the per-soneme results shown in Figures 4.9 & 4.10, raise an interesting question: Why are some sonemes harder to comprehend than others, particularly at a distance? One contributing factor to difficulties identifying sonemes at a distance may be their duration. A Kruskal-Wallis test performed on accuracy at all distances found no significant effect from the length of a soneme, $H(28) = 42.209, p = 0.0414$. However, when considering accuracy at distances greater than 1m, the same test finds a statistically significant effect $H(28) = 78.726, p < 0.01$. Spearman’s rank correlation agrees with this finding, showing a statistically significant, positive correlation between the soneme length and average soneme accuracy (considering distances $> 1\text{m}$), $r(34) = 0.662, p < 0.01$. This indicates that the longer a soneme is, the easier it is to understand from a distance. This likely does not fully capture the complexity of soneme design and recognizability, which likely has to do with the frequencies used, their relation to background noise, and the frequency response of the audio production device.

4.5.7 Godspeed Questionnaire Results

In the debrief survey, participants were asked to complete a modified version of the Godspeed [107] questionnaire. Participants in their responses indicated positive feelings toward the robot, rating it pleasant ($\mu = 3.83$) and friendly ($\mu = 3.92$). They also considered it to be interactive ($\mu = 3.17$), but mechanical ($\mu = 1.50$) and machine-like ($\mu = 1.58$).

4.5.8 Participant Impressions Of SIREN

At the end of the debrief survey, participants were allowed to write any comments they wanted, some of which resonated with statistically measured quantities and our own impressions of the system. *Participant A* remarked “The higher/more aggressive tones felt easier to hear from a distance than the lower/softer tones...The higher tones seemed to cut through the water white noise much better for me”. This reflects the difficulties that participants had identifying sonemes at a distance, but also the fact that some Tonal-Sonemes performed better than others overall. *Participant B* also backed this up, saying “The 15 meter test was difficult/frustrating. I could only hear a faint noise and I couldn’t make it out.” *Participant C* said “Very short commands were harder to parse.” This reflects the finding that longer sonemes are more easily identified and the fact that *Participant C* was in the TTS-Soneme condition.

4.6 Conclusion and Future Directions

In this chapter, we presented SIREN, a device, software system, and two soneme languages for audible communication from AUVs to divers underwater. We first presented the hardware and software design of our system, along with two versions of a sound-based communication language. With our sonemes defined, we performed a human study of soneme perception in underwater environments with 12 participants. The results from this study revealed reasonable accuracies for both forms of sonemes at close distances, with tonal sonemes operating more effectively than text-to-speech sonemes at greater distances. Our analysis of these results also revealed correlations between soneme length and performance at distance, and indicated some possible directions for

further improvement of sonemes. SIREN is the first system of its kind for AUVs, providing a low-cost device for AUV-to-diver communication. The results of our work have established a baseline for its performance and some strengths and weaknesses of certain types of auditory communication underwater.

Possible Areas for Future Exploration

As with previous chapters, we now provide a small set of possible improvements for SIREN, based on the results of our investigation into audible communication for underwater robots.

Matching Scuba Breathing Rate

An issue that many of our participants mentioned in this study and later, in Study VII (Chapter 10) was being unable to hear sonemes due to outside sound. In Study VII, participants had particular difficulty hearing over the sound of their scuba apparatus. While this did not prevent them from understanding sonemes, it made the process more difficult. If sonemes could be synchronized with the breathing rate of a target diver, possibly by identifying breath patterns via observation of bubbles, SIREN could potentially achieve higher efficacy.

Determining Ideal Soneme Frequency Range/Length

With a correlation established between soneme length and recognition rate at distance, the question of how long a soneme should be arises. This also has an interaction with the previous future direction: could sonemes overcome the issue of outside sound interrupting them by increasing their length? Establishing the ideal length of a soneme would be a difficult process, but likely helpful in increasing recognition rates. Similarly, the question of what frequencies are most easily perceived could be explored, developing a set of guidelines on soneme design predicated on improving perception and recognition of sonemes by using certain frequency ranges and soneme lengths.

Increasing Intuitiveness of Tonal-Sonemes

While not directly evaluated in our study, it is readily apparent that Tonal-Sonemes are less intuitive to understand than TTS-Sonemes, as they do not take advantage of an established language. This question of how to design intuitive sound dovetails with similar questions surrounding design of kinemes and lucemes, and has no less potential depth. The design of sound encompasses not only pitch and volume, but timbre, prosody, and dynamics. A great deal of existing HRI research on synthetic speech could be applied to TTS-Sonemes, but the design of communicative tones must draw much more inspiration from the design of earcons and other sound effects in human-computer interaction.

Part II: AUV Perception of Humans For Interaction

Perception of humans is just as critical to successful collaboration between AUVs and divers as interaction capabilities. Without the ability to effectively perceive divers, an AUV would be severely limited in its usefulness, unable to effectively maneuver around divers, understand the diver's activities, or receive input. AUVs use various exteroceptive sensors, but co-AUVs are commonly equipped with either single or multi-camera vision systems. This reflects the fact that co-AUVs are meant to be deployed alongside divers, in environments that lend themselves to visual observation and communication. For this reason, we present a corpus of work on the visual perception of divers in the following chapters. Chapter 5 introduces an enormous dataset that enables the detection of divers in video streams using deep neural networks and contains an exploration of the temporal stability of such methods. This work provided a notable improvement in the state of the art for diver detection. Next in Chapter 6, we explore the question of predicting the future motion of a diver, a first for underwater contexts. Chapter 7 builds on the ability of diver detection and human body pose estimation, introducing a novel method for estimating the relative distance to a diver using only a single camera view. This additional information enables the development of a new capability for AUVs: diver approach. Lastly, in Chapter 8 we discuss the adaption of existing body pose estimation methods to diver contexts to use as the input for gestural control of AUVs. We develop a one-shot learning method using these body pose trajectories, which is currently insufficiently robust for field deployment, but an important first step in improving the reconfigurability of AUV gestural control.

Chapter 5

Evaluation and Improvement of Diver Detectors

Many of the potential applications of co-AUVs involve close proximity with human diver partners. A key perception capability for these situations is diver detection: an AUV must have an understanding of where humans are in order to safely move in the environment, communicate with its operator, or follow a human to a location. Diver detection research began with frequency domain analysis methods [4] but has more recently been dominated by methods based on convolutional neural networks. While previous diver detectors have achieved success in terms of traditional object detection metrics (precision, recall, and intersection over union), neural network-based methods have had difficulty with producing stable and consistent diver detections in a temporal sense and struggle with non-following contexts. In particular, previous work we completed on diver detection using the Deep Diver Dataset (DDD) achieved high accuracy results on individual images but had significant temporal disruptions in videos (false negatives for some images, changing scale and position). Similar methods also struggle with detecting divers who are depicted in positions different than the expected position for diver following contexts. To address these shortcomings, we introduce VDD- \bar{C} , a new dataset for training and evaluating diver detectors on video sequences, then train and evaluate a set of object detection methods (including one video object detector) on it.



Figure 5.1: Divers operating the Aqua AUV in the ocean. A visual diver detection algorithm is responsible for determining the position of each diver within the image.

5.1 Background: Diver Detection In Video Contexts

Diver detection has long been a topic of interest in AUV, with techniques applied such as sonar signal processing, traditional computer vision methods, and the use of object detectors based on deep neural networks. This section provides a brief overview of object detection and its weaknesses in the context of videos, the evaluation of object detectors across time, and the field of diver detection as a whole.

5.1.1 Object Detection

Object detection is a computer vision task that involves identifying and localizing objects. Convolutional neural networks (CNNs) have typically been the highest performing object detection models [137, 138] and can generally be divided into two groups: two stage region-based detectors, which propose object regions in stage one and extract features from these regions in stage two (*e.g.*, Region CNN [139] and its descendants Fast R-CNN [140], Faster R-CNN [141], Region FCN [142], Mask R-CNN [143]), and one stage grid-based detectors, which skip the region proposal step and instead extract features over a dense, static grid of possible object locations in the image (*e.g.*, SSD [144],

YOLO [145]). One stage detectors are less accurate but fast enough for realtime inference, while two stage detectors are more accurate but often insufficiently fast.

State-of-the-art CNNs perform impressively well on vision benchmarks. However, these CNNs can stumble on images that come from a video stream [146–149], which is particularly problematic for robotic applications [150]. Research investigating this phenomenon points to multiple reasons behind this performance deficit. One issue is that as objects move in a video, they appear at a variety of locations. Image translations as small as one pixel can result in a radically different image representation at the deepest layers of state-of-the-art CNNs [146], which means that CNNs can struggle to generalize to the wide range of translations seen in video data. It is also important to note that CNNs are often trained on datasets like ImageNet [151] that have demonstrable location bias: the photographed objects’ locations are not equally distributed throughout the dataset, and traditional data augmentation strategies do not sufficiently address this problem [146, 152]. Similarly, objects in videos appear in a variety of orientations, which is also challenging for CNNs. Datasets typically present relatively head-on views of objects, whereas videos typically capture objects from a wide range of vantage points [153]. There is significant evidence that state-of-the art object detectors generalize very poorly to certain rotations [152, 153].

5.1.2 Video Object Detection and Temporal Stability

Learning to detect objects well in video is motivated by many applications, from robotics to surveillance. Since the release of ImageNet VID [154] in 2015, researchers have developed many models for video object detection. These detectors can learn to exploit temporal information in video streams in order to make better detections on video data, and they typically outperform static image detectors on video datasets [155]. Video detectors utilize a variety of strategies for leveraging temporal information, most notably linking static detections across frames in tracklets/tubelets [156], optical flow [157], and spatio-temporal feature memory [158–160]. However, many video detectors cannot perform inference in real time, making them unsuitable for robotic applications. Most realtime-capable video detectors (*e.g.*, [155, 159, 161]) achieve faster speeds by focusing intensive computational efforts on periodic “key frames” and propagating some of these computed features to subsequent frames, rather than computing features for every frame.

When it comes to the evaluation of video detectors, most works focus primarily on typical object detection metrics such as mean average precision (mAP). Notably, these metrics do not take into account the temporal nature of video data. Recently some metrics [162, 163] have been proposed that evaluate video detectors not only on mAP, but also on the stability of bounding box location and scale for a given object across frames (*i.e.*, how much the bounding box jitters around the ground truth) and on how fragmented detections are for each object in the video (*i.e.*, during the duration of an object’s presence in the video, how many times does an object’s status change from “detected” to “undetected”), although these metrics have not yet been widely adopted.

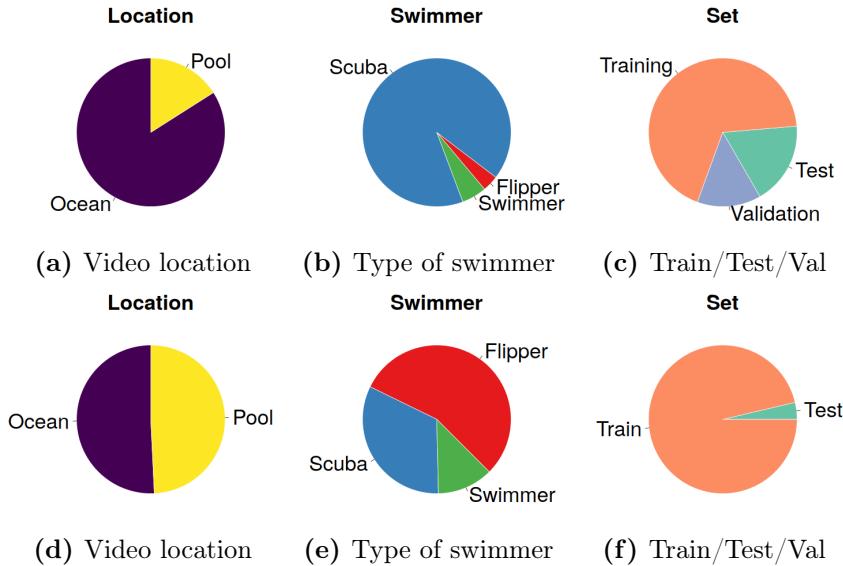


Figure 5.2: Distribution of VDD- \bar{C} (a-c) and DDD (d-f) data.

5.1.3 Diver Detection

Research on diver detection can be viewed as an offshoot of the general task of person detection [164, 165], but has significant challenges not present in typical person detection scenarios. A significant number of works have considered the problem of detecting divers in sonar data [166, 167] and by using hydrophones [168–171]. However, as many co-AUVs use vision systems as their primary exteroceptive sensors, visual diver detection methods are also quite common. Early visual diver detection methods depended on traditional

computer vision techniques [172–174] such as visual filtering, image differencing, and template matching. Other methods explored the detection of divers by attempting to extract frequency domain information to identify the periodic motion of a diver’s flippers [6, 175, 176]. In more recent years, the primary focus has been on the application of deep neural networks [13, 177] and other modern machine learning techniques [178] to diver detection, a trend which has also extended into sonar detection [14, 179]. Additionally, methods which not only detect divers but uniquely identify them have been developed [180, 181], though their effectiveness remains in question. The use of deep learning techniques such as CNN’s has led to diver detectors suffering from the same weaknesses as typical object detection algorithms in terms of the quality and stability of detections across time. This is problematic, as any autonomous behaviors of an AUV depending on diver detection will be degraded in their effectiveness by significant temporal instability. For instance, diver following is less effective, because the AUV’s belief state about the position of the diver can change significantly from frame to frame, leading to inappropriate control inputs. To some extent, these effects can be reduced by applying filtering techniques. However, improving the stability of deep neural network-based diver detectors is likely to have a greater effect, as improving the quality of a filter input will naturally improve the overall quality of filtered data.

5.2 VDDC: An Order of Magnitude

Upon beginning our investigation of the temporal stability of diver detectors, we quickly found that detectors trained on the publicly available Deep Diver Dataset (DDD) suffered from significant issues in video contexts. While detectors trained on DDD performed well in terms of traditional accuracy metrics and in pool environments, their performance was degraded on videos from field environments, and detections were not consistent across time. An investigation of the contents of DDD revealed the following issues:

- (i) DDD is a relatively small dataset from a deep learning perspective, with 6,011 images in its training set.
- (ii) While many of the images are from videos, the organization of the dataset does not lend itself to temporal stability testing or training video detection methods.

- (iii) The majority of the training images are biased to the application of diver following: a single diver swimming away from the camera is a common image.

To address these shortcomings, we present a new dataset, the Video Diver Detection dataset ($\text{VDD}-\bar{C}$), available at <http://irvlab.cs.umn.edu/vddc>.

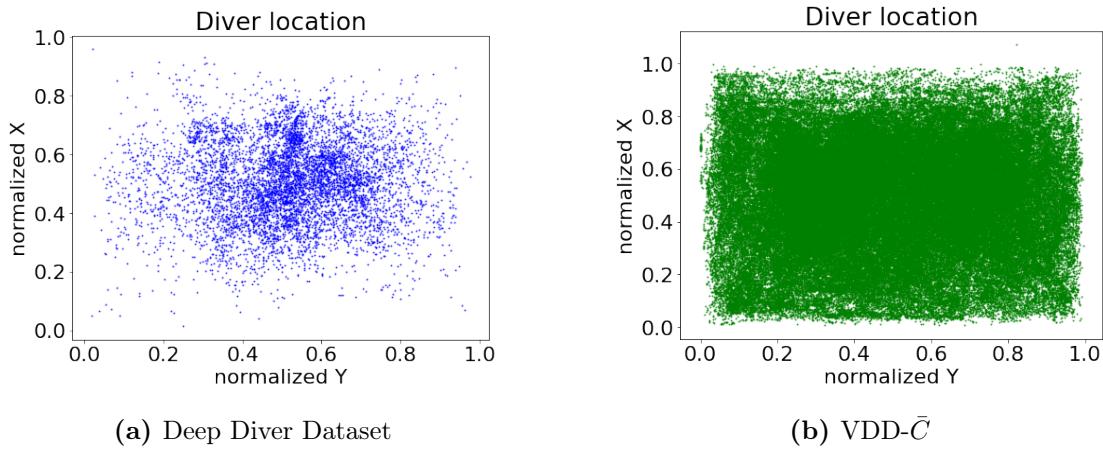


Figure 5.3: Distribution of bounding box centers.

5.2.1 Source Data

With the goal of temporal stability testing and video detection in mind, we chose to create our dataset out of videos, extracted into images at a rate of 20 frames per second for annotation. The majority of the videos were from dives off the coast of Barbados in the Caribbean Sea, but a number of videos were taken in pool environments. The percentage of the dataset containing images from different environments (ocean/pool), images featuring different diver gear types (scuba/flippers/no gear), and images allocated to training, test, or validation sets is visualized in Figure 5.2 for $\text{VDD}-\bar{C}$ (5.2a - 5.2c) and DDD (5.2d - 5.2f). Note that the figure shows percentages rather than number of frames. For instance, while a smaller percentage of $\text{VDD}-\bar{C}$'s total data is from pool environments, it has nearly three times as many images of pool environments (16,657) as DDD has images of any type. Additionally, while there is less variety in what equipment divers are wearing, a much wider array of viewpoints are represented: divers were recorded swimming with or without a robot, viewed from many different angles, and

sometimes merely treading water. Analysis of bounding box centroid location (Figure 5.3) clearly shows the greater variety in diver locations present in our new dataset. In comparison to previous datasets, our VDD- \bar{C} dataset is more numerous, contains a sufficient amount of diver and environment variation, and has a much wider range of viewpoints and diver activities represented.

5.2.2 Labeling Process

Once the videos for the dataset had been selected and extracted to frames, the task of labeling them was addressed. Labeling 105,000 images one by one was a time-consuming task, but it was improved by our choice of labeling tool. We used EVA [182], a web-based tool for labeling video data. Annotation is completed normally for the first frame in a video, with a user drawing a bounding box around every object they wish to label. Then, the user clicks the track button, and the initial annotations are used to initialize a kernelized correlation filter tracker [183] which propagates those bounding boxes over the following frames. Depending on the difficulty of the tracking, the generated bounding boxes need to be adjusted and re-tracked somewhere between every frame and every 30 frames.

5.2.3 Post-Processing

With the initial labels generated, we began post-processing our data, beginning with a significant proofreading effort. To proofread, we watched every labeled video from start to finish to look for labeling errors, and we then corrected all observed labeling errors. Following the correction of these errors, a number of sections of video were cut due to significant motion blur or degradation of the visual quality. Additionally, any frame in a pool video which contained no diver was cut, as these frames were almost entirely from portions of the video with the camera out of water. A significant number of images were cut, bringing the total number of images down from approximately 114,000 to the 105,000 images previously mentioned. Finally, the exported annotations were automatically filtered for bounding box coordinates out of the acceptable range of the image size before being converted to YOLO [184]-style labels, TFExample [185] and TFSequence [186] records.

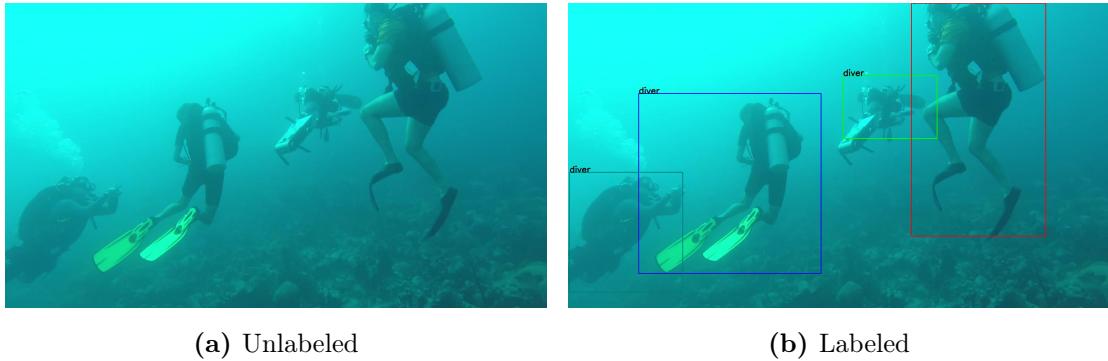


Figure 5.4: An image from VDD- \bar{C} , with and without labels.

5.3 Neural Networks For Diver Detection

We trained four models (some with a few variants) on VDD- \bar{C} with the aim of answering the following questions:

1. Do models trained on VDD- \bar{C} generally perform better on both the VDD- \bar{C} and DDD test sets than models trained on the DDD dataset?
2. Do models with the highest accuracy (as measured by mAP) also have the highest temporal stability?
3. Which models can perform inference fast enough to be usable on mobile AUVs?

In the following sections we briefly explain these models, along with the training process we used for each, and any variants for which we report results.

5.3.1 Faster R-CNN

Faster R-CNN [187] is a two-stage object detector in the R-CNN family and a staple high accuracy object detector. Although it is not fast enough for use on a robotic platform, we chose to train this model to loosely represent a “top end” accuracy of state-of-the-art CNNs on our dataset. We utilize the Tensorflow Object Detection API [188] and its Faster R-CNN with an Inception-ResNet-v2 [189] feature extractor for training. Two hyperparameters, learning rate and batch size, were tuned with the validation set.

5.3.2 SSD with Mobilenet

SSDs [190] are among the most accurate real-time object detectors and therefore are good candidates for eventual deployment on a robotic platform. We train SSD320 (*i.e.*, SSDs for inputs sized 320×320) models with multiple Mobilenet [191] backbones to find the optimal model for our use case. We utilize the Tensorflow Object Detection API and the provided models for training. Learning rate and batch size were again tuned with the validation set.

5.3.3 YOLO

You Only Look Once (YOLO) [192] is a well established object detection model, valued for its high accuracy and speed. YOLO predicts a set of bounding boxes in a grid across the image, with confidences for each, and class probabilities for each grid box, matching class probabilities to the boxes with the highest confidences. We evaluate a variety of versions of YOLO (v2 [184] and v4 [193]) in this work, although YOLOv4 is our primary version for comparing with other networks. For every version of YOLO we train, we also train Tiny-YOLO, which reduces the number of convolutional layers and filters to improve the inference runtime of the network. To train these networks, we fine-tune them using initial weights trained on Imagenet.

5.3.4 LSTM-SSD

The only video object detection network evaluated in this work is the LSTM-SSD detector proposed by Liu and Zhu [194]. This model is based on Mobilenet SSDs, but adds a number of Bottleneck LSTMs [194] after the feature extraction network, followed by output layers. On the next frame’s inference, features extracted by the convolutional layers will be combined with the LSTM’s state, propagating feature maps through time. In order to train the LSTM-SSD, we initialize the convolutional portion of the network from a fine-tuned MobileNetV1 SSD, then train the LSTM portion of the network in order to improve the feature propagation.

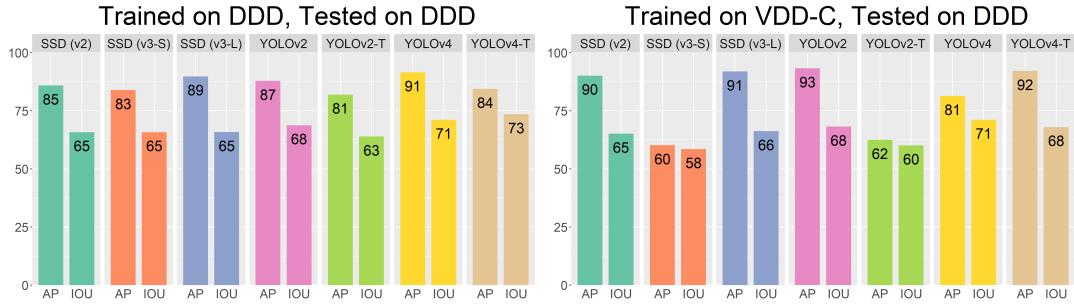


Figure 5.5: Models trained on both datasets, evaluated on DDD in terms of Average Precision at 50% IOU and IOU itself.

5.4 Traditional Accuracy Evaluations

5.4.1 Dataset Comparisons

To quantify how effective the new $VDD-\bar{C}$ dataset is in training deep vision models compared to previous datasets, we train one version of each SSD and YOLO variant on the $VDD-\bar{C}$ dataset and a second version on the existing DDD dataset. (We do not train Faster R-CNN or LSTM-SSD on DDD, since these models are not likely to be deployed and training is time- and resource-intensive.) We then compare the models’ respective performances on each test set as shown in Table 5.1. Results show that models trained on $VDD-\bar{C}$ outperform those trained on DDD on both the $VDD-\bar{C}$ test set and, to a lesser extent, the DDD test set. These results support our expectations that $VDD-\bar{C}$ ’s more complex data will lead to more successful detectors, because the $VDD-\bar{C}$ trained models outperform the DDD-trained models with few exceptions. Additionally, the fact that our $VDD-\bar{C}$ dataset is more challenging is reflected in these results, as DDD-trained detectors perform more poorly on the $VDD-\bar{C}$ test set than they do on the DDD test set.

5.4.2 Average Precision and IOU

To evaluate the accuracy of each model, we calculate the average precision (AP) of each model on diver identification. The average precision is found by evaluating the model’s precision at different recall values. Specifically, since models output a confidence score for each detection, model recall values can be manipulated by changing the confidence

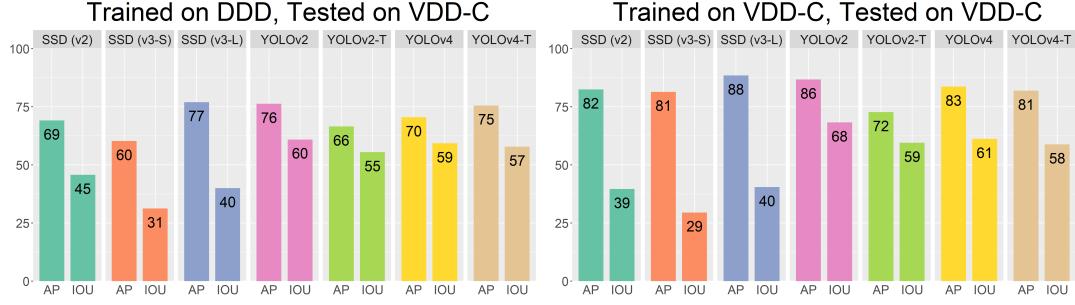


Figure 5.6: Models trained on both datasets, evaluated on $\text{VDD}-\bar{C}$ in terms of Average Precision at 50% IOU and IOU itself. Both versions perform well.

Model	Trained on $\text{VDD}-\bar{C}$						Trained on DDD					
	Tested on $\text{VDD}-\bar{C}$			Tested on DDD			Tested on $\text{VDD}-\bar{C}$			Tested on DDD		
	AP ₅₀	AP ₇₅	IOU	AP ₅₀	AP ₇₅	IOU	AP ₅₀	AP ₇₅	IOU	AP ₅₀	AP ₇₅	IOU
SSD(MnV2)	82.43	41.07	39.61	90.12	33.09	65.14	69.14	25.61	45.73	85.90	19.8	65.80
SSD(MnV3-Small)	81.43	34.03	29.50	60.20	11.10	58.50	60.24	11.10	31.23	83.90	17.46	65.80
SSD(MnV3-Large)	88.47	44.16	40.42	91.29	27.94	66.21	77.01	26.19	40.01	89.81	30.20	65.85
YOLOv2	86.73	38.04	68.25	93.30	23.89	68.19	76.27	19.3	60.92	87.84	21.93	68.76
YOLOv2-Tiny	72.69	8.82	59.48	63.50	3.14	60.12	66.51	6.17	55.47	81.95	3.27	63.95
YOLOv4	83.65	34.13	64.16	81.38	21.64	71.13	70.47	20.86	59.25	91.57	18.23	71.13
YOLOv4-Tiny	81.93	34.7	58.82	92.15	26.26	68.00	75.56	19.43	57.83	84.41	11.44	73.56

Table 5.1: Comparison between performance on test sets of the $\text{VDD}-\bar{C}$ and DDD with training on either train set. This same data can be viewed in Figures 5.5 & 5.6.

threshold required for a detection; the AP is the weighted mean of the model’s precision values at each recall value, where the weight for the recall at a given confidence threshold is the increase in recall from the previous threshold. Note that since our models are only trained to identify divers, the diver AP is equivalent to the mean average precision (mAP), which is a widely used object detection metric [195]. For each model, we pick a confidence threshold that results in the best precision and recall scores. Using that confidence threshold, we calculate AP at IOU thresholds of 0.5 and 0.75, average IOU, and an average of APs with thresholds between 0.5 and 0.95 with a step size of 0.05 (0.5-0.95). These values are shown in Table 5.2.

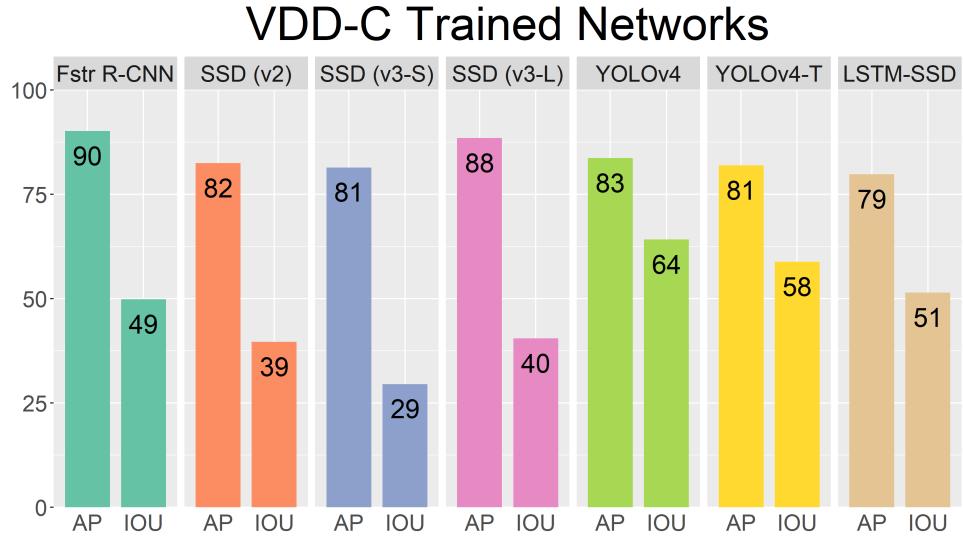


Figure 5.7: The overall average precision at 50 and IOU results for VDD- \bar{C} trained networks on the VDD- \bar{C} test set.

5.4.3 Efficiency Results

YOLO, SSD, and LSTM-SSD are high speed models designed for real-time inference use cases. While Faster R-CNN has demonstrated high accuracies, it is computationally involved, and is not quite suitable for real-time inference as shown in [13]. In order to quantify the usability of our real-time models on robotic platforms, we quantify their inference run-time in terms of frames processed per second (FPS). The results of these tests can be seen Table 5.3. Due to the size of the test dataset, we only tested a portion of the test set for runtime calculation: 5,000 randomly selected frames. We tested each network on two devices: an Nvidia 1080 GPU and an Nvidia Jetson TX2. These results do not represent the maximum inference speed possible, as no platform-specific optimization was done, but they provide a guide to the applicability of these networks in embedded contexts, on board AUVs. While the SSD variants achieve relatively high framerates on embedded devices, the clear standout is YOLOv4-Tiny, which achieves real-time performance with accuracy closer to other methods. LSTM-SSD also performs quite well, surpassing the traditional SSD variants.

Model	AP	AP ₅₀	AP ₇₅	IOU
Faster R-CNN	55.50	90.18	60.50	49.81
SSD(MobileNetv2)	43.45	82.43	41.07	39.61
SSD(MobileNetv3-Small)	39.81	81.43	34.03	29.50
SSD(MobileNetv3-Large)	47.05	88.47	44.16	40.42
YOLOv4	41.01	83.65	34.13	64.16
YOLOv4-Tiny	33.39	81.93	34.70	58.81
LSTM-SSD	39.00	79.80	33.10	51.40

Table 5.2: Precision and IOU values for each model trained on VDD- \bar{C} .

Model	FPS(GPU)	FPS(TX2)
SSD(MobileNetv2)	50	9
SSD(MobileNetv3-Small)	52	9
SSD(MobileNetv3-Large)	51	8
YOLOv4	50	5
YOLOv4-Tiny	88	35
LSTM-SSD	73	19

Table 5.3: Frames per second for inference.

5.4.4 Failure Scenarios

When considering a diver detector for use on an AUV, there is information of interest beyond accuracy, stability, and efficiency: when and why the detector fails. By inspecting the instances of false negative detections, we can gain some intuition on the circumstances of detector failures in the diver detector task. A significant portion of false negatives stem from one of two cases: divers not fully in the frame, or diver occlusions, as shown in Figure 5.8. We define a diver as not fully in the frame if an edge of their ground truth bounding box is on the edge of the frame. We define a diver occlusion as two ground truth bounding boxes with an IOU above 0.

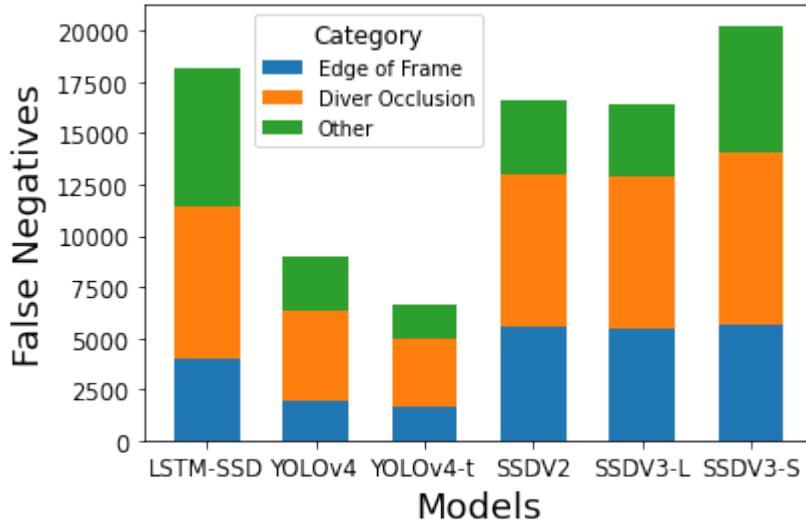


Figure 5.8: The source of false negative errors in different models.

5.5 Analysis of Temporal Stability

In robotic applications, it is often desirable to have temporal stability for object detections. That is, detections for a given object should be stable over time with respect to:

- *Translation.* If detected bounding boxes are not consistently located with respect to the ground truth bounding boxes from frame to frame, it is difficult to estimate the object’s location and trajectory.
- *Scale and aspect ratio.* Similarly, if detected bounding boxes have inconsistent scales and aspect ratios, it is difficult to estimate the object’s location and trajectory.
- *Fragmentation.* For any given diver that appears in the video, the diver should be consistently detected (*i.e.*, the object should not be undetected in one frame, then detected in the next, and so on). At worst, fragmentations make it difficult for the robot to confidently determine that the object is present, and at best, they increase uncertainty of estimations of the object’s location.

We adapt methods from [196] and [197] to evaluate diver detectors with respect to these three aspects of temporal stability. As in [197], we do not have ground truth tracks for the divers in our dataset and therefore computationally calculate ad-hoc tracklets for each diver by matching ground truth annotations from frame to frame with intersection over union (IOU) as in [198]. Using these tracklets, we then calculate the stability metrics from [196] as follows:

5.5.1 Translation error

The translation error of each tracklet's detection is measured with the center position error e_c . To calculate e_c , for each detection d in the tracklet, we find the standard deviation of the distance between the normalized center x bounding box coordinate, x_d , and the normalized ground truth x_g . We do the same for the y_d and y_g coordinates. The translation error is the mean of these standard deviations across all tracks. Formally, for each tracklet t :

$$e_c(t) = \sigma(x_d - x_g) + \sigma(y_d - y_g), \forall d \in t$$

Then the detector's overall translation error is

$$\frac{1}{N} \sum_{t=1}^N e_c(t)$$

5.5.2 Scale and aspect ratio error

For each detection, the aspect ratio error is defined as the ratio between the bounding box aspect ratio and the ground truth aspect ratio. The scale error is defined as the square root of the bounding box area over the ground truth area. To find the scale and aspect ratio error, we find the average standard deviations of each track's summed scale error $e_s(t)$ and aspect ratio error $e_r(t)$. Formally, for each tracklet t :

$$\begin{aligned} e_s(t) &= \sigma \left(\sqrt{\frac{w_d h_d}{w_g h_g}} \right), \forall d \in t \\ e_r(t) &= \sigma \left(\frac{w_d}{h_d} / \frac{w_g}{h_g} \right), \forall d \in t \\ e_{sr}(t) &= e_s(t) + e_r(t) \end{aligned}$$

Then the detector's overall scale and aspect ratio error is

$$\frac{1}{N} \sum_{t=1}^N e_{sr}(t)$$

5.5.3 Fragmentation error

For each track, we count the number of fragments as the number of times the track's status changes from detected to undetected or vice versa. Then the fragmentation error is the average number of fragments f per track, normalized by track length l :

$$\frac{1}{N} \sum_{t=1}^N \frac{f_t}{l_t - 1}$$

Because the translation and scale metrics rely on standard deviations of a tracklet's detections, they become meaningless for tracklets with only one detection. Our analysis therefore excludes any tracklets with only one detection.

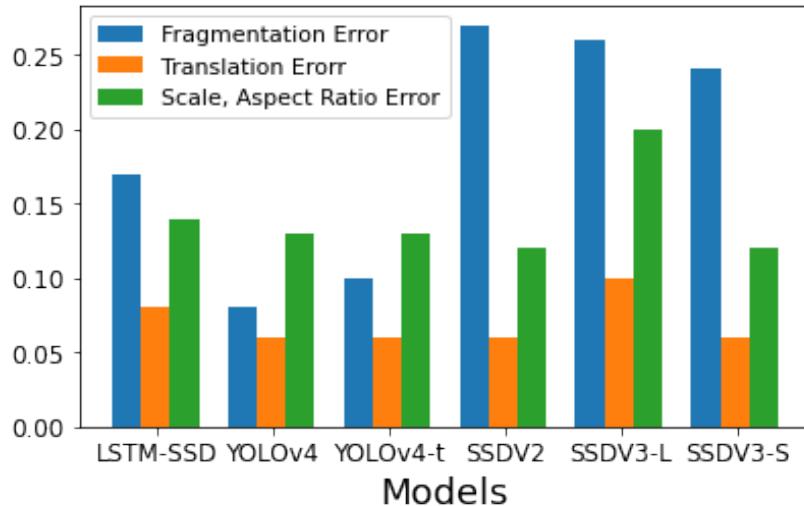


Figure 5.9: Measured stability errors for different models.

5.5.4 Stability Results

The average translation, scale, and fragment errors across all diver tracklets are calculated for each model using the equations discussed in Section 5.5 and shown in Figure 5.9.

Most of the models perform very similarly with respect to translation error and scale error, with the exception SSD-MobilenetV3-L, whose errors are higher than the other models'. Fragmentation error varies more across models. The YOLOv4 and YOLOv4-tiny models have the lowest fragmentation errors, despite not having the highest AP. Notably, while the LSTM-SSD and SSD-Mobilenets have comparable AP, the LSTM-SSD has a lower fragmentation error, indicating that it outperforms SSD in detecting divers consistently. Finally, we note that despite SSD-MobilenetV3-L having the highest AP, it also has the highest scale and aspect ratio error and the second highest fragmentation error, which suggests that it may not be the best choice for deployment.

5.6 Conclusion and Future Directions

In this chapter, we have presented VDD- \bar{C} , a new video context dataset for diver detection. This dataset is an order of magnitude larger than any previously released dataset of its type and has a greater variety of diver appearances, visual environments, and positions/orientations of divers than previous datasets. Due to this increase in data quantity and quality, the deep neural networks we trained on VDD- \bar{C} outperform those trained on DDD, a previous diver detection dataset. Our networks trained on VDD- \bar{C} demonstrated expected results: two-stage networks perform well but are slow, and one-stage networks such as SSD and YOLO are sufficiently robust and fast to be deployed on robotic platforms. The video object detection method we evaluated, LSTM-SSD, did not achieve precision as high as SSD and YOLO networks but ran at a much faster inference time than any network except YOLOv4-Tiny. Critically, we also evaluated models on their temporal stability, since consistent detections are important for interactive and collaborative applications of AUVs. We found that while SSD models generally had an edge over YOLO models in terms of AP, YOLO models had better detection stability in terms of fragmentation error and scale and aspect ratio error. In particular, SSDs had more than double the fragmentation error of YOLO models. This suggests that those whose work depends on consistent detections across frames may wish to evaluate vision models on stability metrics in addition to traditional metrics.

These contributions and results push the state of the art for robotic detection of divers forward significantly. While our use of models such as SSD and YOLO is a

standard in the field, the dataset we created improves their performance significantly. Furthermore, our evaluation of temporal stability adds much-needed information to our understanding of how these models perform in video contexts. This, along with our evaluation of a video object detector such as LSTM-SSD, has provided a great deal of useful information on how best to approach the problem of diver detection for interaction and collaboration.

Possible Areas of Future Exploration

As in previous chapters, we now provide a small set of possible future directions for this and related research.

Further Investigation of Video Object Detectors

Our investigation of LSTM-SSD demonstrated a higher-than-expected accuracy, with extremely quick performance and relatively low temporal stability error. Further investigation is certainly warranted, based on these results, into video object detection, as it appears to be a very promising method for this task. Indeed, if the accuracy of video-object detectors can improve somewhat, they could likely replace the currently used single-stage models entirely, at least in contexts such as diver detection where fast and stable detections are at a premium.

How Does Post Hoc Filtering Impact Temporal Stability?

As mentioned previously, post hoc filtering could be applied to detector outputs to improve the stability of detections over time. This could largely eliminate fragmentation error, although scale and aspect ratio, and translation error would be somewhat less affected. Adding this to the evaluation of the temporal stability of models would be a useful step. If, for instance, the fragmentation error of SSD models could be significantly reduced using filtering, it is possible that their high speed inference would then make them the ideal choice for deployment, rather than the current best option: YOLO-Tiny models.

Chapter 6

Predicting the Future Motion of Divers

Diver detection enables a wide variety of robot behaviors, including diver following [4,13]. Following behaviors are typically achieved by implementing a proportional-integral-derivative (PID) controller [73] with an error term based on the position of a diver bounding box within an image. However, other AUV capabilities such as leading divers or following from nonstandard orientations (such as side-by-side) require more than reactions to the diver’s current location. Furthermore, diver following methods can sometimes fail when the diver moves out of the visual range of the AUV too quickly, such that the robot loses sight of it. To enable diver leading and improve the robustness of diver following, a capability beyond diver detection would be required: predicting the future motion of divers. Human motion prediction has been studied in other contexts [199] and has even been used to control robot leading/following behaviors [200]. No previous work has attempted diver motion prediction for AUVs, and so in this chapter, we detail our efforts to apply deep learning methods commonly used for pedestrian motion prediction to the task of diver motion prediction for the purpose of expanding diver-relative navigation AUV capabilities.

6.1 Background:Predicting Human Motion

6.1.1 Diver Detection and Following

A pre-requisite for predicting the motion of a diver is the detection of said diver. This was the topic of the previous chapter, and many of the works cited in Section 5.1.3 have been developed with the goal of enabling diver following. For instance, the frequency-based diver detectors [175, 176], as well as the deep neural networks [13, 177], and even the diver identification methods [180, 181] were all designed with the goal of enabling diver following behaviors. While considerable work has been done in the domain of diver detection and diver following, a smaller amount of research has focused on robots leading divers. One of the few works to discuss non-standard positions for AUVs relative to the diver such as leading is Nad *et al.*, [14]. The majority of current research investigates robots following divers from behind, but no model exists yet for robots following underwater divers side-by-side or in front of the diver. These cases have been investigated for terrestrial robots and aerial robots [165, 200, 201] but given the significant differences between terrestrial and underwater robotics, these algorithms cannot be easily applied to aquatic robots. Underwater robots often utilize data that is noisy and distorted and have to operate in an environment in which both the robot and the leader (diver) can move in 6 degrees of freedom. Therefore, there is a need to develop algorithms that can not only improve current diver following by integration motion predictions but also provide an efficient way for robots to follow the divers on the side or lead divers from the front.

6.1.2 Human Motion Prediction

Predicting the future motion of entities has been a widely investigated problem, especially in terrestrial environments. Predicting the motion of pedestrians can provide important information to help autonomous vehicles avoid collisions with pedestrians. In the past few decades, many different models have been constructed to predict the trajectories of pedestrians, ranging from classical models that aim to explicitly define a motion model to recurrent neural networks that learn the motion model from a large dataset.

Social Forces Models

"Social forces" models utilize hand-crafted behavioral models to predict future motion trajectories in multi-pedestrian scenarios [202–204]. These models contributed to some of the first variants of motion prediction algorithms that took the interactions of entities into account. However, they face a drawback in that the model is pre-defined and is not able to learn from data, meaning that it can only capture the complexities of pedestrian motion that the designer is able to understand, and cannot infer new aspects of these behaviors through data.

Activity Forecasting

Activity forecasting models attempt to predict future actions or motions of individuals in a video. While Kitani et. al. [205] use inverse reinforcement learning to predict human paths, many other models use scene semantics to predict future events [206, 207]. While these models have proven to be fairly successful, the current state-of-the-art models utilize Recurrent Neural Networks (RNNs) to predict future events in a video sequence, such as the model proposed by Ranzato et. al. [208]. One variety of RNNs which have seen significant success in motion prediction are Long Short Term Memory (LSTM) networks, which we adapt in this chapter to the task of diver motion prediction.

6.2 LSTMs For Diver Motion Prediction

Our approach to diver motion prediction depends on the use of Long Short-Term Memory networks (LSTMs) [209] to predict time series data based on previous input. This method has been used in the past to predict the future motion of pedestrians modeled as two-dimensional points on a plane, but in our work we use LSTMs to predict the future locations of bounding boxes (the positions of divers) in future images based on the input of previous bounding boxes. Using our dataset from the previous chapter, VDD- \bar{C} , we train two types of LSTMs for this task: the Vanilla-LSTM and the Social-LSTM [199], a modification that attempts to encode human tendencies to move differently when close to other humans. In order to compensate for the motion of an AUV camera, we perform optical-flow-based stabilization on our bounding boxes.

6.2.1 Vanilla-LSTM

The Vanilla LSTM, as first introduced by Greff et. al. [209] refers to the most commonly used version of the LSTM model. This model features three gates - input, forget [210], and output. The output from the output block of the LSTM cell is connected back to the input of the cell. The sigmoid function is used as the activation function at each gate, while the hyperbolic tangent function is used as the input and output activation function. Our training methodology, explained further in Section 6.2 trains the LSTM network such that one forward pass through the LSTM cell corresponds to predictions one frame in the future.

6.2.2 Social-LSTM

The Social-LSTM model, proposed by Alahi et al. [199], shares the same base model as the Vanilla-LSTM but allows sharing of LSTM hidden states between targets. Developed to predict the trajectory of pedestrians, the Social-LSTM represents them as an (x,y) coordinate in the input frame, we modify the network in our approach so as to accommodate the two-dimensional bounding box associated with a diver.

Determining Neighbors of Divers: In the training process for our LSTM models, each diver in the frame has a hidden state associated with them. The Social-LSTM model enables encoding of spatial data by allowing the hidden state of a diver to influence the hidden states of its neighbors. We classify a diver as a neighbor of the current diver if:

1. The euclidean distance between the centroids of the bounding boxes of the two divers is less than the neighborhood size, and
2. The difference in the length of the diagonals of the bounding boxes of the two divers is less than a threshold value

The centroid distance is a measure of the proximity of the divers in the x and y dimensions, while the difference in length of diagonals encodes depth information for the divers in the frame. Since divers that are further away from the camera are expected to have smaller bounding boxes than divers that are closer to the camera, if the difference in the lengths of the diagonals of the two divers is greater than the threshold value, then

it signifies that the divers are significantly separated along the z-axis and hence should not be considered neighbors.

Social Pooling: Following the methods of Alahi et al., [199], we then generate a matrix associated with each frame of the shape ($nDivers \times nDivers \times gridSize^2$), where $nDivers$ is the number of divers in the frame. In this matrix, each diver in the frame holds $nDiver$ grids (one for each diver in the frame), each of size $gridSize \times gridSize$. Each binary grid for a diver is centered at that diver's position and conveys the grid cell in which a neighboring diver is present. If the diver is not a neighbor of the current diver, the grid associated between this diver and the current diver holds *False* in each cell. Since in our implementation, we use a grid size of 8×8 for an image of size 320×240 , we lose some specificity of spatial information but retain the relative spatial location of a neighboring diver to the diver in consideration.

Given the hidden state associated with each diver in the frame, we create the current hidden state for the frame as a matrix H , such that the rows of H are the hidden states for each diver. Next, as in Alahi et al., [199] we compute a social pooling matrix, S , with $nDiver$ rows, such that the j-th row of the matrix is:

$$S_j = GridMatrix_j^T H \quad (6.1)$$

where $GridMatrix_j$ is the neighborhood grid matrix for the j-th diver in the frame. Lastly, we compute the input to the LSTM cell by concatenating this social pooling matrix to the embedded input nodes containing bounding box coordinates. The forward pass of the LSTM cell is then given as:

$$\begin{aligned} a_i &= \phi(\beta(e_i)) \\ s_i &= \phi(\gamma(S_i)) \\ g_i &= a_i \oplus s_i \\ o_i &= LSTM(g_i, h_i, c_i) \end{aligned} \quad (6.2)$$

where e_i are the input nodes, S_i is the social pooling matrix, ϕ is a composite function of the ReLu activation and dropout layers, β is the linear input embedding layer, γ is the linear embedding layer for the social pooling matrix, and \oplus denotes the concatenation

operation. h_i is the current hidden state for the LSTM, c_i is the current cell state for the LSTM and o_i is the output from the LSTM forward pass. All operations are done for the i -th frame.

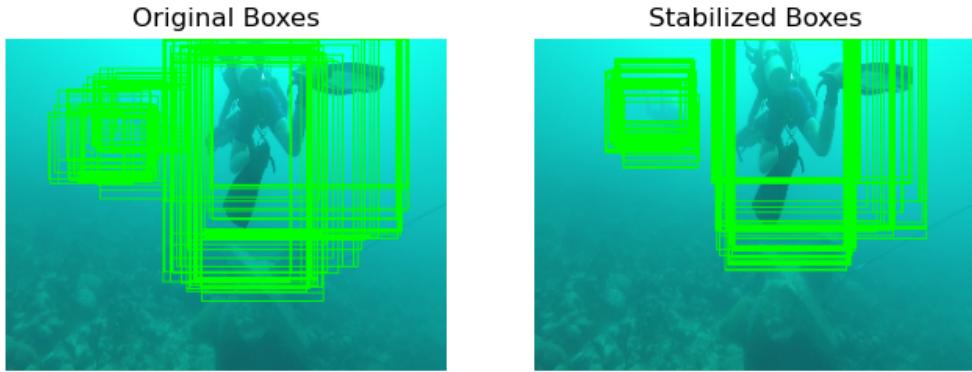


Figure 6.1: Stabilization of the bounding boxes over 50 frames for two divers using the dense optical flow based method. Bounding boxes are visualized in the frame of reference of the 50th frame.

6.3 Optical Flow Stabilization

Popular datasets used for pedestrian motion prediction, such as the ETH [211], UCY, and Zara datasets [212], consist of videos of pedestrians recorded from a stationary camera from a rooftop or window. This is common in the field of pedestrian motion prediction, however, it is extremely challenging to obtain videos recorded underwater that are from a completely static perspective. The videos in VDD- \bar{C} possess ego-motion of the camera in 6 degrees of freedom, as a result of which the raw position of the bounding boxes in pixel space is not consistent across frames. Hence, there is a need to stabilize and transform the bounding boxes to a single frame of reference before giving them as input to the LSTM networks.

In order to accomplish this, we utilize an optical flow-based method to calculate the transform between two frames, assuming pure 2D translation. We first use the Farneback method [213] to calculate the dense optical flow between two consecutive frames. We then remove any optical flow that lies inside the bounding boxes for the current frame, so

as to eliminate any motion of the divers in the frame. Next, we threshold the calculated optical flow against a small value to de-noise the flow data and calculate the mean of any remaining flow in the x and y directions to get the x and y transform respectively. These transforms are concatenated to the annotations for each frame in the video diver dataset. During training, we pre-process each input sequence so that all bounding boxes in the sequence are stabilized to the frame of reference of the last frame in the sequence. A result of the stabilization of bounding boxes over 50 frames can be seen in Figure 6.1.

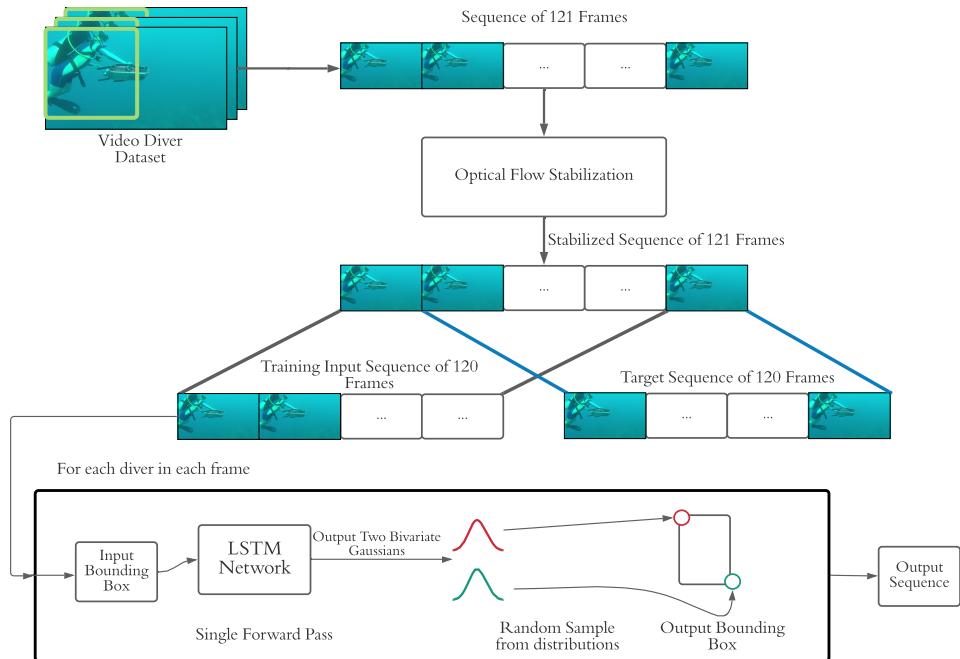


Figure 6.2: Training Pipeline for all LSTM models. VDD- \bar{C} is divided into sequences of 121 frames, each of which is stabilized using the dense optical flow method. The first 120 frames form the training sequence, while the second to 121st frames form the target sequence.

6.4 Training Methodology

Each instance of pre-processed training data consists of a sequence of 121 consecutive frames from VDD- \bar{C} . When stabilizing the bounding box, optical flow stabilization is

applied such that all bounding boxes are measured in the frame of reference of the 121st frame. Frames 1 to 120 then become our input to the model, while frames 2 to 121 become the target output. Therefore, the algorithm is trained such that one forward pass from the LSTM corresponds to a prediction one frame in the future. This training process is also described in Figure 6.2.

For processing the output, we use the same basic methodology as Alahi et. al. [199], but adapt it for two points representing opposite corners of the bounding box. For the i -th frame with N divers, the outputs from the LSTM cell are mapped to a linear layer with an output size of $N \times 10$, which defines two bivariate Gaussians for each diver, each parametrized by $\mu_x, \mu_y, \sigma_x, \sigma_y, \rho_{xy}$ - the mean for the x and y dimensions, standard deviation for x and y dimensions and the correlation coefficient between x and y dimensions respectively. A random sample from each Gaussian then defines the corners of the predicted bounding boxes for each frame. For the i -th diver at time step t , we define this as:

$$\begin{aligned} (x_t^{i,1}, y_t^{i,1}) &\sim N(\mu_t^{i,1}, \sigma_t^{i,1}, \rho_t^{i,1}) \\ (x_t^{i,2}, y_t^{i,2}) &\sim N(\mu_t^{i,2}, \sigma_t^{i,2}, \rho_t^{i,2}) \end{aligned} \quad (6.3)$$

The training loss is then calculated as the mean of the negative log-likelihood for each corner of the bounding box. For the i -th diver, this is given as:

$$L^i = -\frac{1}{2} \sum_j \sum_t \log\left(\frac{P(x_t^{ij}, y_t^{ij})}{\mu_t^{ij}, \sigma_t^{ij}, \rho_t^{ij}}\right) \quad (6.4)$$

We accumulate this loss and normalize it to the number of divers summed across all frames in the sequence.

6.5 Results

6.5.1 Metrics

To compare the results of our trained LSTMs, we consider their performance with respect to the following metrics:

- **Image Normalized Average Centroid Error:** We calculate the Euclidean distance between the centroids of the predicted and true bounding boxes. This distance is then averaged for all divers in a frame and all frames in the prediction length. All distances are normalized to the size of the image in the corresponding dimension.
- **Box Normalized Average Centroid Error:** This metric is identical to the Image Normalized Average Centroid Error, with the exception that distances are normalized to the dimensions of the true bounding box instead.
- **Average Intersection over Union:** Intersection over union (IOU) is a simple ratio of the area in common between two bounding boxes over the union of their areas. We calculate the Average IOU as the mean of IOU across all divers in all frames. While the centroid errors illuminate the deviation of the predicted bounding boxes, the IOU describes the relative overlap between the two boxes.

6.5.2 VLSTM vs. SLSTM

Figure 6.3 shows the results for the Image Normalized Average Centroid Error, the Box Normalized Average Centroid Error, and Average IOU for the Vanilla LSTM (VLSTM) and Social-LSTM (SLSTM), with both stabilized and unstabilized imagery. Generally, both VLSTM and Social-LSTM perform similarly, with the VLSTM slightly outperforming the Social-LSTM. Furthermore, from the plot for Box Normalized Centroid Error in Figure 6.3b, we can infer that the error in the predicted bounding box is reasonably greater than the dimensions of the true bounding box at roughly 30 frames of prediction. Furthermore, the stabilized LSTM models present an IOU of 0.3, 30 frames into the future, indicative of considerable overlap between the predicted and true bounding boxes. Therefore, we can conclude that the diver predictor utilizing the stabilized LSTM models makes reliable predictions for a length of 30 frames, or 1.5 seconds into the future for our input videos with a frame rate of 20 frames per second (fps). This prediction length, while short, will likely give a diver following algorithm enough foresight to improve its ability to not lose the diver when they make abrupt changes in motion. Lastly, in the plot for the image normalized centroid error, Figure 6.3a, we observe similar trends with VLSTM performing slightly better than the SLSTM and note that the size of the errors

never exceeds the size of the image itself.

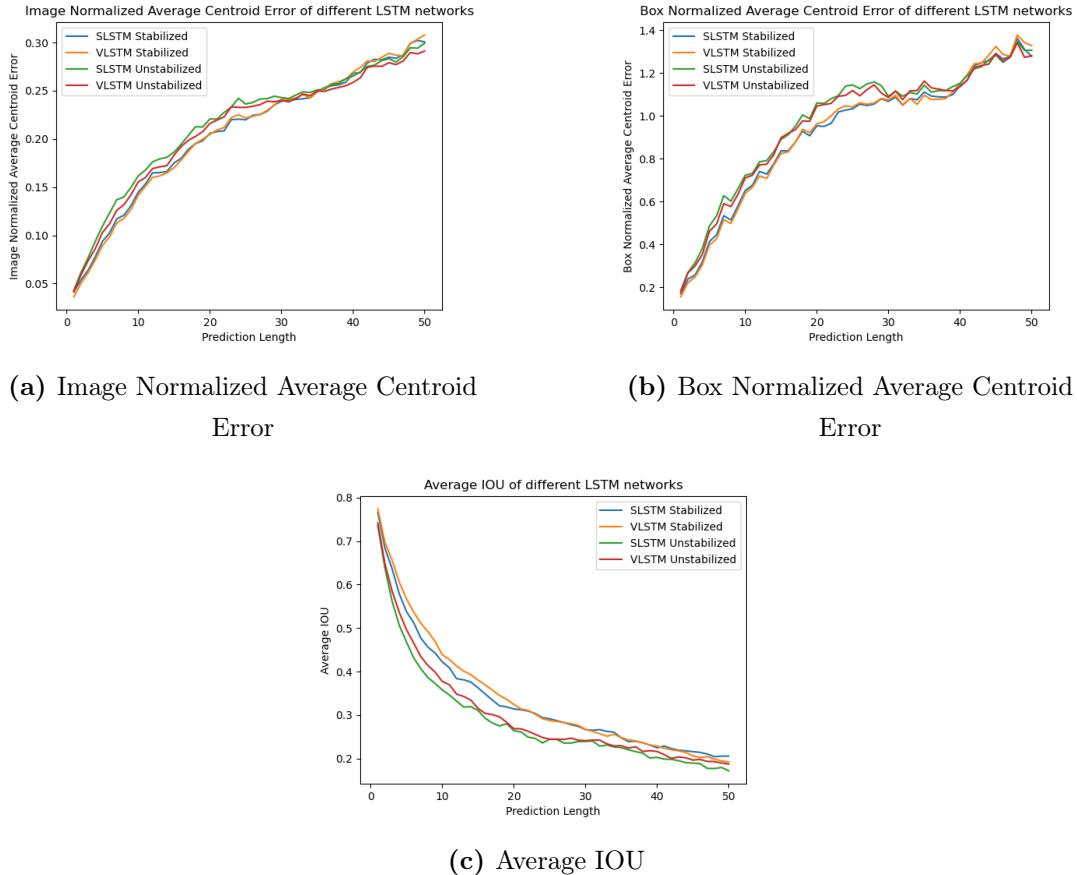


Figure 6.3: Comparison of metrics over 50 frames of prediction for stabilized and unstabilized Vanilla LSTM and Social-LSTM.

6.5.3 Effect of Stabilization

The performance of the stabilized and unstabilized LSTM networks can be seen in Figure 6.3. Across all three metrics, the stabilized LSTM models outperform the unstabilized models, with the difference being the most pronounced for the average IOU and Box Normalized Centroid Error. However, an inflection point occurs at a prediction length of 40-50 frames, beyond which the unstabilized models show at least similar performance

as the stabilized models. This result can be attributed to the instability of video sequences in the dataset which can cause rapid changes in the scene, and even distort the sequences such that divers are not present in the frame at a prediction length of 100 frames. Therefore, while stabilized models are more likely to estimate a well-defined trajectory for the diver, the unstabilized models are susceptible to making predictions with high variance and low consistency as can be seen in Figure 6.4. As the scene changes due to camera ego-motion, the divers do not maintain their initial trajectory anymore and the stabilized models have a higher probability of failing than the higher variance unstabilized models. This causes stabilized LSTMs to have higher errors for longer prediction lengths as compared to unstabilized LSTMs and we see the benefits of stabilization most strongly in prediction lengths of 5 to 35 frames.

Model Type	Vanilla LSTM	Social LSTM
Stabilized	558 ms	772 ms
Unstabilized	527 ms	737 ms

Table 6.1: Inference Time on a Jetson TX2.

6.5.4 Efficiency on Embedded Hardware

We present results for inference time of our models in Table 6.1 as measured on an NVIDIA Jetson TX2, a mobile GPU of the type commonly used onboard aquatic robots. An onboard implementation of our diver predictor would be paired with a suitable diver detector, such as the Tiny YOLOv4 diver detector we discussed in Section 5.3.3, which performs with an inference time of 35 fps on the Jetson TX2. As seen in Table 6.1, our stabilized Vanilla LSTM model can make predictions in roughly 0.5 seconds, giving us a prediction frequency of 2 Hz, which synchronizes well with a reliable prediction length of 1.5 seconds into the future. We can also note that stabilization takes roughly 33 ms for both models and that the Social LSTM takes roughly 200 ms more than the Vanilla LSTM, rendering the stabilized Vanilla LSTM our model of choice for an onboard implementation.

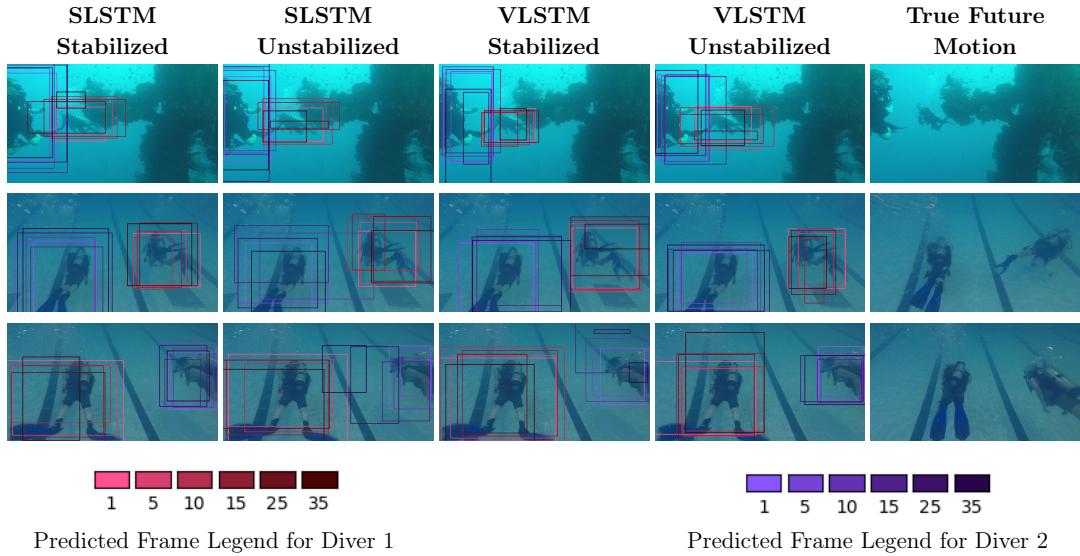


Figure 6.4: Comparison of future motion predicted by the Social-LSTM and Vanilla-LSTM, both stabilized and unstabilized. LSTM outputs are shown in the frame of reference of the last observed frame. For each row, the last figure represents the true motion of the divers 50 frames in the future (15 frames for row 1).

6.5.5 Applicability for Human-Robot Interaction

Understanding these metrics in the context of their intended use for improving diver following and enabling robots to lead divers is imperative. In these applications, while predicting the precise future position of the diver with high accuracy could be beneficial, it is not needed for robust diver following or robot leading performance. Rather, it is essential for aquatic robots to make predictions that accurately indicate the diver’s intended direction of motion. To analyze the directionality and consistency of the predicted trajectory of the diver, we visualize the outputs from our models for various scenarios in Figure 6.4.

In the first row, we present a scenario in the ocean with two scuba divers. In the future, both divers stay in their positions but rotate towards the right. While all LSTM models have similar results, the SLSTM most accurately captures the turning motion of the diver on the left and predicts boxes shifted more towards the right than the current position of the diver. Note that the unstabilized SLSTM here outputs larger boxes

that are more spread out. In the second row, two divers are swimming in a pool and the diver on the right is just beginning to turn towards the right. In this track, the stabilized Social-LSTM performs the best and predicts a well-defined path for the diver on the right to turn right, while the diver in the center is predicted to continue moving straight. The unstabilized models predict paths that do not follow a consistent direction of motion. In the third row, the diver is beginning to re-enter the frame. Here again, the stabilized Social-LSTM model shows better results in predicting a trajectory moving towards the left for the diver on the right as compared to the other models. Therefore, while the prediction of bounding boxes by the LSTM models at a length of 50 frames is not highly accurate, the stabilized models perform well at estimating the intended future motion of the divers. Since our models make predictions after observing only the last 5 frames and our input videos have a frame rate of 20 fps, this corresponds to observing only 0.25 seconds of motion. This enables our models to be sensitive to small changes in the diver’s motion, and hence be highly suitable for our intended applications.

6.6 Conclusion and Future Directions

This chapter introduced a new perception capability for AUVs, that of predicting the future motion of divers. We accomplished relatively accurate predictions up to 1.5 seconds in the future using Vanilla and Social LSTM methods originally proposed for pedestrian motion prediction. The accuracy of these methods was improved for our context by the addition of an optical flow based stabilization algorithm, which helped to reduce the effect of camera ego-motion. Both methods also run in near real-time on embedded hardware, making them potentially viable for deployment in field environments. When integrated into a diver following or leading program, these networks should enable an AUV to predict a diver’s motion and follow their trajectory more tightly, or determine when a diver has stopped following the AUV to stop and wait for them. Further investigation into this area is necessary, but this chapter’s introduction of these methods for diver motion prediction has established a base level of success to improve from, along with exploring relevant issues such as the elimination of egomotion.

Possible Areas of Future Exploration

As in previous chapters, we now provide a few possible directions for future research.

Three Dimensional Prediction

One obvious improvement to this method is adding three-dimensional information so that the area of a bounding box is not our only indication of the diver's distance from the camera. This could be achieved using stereo vision, sonar, or the method of relative distance estimation introduced in Chapter 7. Once the location of the diver can be determined in three dimensions, the same algorithm can be applied, but with a greater level of understanding of the future position of the diver. This could also be used to improve neighbor calculation, as the current method utilizes the diagonal length of a bounding box to estimate the distance to divers.

Richer Input State

Our current inputs to the LSTMs consist only of bounding box coordinates. It would be interesting to explore the use of more robust sources of information, such as a feature-space representation of the patch of the image containing the diver or the locations of a diver's limbs as calculated by a body pose estimator. These additions may lead to higher levels of accuracy by encoding more complex information, but their impact could have negative effects as well. For instance, using image information would make the method more susceptible to being affected by changes in the visual environment. Ablation studies would be required to determine what information would be useful.

Integrate IMU and Control

While our optical flow based method of bounding box stabilization achieved successful results and improved the downstream accuracy of our LSTMs, it is inherently limited by its two-dimensional nature. A more useful source of information would be the IMU and control inputs of the AUV itself, which could either be used as input to a different stabilization algorithm or simply added to the inputs for the LSTM, using the network to encode the relationship between egomotion and apparent motion of stationary divers itself.

Chapter 7

ADROC: Autonomous Diver Approach Using Monocular Vision

AUVs have been following divers since the early 2000's [4]. Despite the improvements in diver-relative navigation since then [13, 14], a topic that remains under-studied is the diver approach scenario, where an AUV locates a diver in the distance and navigates to a position directly in front of a diver, stopping at an ideal distance for interaction. This capability is a key one for HRI, as AUVs may be required to seek out divers to begin an interaction, to provide divers with relevant information, or to receive further instructions after a previously ordered task has been completed. For AUVs to truly augment the abilities of divers to work underwater, they must be able to operate independently, which makes the task of locating and approaching a diver a necessity. To enable AUVs to track and approach divers for the purpose of initiating interactions, we present the Autonomous Diver-Relative Operator Configuration (ADROC) algorithm. Utilizing human body information as a prior for a novel method of distance estimation, ADROC enables an accurate diver approach using only a single camera. While it is closely related to diver following methods, ADROC utilizes a novel method of estimating the distance to a diver: measuring the width of the diver's shoulders to roughly approximate distance. Additionally, while diver following algorithms simply follow a diver, ADROC must find the diver, approach them, and then terminate its operation, requiring both a state machine and a more complete understanding of the diver's position and distance relative to

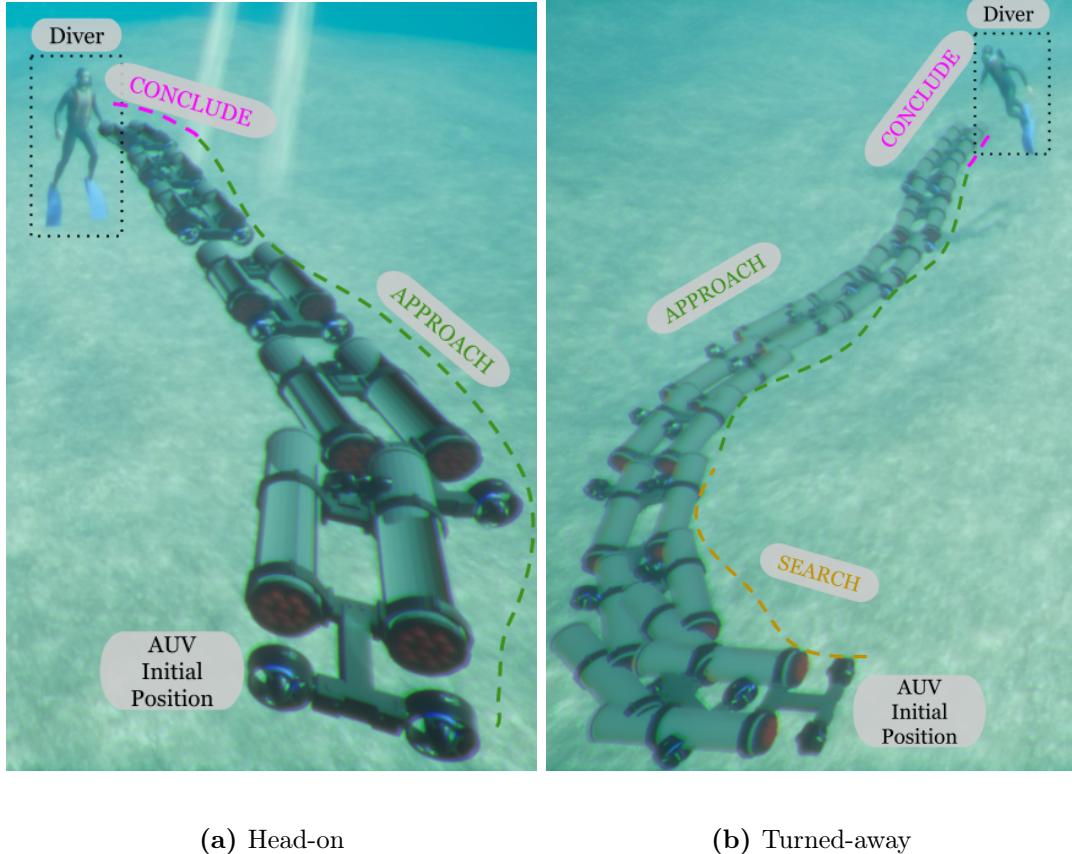


Figure 7.1: Two examples of ADROC diver approaches: (a) a diver is in the AUV’s field of view, so the AUV simply moves to the diver, (b) no diver is visible, so ADROC begins a search procedure, approaching once the diver is found.

the robot. The diver approach task had not been previously studied prior to ADROC, making this algorithm and evaluation formative for the task, hopefully the first in an improving series of methods.

7.1 Background: Diver Following and Approach

Prior to introducing our algorithm, we briefly discuss some relevant research on the topics of approach tasks in other environments, diver-relative AUV navigation including diver following, and diver distance estimation.

7.1.1 Robots Approaching Humans

The problem of robots approaching human targets has been extensively explored in HRI research, having been mostly considered in the context of service robots [214]. A large amount of robot approach research focuses on the effect of the proxemic behaviors of human interactants: the way in which the interactant's concept of personal space affects how a robot should approach them for the most optimal interactions [5, 214–217]. Research on robot approaches in field environments (which pose more of a challenge in terms of sensing and navigation) is rare, but some research has explored approaches in aerial environments [218], focusing on the algorithms necessary to enable an unmanned aerial vehicle (UAV) to approach a human interactant. However, there is no work on this topic for AUVs. The majority of research related to approaching humans with an AUV is instead focused on the broader problem of diver-based AUV navigation.

7.1.2 Diver-Based AUV Navigation

Diver-based AUV navigation is a common topic in AUV research for co-AUVs (AUVs designed to work collaboratively with humans) dealing with the varied scenarios in which an AUV must navigate its environment with respect to a diver. The most common form of this problem is diver following [13, 175, 219], a subset of the larger field of person-following research [165]. Many diver following works in recent years have utilized deep neural networks to detect a diver, then used closed-loop control algorithms to follow the diver based on detections. Some works have gone further than the simple scenario of diver following, such as the work of Nad et al. [14], which utilizes acoustic and visual information, allowing the robot to both act as a follower and leader. However, there are fundamental differences between following a diver and navigating to a position relative to a diver to facilitate further interaction. Approaching a diver requires complex algorithmic structures not typically found in diver following: the ability to actively search for lost or unseen divers, evaluation of the quality of relative position for interaction, and strict constraints on the distance between diver and robot. While many diver following methods contain some of the same capabilities necessary for our task (detection of a diver and navigation based on that detection), diver following is a different problem with different goals.

7.1.3 Estimation of Distance to Diver

A capability common to both our task (AUV approaching a diver) and other diver-based navigation tasks is the estimation of the relative position and range of a diver. The majority of diver-following methods do not estimate the distance to a diver accurately. Methods which do depend on the high fidelity data exploit sonar, stereo camera, USBL, and DVL sensors. A method for distance estimation that avoids the most expensive sensors is the use of stereo vision [220]. Stereo-based distance estimation could be useful to estimate the range to a diver at close range with a specific environment setup, but it will be prone to errors if not specifically tuned to its environment. This is because the camera parameters are affected when deployed in a new optical environment and this change could degrade the accuracy of the distance estimation. Additionally, the error in the depth estimate increases quadratically as the distance from the camera increases [221, 222], which limits the range of any algorithm depending on it. Our algorithm achieves reliable distance estimation using only low-cost monocular camera data, distinguishing it from the previously discussed work by improving distance estimation for diver-relative navigation without the use of complex and expensive sensors.

7.2 Designing Diver Approach

To guide our algorithm’s development, we select the following set of desiderata (features desired or required) from our understanding of the AUV interactions we plan to support. We focus on AUVs working alongside humans in a supporting role, with limited sensing and no global localization, as this is the environment where we believe our algorithm will be most useful.

ADROC Desiderata:

- D1:** Reliably navigate to a beneficial position for interaction with the diver regardless of the initial location and orientation of both parties.
- D2:** Accurately approach the diver regardless of diver movement after ADROC initiation.
- D3:** Approach the diver in a timely manner.

- D4:** Operate without global localization, 3D diver information, or any off-AUV computation whatsoever.

These desiderata are integrally tied to our expected deployments and create a very challenging problem. As such, we make the following assumptions to simplify the problem allowing us to formulate a feasible solution.

ADROC Assumptions

- A1:** The diver and robot are generally within the visual observation range of one another, and visibility is sufficient to detect the diver in the AUV’s cameras.
- A2:** The diver is generally upright with respect to the robot.
- A3:** Only one diver will be present in the scene.

Assumptions **A1** and **A2** are likely to be true in actual deployment scenarios. While we cannot guarantee ideal visibility conditions in the field, we must assume some level of visibility (**A1**) for the dive to occur. The diver’s orientation relative to the robot (**A2**) affects the accuracy of our body pose estimation algorithm, likely because the algorithm is trained on images of humans in upright positions (standing, sitting, etc.) Our approach algorithm continues to operate with divers in nonstandard orientations, but has the highest accuracy when the divers are in upright positions. This shortcoming could be overcome by developing a body pose estimation algorithm and training it on divers with a wide variety of body positions. Lastly, **A3** is not likely to be true in deployment scenarios, so future versions of this algorithm should account for this by enabling the algorithm to differentiate between divers and approach only a chosen target.

7.2.1 The ADROC Algorithm

ADROC has 3 components: a **state machine**, a **diver-relative position estimator**, and a Proportional-Integral-Derivative (PID) [73] **approach controller** tuned for diver approach operations. These components, along with an AUV motion controller and diver perception modules, are pictured on the right of Fig. 7.2 where the interactions between these components are shown. The diver perception modules are considered external to

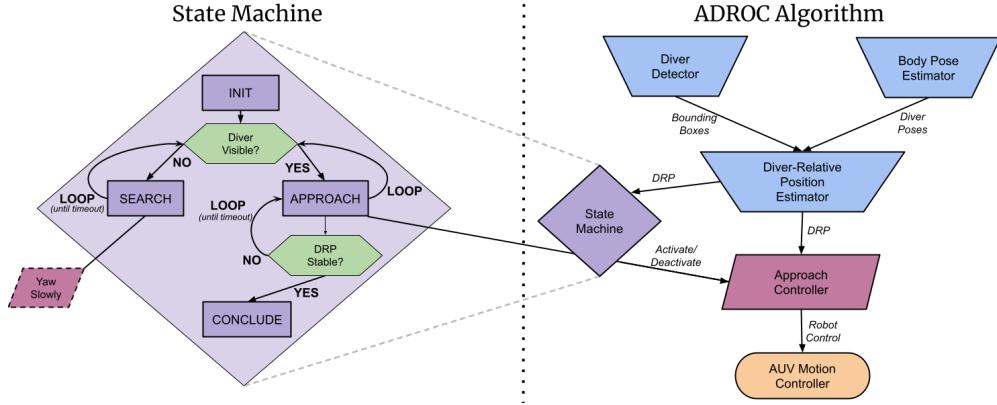


Figure 7.2: Diagram showing the ADROC algorithm (*right*) with detail on the state machine (*left*). Different components of the algorithm (perception, states, approach controller, and conditions) are presented in consistent colors and shapes.

the algorithm as they may be replaced with other systems if necessary; these modules are briefly discussed in Section 7.3.1. The diver-relative position (DRP) estimator, which is responsible for producing information on the position and distance of a diver relative to the AUV, is addressed in Section 7.3.2, and we discuss the approach controller in Section 7.3.3.

Therefore, we largely focus on the state machine portion of ADROC in this section, which is pictured on the left side of Fig. 7.2. The state machine consists of four states and two conditions which manage transitions between them. To begin, the state machine checks if a diver is currently visible in the scene. If there is an estimate of the diver's relative position available, it transitions from the INIT state to the APPROACH state, which activates the approach controller. During the APPROACH state, the approach controller manages the robot's orientation and distance relative to the target diver, continually checking if a diver remains visible. If at any point the AUV does not see the diver, it transitions into the SEARCH state. The SEARCH state triggers the AUV's search behavior, which is currently a slow yaw motion, turning in circles and scanning for divers. In the future, this could be redefined with a more complex behavior, utilizing information about the environment the robot is in and the last known location of divers in that environment. Once a diver is visible, the SEARCH state transitions to the

APPROACH state. Finally, the APPROACH state is successfully terminated once the diver-relative position is stable at a point close to the ideal position within a configurable margin of error. Once the AUV is stable in that position, the state machine transitions to the CONCLUDE state, which currently simply terminates the ADROC algorithm. To understand the APPROACH state and how the conditions responsible for state transition are determined, we must first explain the components of ADROC.

7.3 Implementation of ADROC

7.3.1 Diver Perception Modules

ADROC is currently designed to operate with diver bounding boxes and diver body pose estimations, which can be obtained from any source. We present the modules currently in use, which operate on monocular images, allowing ADROC to be used by any AUV with at least one camera. The modularity of our approach allows the adaptation of new sensors or perception algorithms in the future. Our two modules at this time are:

- **Diver Detection:** We use our YOLOv4-Tiny diver detector trained on VDD- \bar{C} from Chapter 5, Section 5.3.3 due to its relatively high accuracy with a fast inference time on embedded hardware. The detector outputs a set of detections with associated confidences, from which we select the highest confidence detection (due to assumption **A3**).
- **Diver Body Pose Estimation:** Body pose estimation aims to locate a set of body joints from a given image. We obtain a diver pose with a real-time pose estimation implementation [223] (a TensorRT implementation of human pose estimation [224, 225]). From the pose, we only utilize the left and right shoulder joint coordinates of a single body pose detection for our algorithm.

7.3.2 Diver-Relative Position Estimation

With the perception information being provided by our diver detector and pose estimation modules, we process the resultant data to produce a diver-relative position estimate (DRP), composed of a **target point** (TP) and **pseudo distance** (PD). This data is

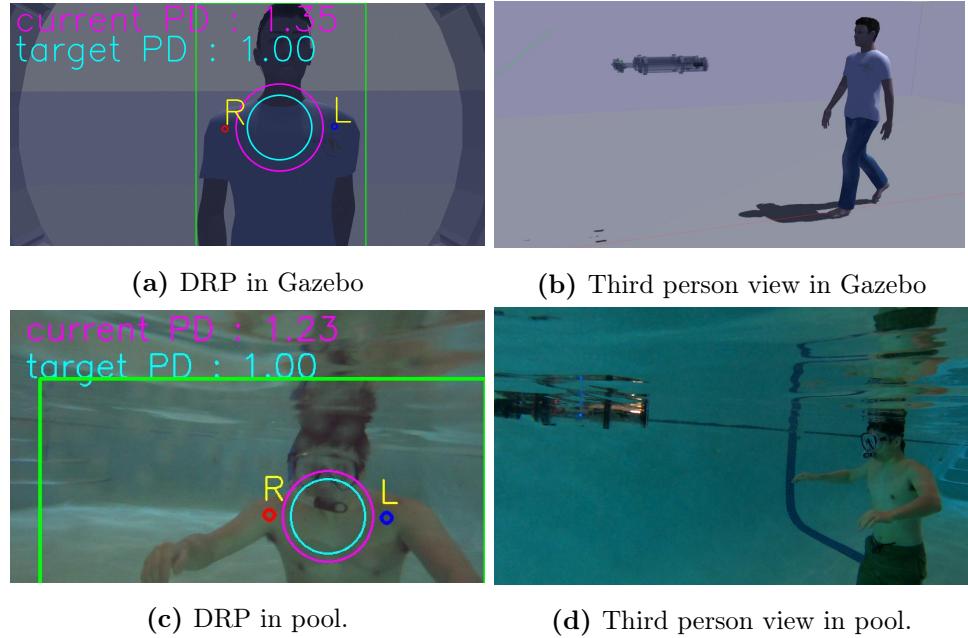


Figure 7.3: Diver-relative position (DRP) visualizations with a third person view, displayed in the Gazebo simulator and a pool scene. In the DRP visualization, the center of the circle is the target point while the radius represents pseudo distance.

used by our approach controller to center the diver in the frame of the camera and to reach a desired distance from the diver. In calculating both the target point and pseudo distance, we utilize all available information from both the bounding boxes and shoulder coordinates returned by our diver perception modules. Our algorithm is robust to missing information from either module.

We define the target point based on bounding box information to be the centroid of the bounding box. Alternatively, based on body pose estimates, we define the target point as the center point between shoulder joints. If both bounding boxes and body pose estimates are available, the target point is defined as the mean of these two points, but if only one is available, it is used as-is. Common approaches to estimate the range to a diver would be to use sonar data or calculate disparity and estimate depth from stereo imagery, which has its own challenges [226, 227]. Our algorithm is designed to work with only monocular vision available so that it functions even on the least sensor-equipped AUVs.

This creates a challenge, as monocular images do not contain sufficient information to accurately estimate distances, which we need to navigate to an appropriate distance relative to the diver. We overcome the problem of estimating the range to the diver without introducing new sensors by making a rough estimate which utilizes the shoulder width of the diver, called *Biacromial breadth* [228]. Biacromial breadth can differ based on demographic and individual variation, but we use average data available from national surveys [228], with the option of fine-tuning our estimate with specific measurements of diver shoulder width for an individual. For body pose estimation input, the shoulder width is the distance between the estimated shoulder points, but for diver detector input we treat the bounding box width as a rough estimate of shoulder size.

Table 7.1: The Pseudo Distance (PD) metric based on distance d between a diver and robot.

Distance ($d(\text{mm})$)	PD
$d > D_{ideal}$	$0 < PD < 1$
$d = D_{ideal}$	$PD = 1$
$d < D_{ideal}$	$PD > 1$

We propose a metric embedding distance information, *Pseudo Distance (PD)*, using diver detection and pose estimation as individual sources of shoulder width information (Eq 7.1). Psuedo distance is defined as inversely proportional to the ideal interaction distance. It is greater than 0 at any distance, less than 1 when farther from the diver than desired, 1 at the ideal distance, and greater than 1 when the robot is closer than the ideal distance (Table 7.1).

The following camera sensor, image, and physical specifications are used to develop PD: CMOS sensor size (mm), focal length (mm), image size (pixel), and shoulder width of a diver (mm , pixel). The detailed steps are:

1. Empirically select an ideal distance $D_{ideal}(\text{mm})$ for interaction between a diver and AUV.
2. Take the average shoulder width reported in [228] as the width of the diver's

shoulder. Alternatively, measure the true shoulder width of the diver.

3. Measure the shoulder width in image pixels ($w_{shoulder}$) from an image with the diver at the ideal distance $D_{ideal}(mm)$.
4. Select $target_shoulder_ratio$ as the ratio of the shoulder width ($w_{shoulder}$) to the image width (w_{img}).

The $target_shoulder_ratio$ is separately defined for diver detection and pose estimation information, as the source of the shoulder width estimate varies drastically in accuracy, with body pose estimation being more accurate. This is due to the fact that a bounding box changes size significantly based on the diver's body pose (*e.g.*, a diver with open arms yields a much wider bounding box than one with arms held at the side), while the distance between shoulder joints does not change based on the rest of the diver's body pose.

$$PD = \frac{w_{shoulder}}{w_{img} \times target_shoulder_ratio} \quad (7.1)$$

Whenever body pose estimation information is available, we default to using a pseudo distance based on pose data, as it is significantly more accurate. However, when such information is not available, we fall back to our estimate based on diver detection bounding boxes.

7.3.3 Approach Controller

The calculated DRP is passed on to our approach controller, which manages the approach procedure based on the target point and pseudo distance estimations produced by the DRP estimator. The approach controller maintains three separate PID controllers for the three axes of control it has: one controller for surge (forward and back), one for yaw (left and right), and one for pitch (up and down). These controllers are based off of error measurements (Eq. 7.2-7.4) comparing the target point to the image's center point and comparing pseudo distance to the ideal pseudo distance, which is defined as 1.0 (see Table 7.1).

$$error_forward = 1.0 - PD \quad (7.2)$$

$$\text{error_yaw} = \frac{\text{target_x} - \text{center_x}}{\text{image_width}} \quad (7.3)$$

$$\text{error_pitch} = \frac{\text{target_y} - \text{center_y}}{\text{image_height}} \quad (7.4)$$

The PID controllers in the approach controller operate on these errors in the standard fashion [73], which we will not detail here. Their parameters are tuned separately based on experimental results. The approach controller constantly calculates motor control values based on the available target point and pseudo distance, but only applies those controls to the motors (resulting in motion) when enabled by the ADCROC state machine switching to the APPROACH state. The approach controller limits its speed to 60% of the AUV's maximum velocity for safety.

7.4 Study VI: Approaching Divers Using AUVs

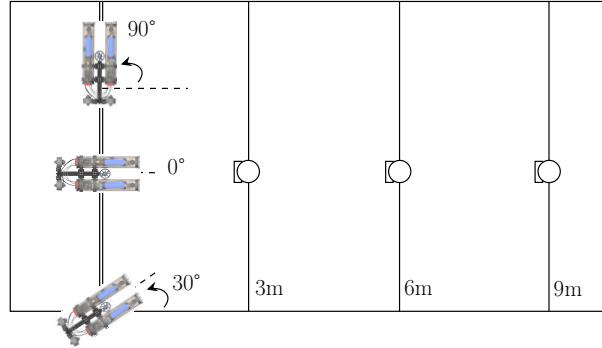
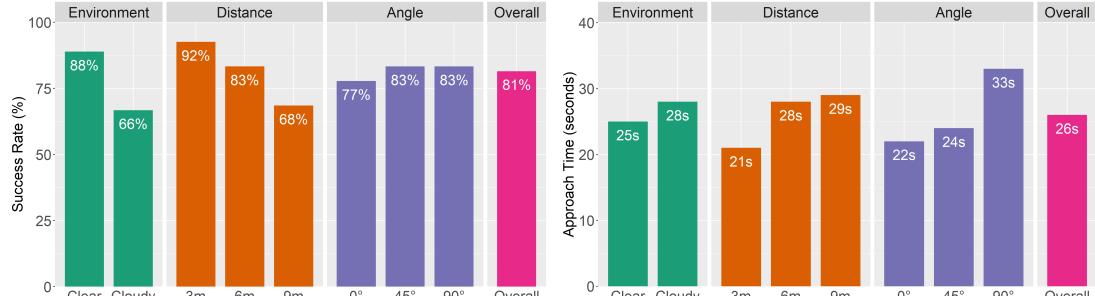


Figure 7.4: Pool experiments setup: three distances (3m, 6m, 9m), three angles (0°, 45°, 90°). Circles represent diver positions for the experiments.

7.4.1 Experimental Platforms

To test this work, we used the LoCO AUV [29], which is a modular, low-cost, open-source AUV equipped with dual monocular cameras and three thrusters. All the computation required for ADROC was done onboard, primarily using a Nvidia Jetson



(a) Success rate vs experimental conditions. (b) Approach time vs experimental conditions.

Figure 7.5: The success rates and average operation time of ADROC based on the trial, distances, and angles.

TX2 mobile GPU. Additionally, we used a LoCO simulation in ROS Gazebo [229] (Fig. 7.3) as a tool for developing the algorithms and evaluating them.

7.4.2 Pool Experiments

A set of pool experiments was performed to validate the ADROC algorithm's ability to successfully approach a diver. These experiments took place in two different pools, one with clear water and bright, even lighting, and another with murky water and dim, irregular lighting. A total of 9 divers were used as the target of the approach algorithm, with shoulder widths between 33cm and 48cm. Six of the divers were tested in the clear pool (Experiment #1) and three in the cloudy pool (Experiment #2). In each trial, the AUV was made to approach the diver from the combination of three initial distances (3 meters, 6 meters, 9 meters) and from three initial angles between the AUV and the diver (0°, 45°, 90°). These variables of distance and orientation combine to create nine distinct conditions of AUV approach. For each condition, two trials were conducted per diver for a total of 162 trials. For each trial, we allowed the AUV to enter the SEARCH state up to twice after the first APPROACH state. If the AUV reached the CONCLUDE state at the desired position before entering the third SEARCH state, we considered the case as a success, but as soon as the AUV entered the third SEARCH state after an APPROACH, the case was deemed a failure. Additionally, we recorded the time taken for each case, regardless of success.

7.5 Results of Study VI

7.5.1 Aggregate Data

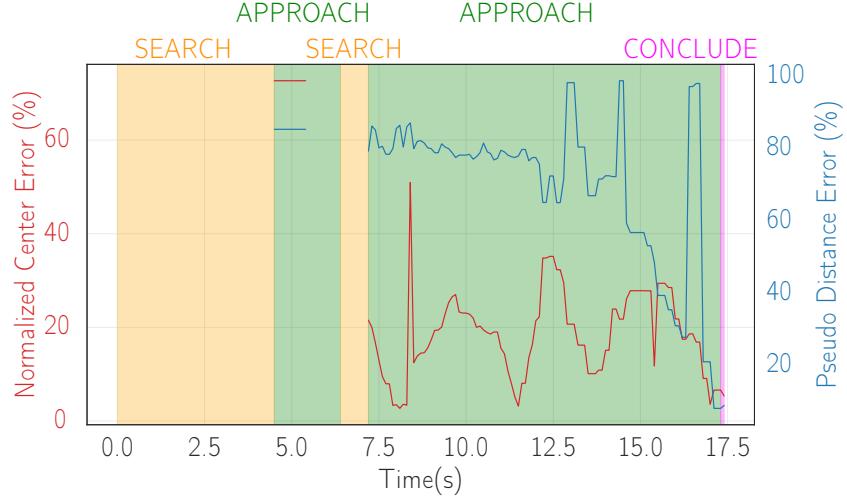
The summarized data in Figure 7.5 represents the averages of our results. We note that Experiment #1 has a higher overall success rate (88.9%) compared to Experiment #2, likely due to the better visual environment which led to higher quality perceptual inputs. Success rates reduce as the initial distance increases, most likely due to reduced accuracy of diver detection and pose estimation, which leads to fewer successful searches. Average times for an approach increase as well, but this is mostly due to the increased distance that the robot has to travel. When considering the effect of the initial angle, we see that approaches from the 0° condition have the lowest accuracy, but not by a statistically significant amount. This is likely due to random chance, as the difference in success between the 0° and the other angle conditions (which have the same success rate of 83.3%) is only 3 failures.

7.5.2 ADROC Runtime

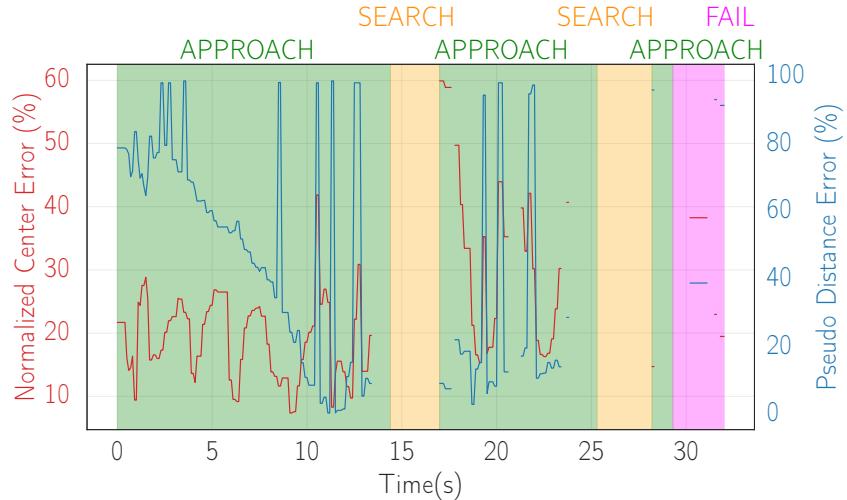
All components of ADROC were run on the onboard computers of the LoCO-AUV: the Raspberry Pi 4 and the Nvidia Jetson TX2. The diver detector and diver pose estimator, running concurrently on the TX2, ran at 15 *fps* and 10 *fps* respectively. Also on the TX2, the diver-relative position estimator ran at a set frequency of 20 Hz, while the ADROC state machine ran at 10 Hz. The only component which ran on the Raspberry Pi was the approach controller, which runs at a set frequency of 10 Hz. This frequency of the overall system (about 10 Hz) was sufficient to operate the AUV at relatively low speeds, although an improved frequency is possible with a more powerful GPU.

7.5.3 Individual Cases

Fig. 7.6 presents one of each success and failure cases from our experiments. In Fig. 7.6a, the AUV was at the turned-away angle, thus starting with the SEARCH state. During its second APPROACH state ($t=7.3\text{-}17.3s$), the AUV occasionally failed to detect pose estimates of the participant, and the diver detection estimate was used to yield the PD. The spikes ($t=12.9s$, $14.4s$, and $16.4s$) in the Pseudo Distance Error were caused when the



(a) Success example with narrow shoulder width diver (35cm), long distance (9m), and turned-away angle (90°)



(b) Failure example with wide shoulder width diver (45cm), long distance (9m), and head-on angle (0°)

Figure 7.6: (a) LoCO was turned away from a diver at the beginning. Through the SEARCH and APPROACH states, LoCO detected the diver and placed itself at the designed distance from the diver, (b) LoCO was facing a diver and started with the APPROACH state. However, spikes in PD error (bounding box only DRP estimation) caused unstable control and failure.

AUV controlled its distance to the participant based on the diver detection as opposed to body pose estimation. In Fig. 7.6b, the AUV started with the APPROACH state since it was at the head-on angle and was able to see the participant right away. The participant was detected by the diver detection consistently while the pose estimation only failed occasionally. More frequent PD estimation based on the diver detection caused an overshoot from the approach controller (more spikes from the beginning), and it resulted in failure. Additionally, the missing data points during the second APPROACH phase ($t=20.7\text{-}21.2s$) are consistent with system lag, possibly due to processing issues with the camera or some competition between processes for computational resources.

7.5.4 Types of Failures

Of the 162 recorded trials, 30 failures were recorded. These failures can be grouped into three categories: **search failures**, **early conclusion**, and **no conclusion**. **Search failures** (12 cases) were characterized by repeated returns to the SEARCH state, usually due to one of two issues: overshooting the initial rotation required for an approach, or receiving inaccurate detection data from the diver perception modules. Re-tuning the approach controller may help to reduce these failures. **Early conclusion failures** (11 cases) were caused by ADROC entering the CONCLUDE state prior to reaching the appropriate relative distance to the diver. These occurred almost exclusively for participants with small shoulder widths and were caused by the pseudo-distance based on bounding boxes reaching a stable point at a distance of 5 meters, prior to the AUV entering the detection range of the diver pose estimation. This type of failure can likely be entirely resolved by tuning the *target_shoulder_ratio* as described in Section 7.3.2. Lastly, **no conclusion failures** (7 cases) were caused by ADROC failing to detect a stable position while in the appropriate relative position to the diver. This failure is likely due to tuning issues with the approach controller, but should also be resolved by further improvement of the approach controller.

7.5.5 Limitation Experiments

To explore the limits of ADORC, a small number of trials were performed with more challenging conditions, in a deep pool similar to the pool used for Trial #1. These

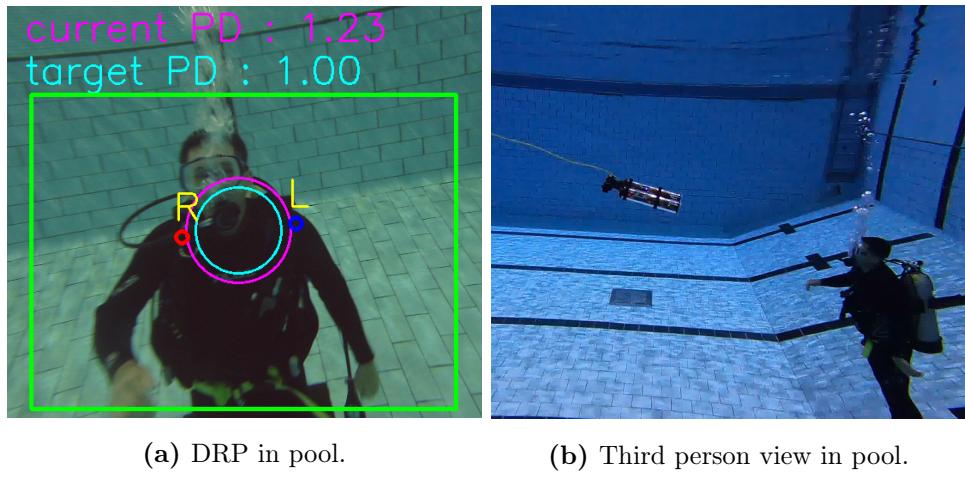


Figure 7.7: Diver-relative position (DRP) visualizations for a scuba diver in a pool.

trials are not included in the aggregate results above, because of the small number of participants for each. We repeated ADROC trials for one participant who was included in the normal trials in scuba equipment such as a wetsuit instead of the general swimwear that our participants wore. No reduction in accuracy compared to non-scuba trials was detected. Trials were also conducted with an adversarial condition, in which the AUV was pushed partway through an approach, mimicking a wave knocking the robot away from its chosen path. This also did not cause a reduction in accuracy. Lastly, trials were conducted up to 15 meters away from the diver. While the algorithm generally still functions, the AUV struggles to effectively switch from SEARCH to APPROACH as the approach controller sends yaw commands of a high intensity causing the AUV to quickly overshoot and miss its target. This could likely be improved with pseudo distance-relative tuning of the approach controller.

7.6 Conclusion and Future Directions

This chapter introduced ADORC, an algorithm for autonomous diver approach. This algorithm is comprised of a novel method for estimating distance to a diver, a PID-based approach controller, and a state machine. In our experiments, ADROC enabled an AUV to robustly approach participants with various shoulder widths from different

distances and angles. The advantages of our method are that it only requires a monocular camera, needs no extra sensors to estimate the distance to a diver, functions without global localization, and runs at real-time speed using on-board hardware. Our presented work clearly demonstrates the efficacy, reliability, and potential of our algorithm, despite the minimal information and sensors utilized. As it improves, ADROC and similar algorithms will become a standard capability for AUVs, allowing natural and straightforward cooperative work between AUVs and humans underwater.

Possible Areas of Future Exploration

As in previous chapters, we discuss a small number of possible avenues of further research.

Higher Intelligence Search State

In our implementation of ADROC, the AUV searches for a diver using a simple method: turning to the right. While this was sufficient for our simple test case, a more complex algorithm must be used in future implementations. One of the most important aspects of this algorithm will be stateful search. For instance, if the robot loses sight of the diver as they disappear to the left of the robot, instead of turning to the right, the robot should try to return to the last known location of the diver by turning left. If localization and mapping information is available, this search state could even take the form of a map coverage algorithm, taking into account the last known location of the diver and any locations they are expected to move to according to the mission plan.

Dialogue Initiation and Approach Vector

As an approach algorithm, ADROC's purpose is to position the robot within communication range of a diver in preparation for interaction. However, as the task of simply achieving robust approaches is challenging, no attention has yet been paid to selecting an appropriate approach vector or actually initiating interactions after approach. A great deal of research on this topic has been conducted in terrestrial environments, which should be considered. One promising approach involves determining the diver's current facing direction and selecting an approach which brings the robot into view as soon as possible, to increase the chance that the diver will be prepared for communication.

Chapter 8

POSH-G: Dynamic, Reconfigurable Gestures For AUV Control

The previous chapters in Part II have presented methods for the perception of divers including diver detection, diver motion prediction, and biological prior-based monocular diver distance estimation. These methods all serve as passive input to a variety of autonomous behaviors and HRI capabilities such as diver following, diver approach, and so on. Now, we turn our attention to a more active type of input: gestural control of an AUV. This is a common vector for human-to-diver communication and has been utilized as a method of AUV control since the earliest co-AUVs were introduced. However, the majority of existing AUV gesture control systems depend on large datasets of gestures and learn to recognize gestures directly from imagery. This makes them intractable to reconfigure, requiring the collection and labeling of large quantities of data simply to add a new gesture or revise an existing gesture. Additionally, gesture recognition systems for use underwater mostly recognize static gestures (a single position of the hands or arms) which can be difficult to hold underwater and are limited in their expressiveness. Indeed, most gestural communication among humans is dynamic: both untaught, non-linguistic gestures and defined gestural languages such as American Sign Language (ASL) primarily use dynamic, full-arm gestures.

With the goal of addressing these issues, we present in this chapter a gesture system for AUVs called Protean One-Shot Hand Gestures (POSH-G). POSH-G takes diver

POSH-G System

Pose estimation creates keypoint trajectories, which are analyzed and used to create training data, which KerasTuner uses to create and train gesture recognition models.

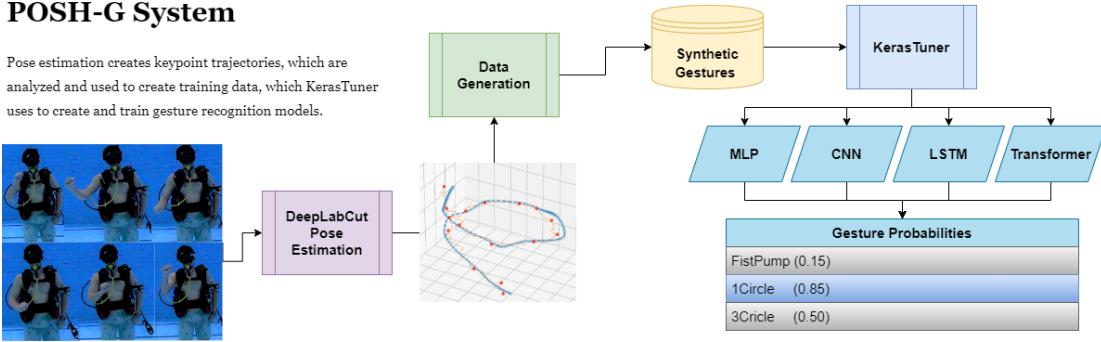


Figure 8.1: An overview of the POSH-G system, from pose estimation to data generation and tuning recognition systems.

body pose estimation trajectories as its input and utilizes a generation system based on Cabrera *et al.*, [230] which can generate thousands of example trajectories based on a single demonstration. This generated data is then used to train a variety of classifiers using Tensorflow and KerasTuner. With these two design choices, we eliminate the data burden of changing gestures and eschew direct learning from visual input to instead learn from keypoint trajectories. In the following chapter, we present a small dataset for diver body pose estimation (OceanPose) and discuss the training of body pose estimation algorithms with various network backbones using DeepLabCut [18] along with the application of filtering algorithms to improve the reliability of the output. We then present Protean, a gesture language for communicating with AUVs which has twenty-two symbols covering a wide variety of concepts including spatial and temporal modifiers as well as commands and nouns. Additionally, we discuss the Protean Gesture Language dataset which contains 3-5 demonstrations of each gesture from five participants in both lab and pool environments, which we use for the following generation and recognition steps. We next discuss the generation process we developed using keypoint trajectory input, which differs significantly from the input in Cabrera *et al.*, [230], requiring a number of modifications. Lastly, we demonstrate the effectiveness of the generation system by training a number of deep neural network classifiers on the generated data based on PGR and evaluate these classifiers on the remaining real data from PGR. An overview of the entire system can be seen in Figure 8.1 While the final recognition results from the classifiers are insufficiently robust for POSH-G to be deployed in real-world scenarios,

this approach is a first in the field of underwater gesture recognition. Furthermore, the research presented in this chapter establishes a base for future work of this type, moving away from the traditional methods of gesture recognition that have been the standard for AUV control.

8.1 Background: Gestural AUV Control

Before discussing the POSH-G system, we must first explore the background of gesture recognition as a whole, with particular emphasis on methods, datasets, and languages in use for underwater environments.

8.1.1 Gesture Recognition

Gesture recognition is a large field, with a wide variety of methodologies in use. The goal of gesture recognition can range from identifying specific movements on capacitive touch interfaces to complete recognition and parsing of full sign languages such as American Sign Language (ASL) [231]. There is a similar breadth in the types of sensors and algorithms applied to these tasks. The sensors used include cameras [232], depth cameras [233], special data capture gloves [234], dielectric elastomer sensors [235], and more. Additionally, a wide variety of algorithmic techniques have been used including traditional computer vision methods for segmentation and hand isolation, machine learning techniques such as multi-descriptor random forests, hidden Markov models, finite state machines, and deep neural networks such as RNNs, LSTMs, and so on. A proper summary of the field is out of the scope of this work, but a number of helpful survey papers have been written on the topic [236–238].

Cabrera *et al.*, and The Gist of a Gesture

One method of gesture recognition in terrestrial environments that is of particular note is the work of Maria Cabrera [239], particularly her work on one-shot gesture learning [230]. This paper introduces an N-shot learning method for gesture recognition based on calculating the “gist” of a gesture. By analyzing the trajectory of the user’s hand, inflection points in the path can be detected. These are then used to generate new trajectories by

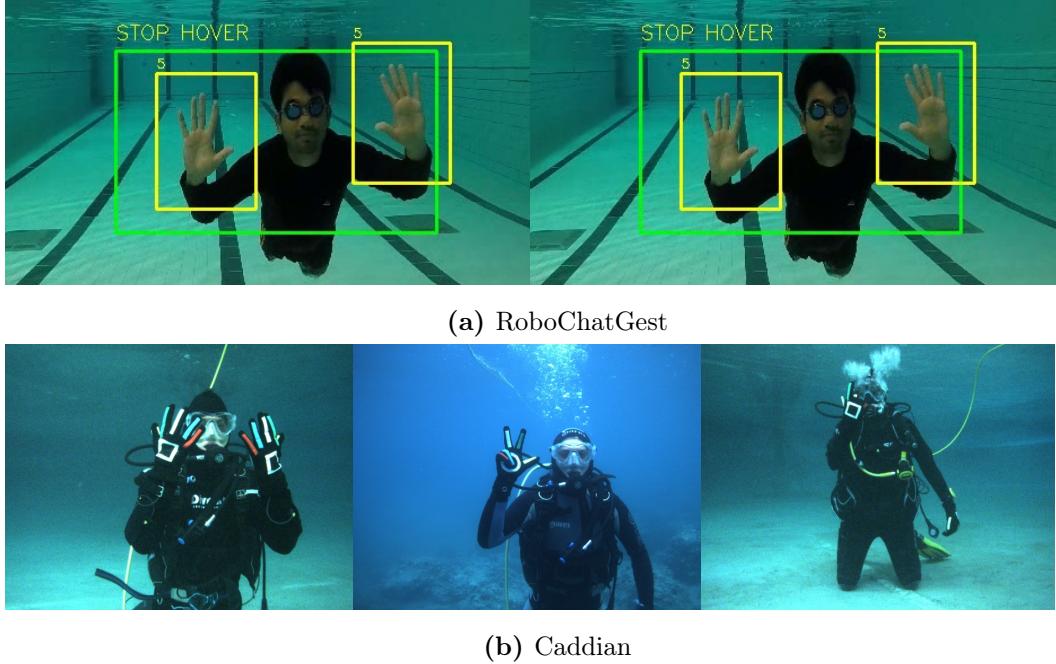


Figure 8.2: Examples of RoboChatGest [2] and Caddian [3].

sampling new points in an area around the inflection point and creating a new trajectory through the newly sampled points. This method is the basis for our data generation method described in Section 8.4, with a variety of changes. Most prominently, the algorithm in Cabrera’s work utilizes trajectories captured by a Microsoft Kinect, using their pose estimation pipeline. In our method, we utilize the output of a pose estimation network operating on a monocular camera’s video stream, which introduces significantly more noise into our source trajectories, a problem which is only increased underwater.

8.1.2 Underwater Gesture Recognition for AUV Control

Turning our attention to the more relevant topic of underwater gesture recognition for AUV control, methods and sensors are still highly varied, with three primary threads of research which are dominant. Firstly, the RoboChat [10] lineage of methods evolved from fiducial marker “flash card” methods to recognizing gestures performed with said markers using an iterative closest point-based recognition algorithm [45]. In more recent years,

this approach has been updated with RoboChatGest [2,240] (shown in Figure 8.2a), identifying unique hand pose “gestures” with a CNN. Second, the Caddian language [241] and dataset [3] (shown in Figure 8.2b) from the CADDY project [242] were used to develop a multi-descriptor random forest gesture recognition method [178]. The Caddian language consists mostly of static gestures similar to the hand poses of RoboChatGest, but contains a small number of dynamic gestures. In Caddian approaches, the hands of the diver are instrumented with colored gloves, which makes processing the image to find the hand and identify the hand pose a much more tractable problem, which has mostly been addressed using CNNs [243] and traditional computer vision techniques [244–246]. Lastly, a number of researchers have explored the use of gloves for diver gesture recognition [247–249]. Additionally, while it is not strictly a form of gesture recognition, recent work [250] introduced a method of estimating diver pointing vectors which will likely prove very helpful in gestural robot control.

Each of the above methods performs quite well in terms of accurately identifying the gestures defined for the system. However, they all share a critical weakness: their dependence on large datasets, which makes them difficult to update. Adding a new gesture requires collecting, processing, and labeling a large amount of data, which prevents the rapid adaptation of AUV interaction systems. Additionally, the majority of these systems utilize static hand poses as gestures, making minimal use of dynamic, full-arm gestures, which are more easily seen at a distance and potentially more expressive and intuitive. Lastly, methods which learn directly from visual input, are susceptible to the environment, the diver’s equipment, or even the diver themselves have the potential to significantly disrupt the efficacy of the system. For these reasons, we began our development of POSH-G by exploring one-shot or N-shot learning of dynamic gestures using pose estimation key points as input to reduce dependency on large datasets, increase visibility and intuitiveness of gestures, and abstract the gesture recognition process out of the image space.

8.2 Estimating Diver Body Pose Using DeepLabCut

The first step in the POSH-G framework is the estimation of body pose from the video stream. This is a well-studied problem, and a wide variety of standard algorithms

exist to solve it. However, none have been explicitly designed for use underwater and most applications of pose estimation systems underwater simply employ a pre-trained algorithm such as OpenPose [251], MediaPipe Pose [252], TRT_Pose [223], etc. In fact, in Chapter 7 we utilized TRT_Pose for the estimation of pseudodistance. However, for the application of gesture recognition, the accuracy of the pose estimator used as input is of utmost importance. For this reason, we developed a dataset called OceanPose, trained pose estimation networks with a variety of network backbones, and explored the effect of filtering algorithms on pose output in an attempt to produce a highly accurate diver body pose estimator.

8.2.1 DeepLabCut

We chose DeepLabCut [18] as our platform of choice for developing a pose estimation network because of its ability to provide accurate pose estimation with a small amount of training data. This is ideal, as collecting and labeling such data is a time-consuming task, particularly due to the overhead required to collect data underwater. DeepLabCut is a Tensorflow [253] based software library and toolset, developed for the purpose of enabling animal behavior researchers to create pose estimation models for their target organisms. For our purposes, we simply developed a body model for divers comprised of twenty-one body keypoints, created and labeled a dataset of swimmers in pool environments, and trained pose estimation networks on the said dataset. DeepLabCut allows for a great deal of configuration of the estimator, including the network backbone used. For our purposes, we trained three different-sized ResNet [254] and MobileNet [255] networks, along with five EfficientNet [256] networks.

8.2.2 The OceanPose Dataset

OceanPose is a dataset for training and evaluation of human body pose estimation algorithms in underwater environments. It is comprised of two entirely separate image sets, one for training and one for evaluation. Examples from both portions of the dataset can be seen in Figure 8.3.

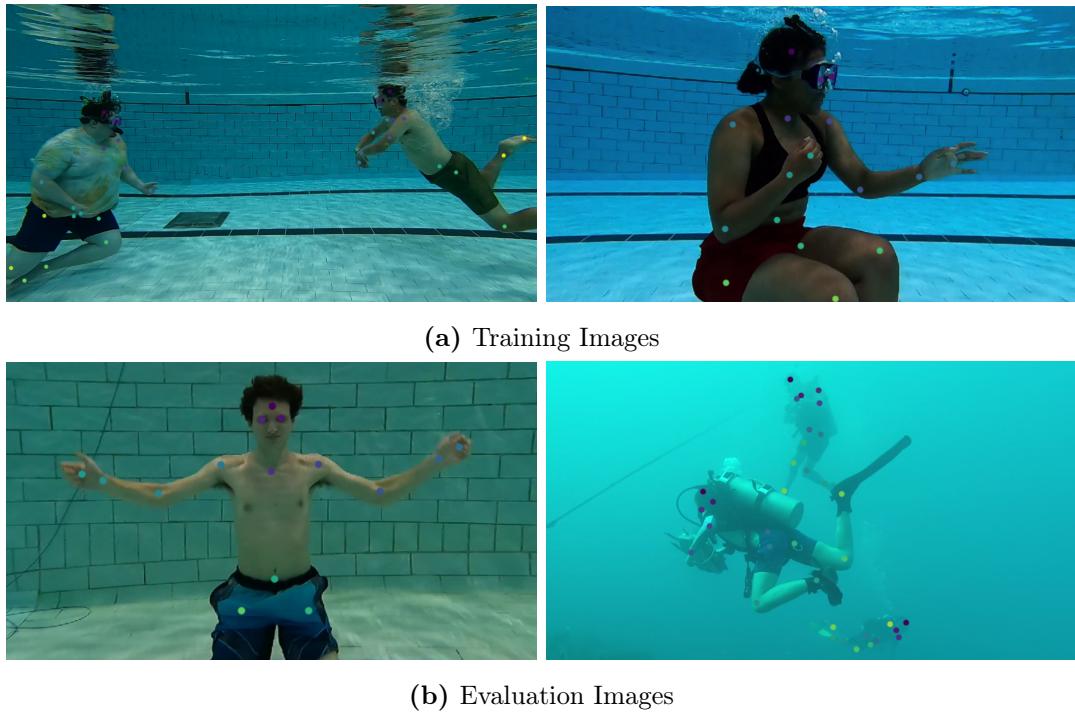


Figure 8.3: Examples of the training and evaluation data of the OceanPose dataset with labels. Note that the videos contained in the evaluation data are entirely separate from the videos in the training data, both in terms of the poses demonstrated and in terms of the divers shown.

OceanPose-Training

For the purpose of training our diver body pose estimation algorithms, we created the OceanPose-Training (OPT) set. Comprised of 1300 images of swimmers in pool environments, the OceanPose dataset mostly depicts a single diver several feet from the frame, though some images contain multiple divers or divers at a further distance. The data was selected from internal data sources, having been recorded over years of pool sessions. The goal of being used for training a body pose estimator was not designated during the recording, most of the recordings were of gesture demonstrations for earlier versions of the POSH-G project. None of the gestures or individuals included in the PGR dataset (Section 8.3.3) are present in the OceanPose dataset, though the pool environments are mostly the same. The images of OPT were labeled using napari, an image annotation tool used by DeepLabCut.

OceanPose-Evaluation

Along with the OceanPose-Training set, we created the OceanPose-Evaluation set, which is comprised of 486 images. OceanPose-Evaluation is sourced from entirely separate videos from OceanPose-Training so as to minimize overlap. While OPT only includes swimmers in pool environments, OPE contains images from several videos of scuba divers, some in ocean environments. Additionally, OPE contains images from the PGR dataset: videos of five different gestures with one participant demonstration for each. These data sources were chosen to create an evaluation set that provides an indication of how well the evaluated network will perform on our gesture demonstrations, but also an indication of the network’s performance on images from ocean environments. The images of OPE were labeled using napari, an image annotation tool used by DeepLabCut.

8.2.3 Training Configurations

OceanPose-Training is a relatively small dataset, even for DeepLabCut’s purported ability to train on small datasets. Therefore, we also utilize the MPII Human Pose Dataset [257], a standard in pose estimation research. However, MPII is a general-purpose dataset, with a wide variety of scenes depicted. Although MPII does contain

Data Configuration	# Images	Initial Weights
MPII	17,385	Pretrained ResNet/MobileNet/EfficientNet
OPT	1,303	Pretrained ResNet/MobileNet/EfficientNet
MPII + OPT	1,807	Pretrained ResNet/MobileNet/EfficientNet
MPII ft OPT	1,303	MPII trained to 150,000 iterations.

Table 8.1: The dataset configurations used for training DeepLabCut models.

some images of underwater humans, the majority are in terrestrial environments. To determine what would serve our purpose best, we experimented with a variety of datasets for training: using only MPII or OPT, using the underwater images from MPII along with OPT, or fine-tuning on OPT after initially training on MPII. The different dataset configurations are listed in Table 8.1. Each of these configurations had 5% of its images set aside as an internal evaluation dataset.

8.2.4 Pose Estimation Results

Each network backbone was trained on every data configuration until loss values plateaued. For MPII, this was 150,000 iterations, 50,000 iterations for OPT and MPII + OPT, while MPII ft OPT was only trained for 20,000 iterations. This smaller number of iterations was selected to avoid overfitting the networks to OPT and losing the benefits of training on MPII. Once trained, each network is evaluated against its internal 5% testing data as well as the OceanPose-Evaluation dataset. Some issues were encountered in training and evaluation stemming from issues in DeepLabCut’s training dataset creation process, which has led to some gaps in the evaluation data. However, as these networks are not the final goal of this chapter, we will continue with the data we have.

Self-Evaluation Accuracy Results

Considering the self-evaluation results shown in Figure 8.4, we can see reasonable levels of error for all networks and data configurations. The four metrics reported are the average pixel error of a predicted keypoint on training data, testing data, training data when only considering keypoints with probability greater than 0.5, and the same cutoff for testing data. Errors in general fall below 30 pixels total, with lower errors when

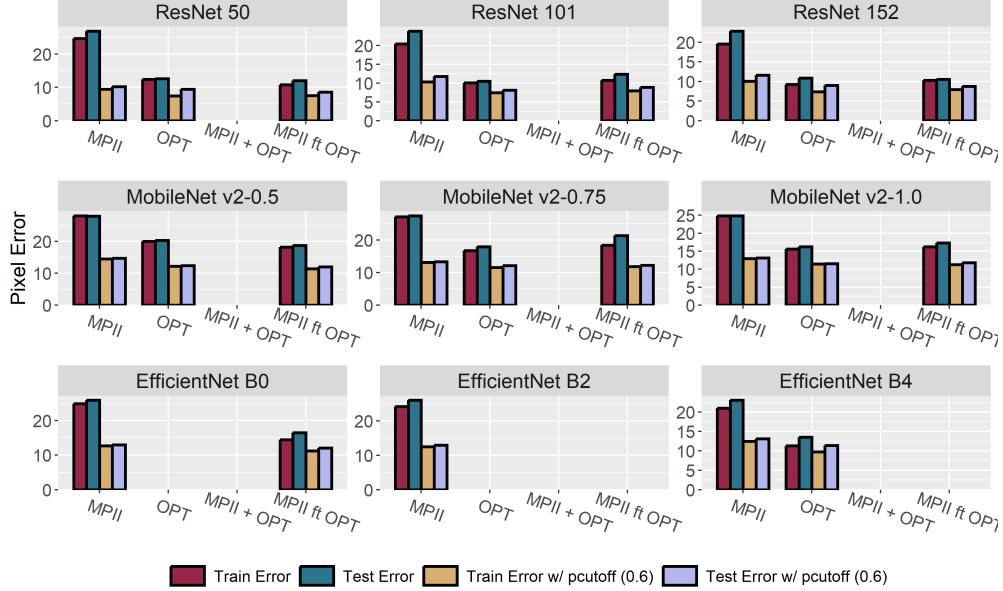


Figure 8.4: Performance of DLC models trained on a variety of dataset configurations, evaluated on their own evaluation dataset. Some data is missing due to unresolved DLC dataset indexing errors.

considering the keypoint probability. In terms of comparisons between network types, the ResNets have the lowest error levels, as expected. In general, however, every network performs within expected limits, with minor fluctuations throughout.

OceanPose-Evaluation Accuracy Results

In terms of the evaluation results on the OceanPose-Evaluation set, we find a much different story. Average pixel errors rise as high as 600 pixels in this context, where we report an average error on testing data along with errors at two different pcutoff values, 0.5 and 0.8. The network backbone used affects the final accuracy significantly: ResNet and EfficientNet networks perform relatively well, with error scaling as the network used grows larger. However, MobileNet errors remain high at all times, even with different dataset configurations. On the front of dataset effects, the MPIII trained models perform poorly on OceanPose-Evaluation, with average errors above 200 pixels, even with probability cutoffs applied. Configurations that use OceanPose-Training do much better, with the lowest errors achieved by the ResNets trained on the MPIII +

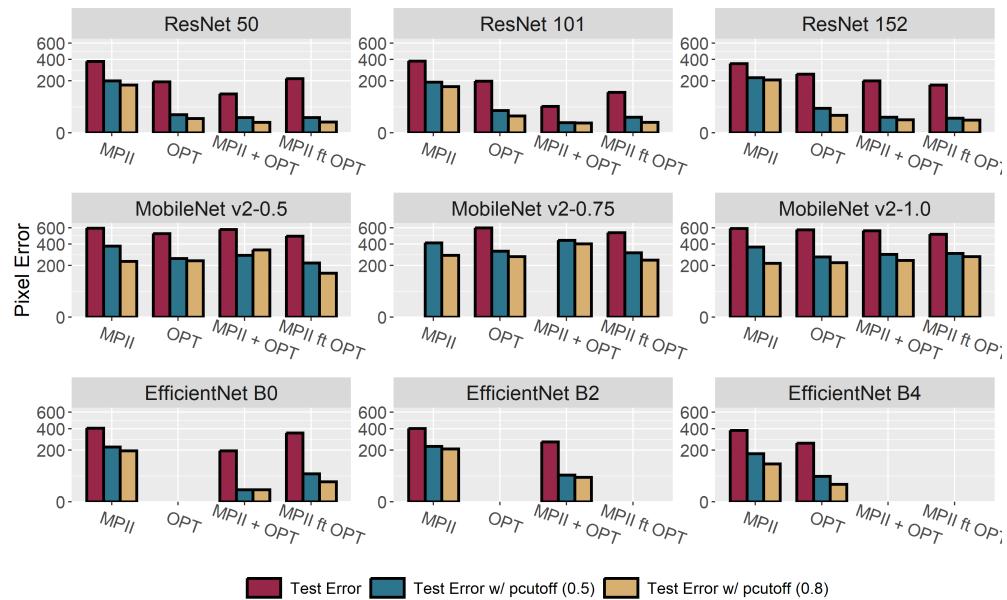


Figure 8.5: Performance of DLC models trained on a variety of dataset configurations, evaluated on a common evaluation dataset. Some data is missing due to unresolved DLC dataset indexing errors. Y axis is in square root scale to better display low pixel error values.

OPT and MPII ft OPT configurations. The error without a pcutoff remains high even in those cases, indicating that a relatively large number of low-confidence keypoints are incorrectly predicted.

Model Selection

Based on these results, we select the ResNet 101 network trained on MPII + OPT as our pose estimator for our further development in the chapter, which achieved a general test error of 51 pixels, with pcutoff errors of 7.7 pixels with a 0.5 pcutoff and 7.1 pixels with a 0.8 pcutoff. We chose this size of ResNet to balance the need for accuracy and efficiency, If a faster inference time is required, the EfficientNet B0 and Resnet 50 networks trained on MPII + OPT would be appropriate, having achieved similar error levels with faster inference times. With a pose estimation network trained for our purposes, we can now turn to designing and producing a dataset for our gestural language.

8.3 The Protean Gesture Language and The PGR Dataset

Protean is an extensible dynamic gesture language for AUV communication and control. Named after Proteus, the prophetic shapeshifter of Greek myth (also the namesake of our UHRI software presented in Chapter 9), the name is an adjective that means “tending or able to change frequently”. This indicates the goals of the language both as a testing language for the POSH-G one-shot gesture recognition system and beyond, as a language for AUV control. Twenty-two gestures are currently defined, but more can be added simply by using the gesture generation method expanded on in Section 8.4.

8.3.1 Designing A Gestural Control Language

Designing gestures is a difficult task, particularly in the context of underwater use. Hand signals are already used by divers for communication, meaning that there is an existing vocabulary that we can either choose to integrate into our AUV gesture language or must avoid overlap with. Additionally, divers must use their hands to help them swim, so gestures cannot be too long or require particularly difficult motions. Furthermore, the method we plan to use to recognize Protean has an impact on design choices, primarily in the following ways:

- Gestures cannot be static, as we need a gesture trajectory to analyze and replicate.
- Gestures must be distinct without taking hand pose into account.
- Gestures cannot be too short, otherwise, we risk not having enough data to generalize from.

8.3.2 Protean: A Dynamic Gesture Language

With these limitations in mind, we developed Protean through an iterative process, beginning by selecting a set of concepts relevant to providing an AUV with commands. As with our development of AUV-to-human communication, the concepts to be communicated were sourced from our experience working AUVs as well as the content of existing diver sign languages. The initial ideas were written down, with similar concepts removed or combined with others, finishing with 22 gestures, organized into five categories: Simple, Control, Verb, Context, and Noun. Simple gestures were created both as basic test cases for gesture recognition, and to represent simple concepts such as a circle or a square. Control gestures deal with conversational control, providing responses to a “speaking” AUV. Verb gestures communicate an action of some kind, and can be combined with a Noun gesture. Lastly, Context gestures provide helpful information about time and place, as well as providing personal pronouns such as G_{You} and G_{Me} .

Once the concepts were selected, we followed a design policy of representation, mimicking the concept whenever possible. For instance, the gesture G_{Diver} mimics air bubbles escaping from a scuba diver’s breathing apparatus, the G_{Follow} involves following one hand with the other, and so on. Some gestures cannot be easily represented, such as $G_{Affirmative}$. In these cases, we attempted to create a dynamic gesture with some similarity (but not exact parity) with common dive signals. For instance, $G_{Affirmative}$ consists of the diver making the “Okay” symbol with their hands and then making a nodding motion by rotating both arms at the elbow. Even though the recognition system will not detect the static hand pose, we define as a part of $G_{Affirmative}$ to provide a mental association with the common dive signal for Okay for those with experience in such languages. Twenty-one of the gestures in Protean can be seen in Figures 8.6, 8.7, & 8.8.

Gesture	Gesture Demonstration
$G_{FistPump}$	
$G_{1Circle}$	
G_{Square}	
$G_{Affirmative}$	
$G_{Negative}$	
$G_{Attention}$	
$G_{RepeatLast}$	

Figure 8.6: Gestures 0-7 gestures of the Protean Gesture Language, demonstrated by a scuba diver. Note that there is also a $G_{3Circle}$, which is simply $G_{1Circle}$ three times.

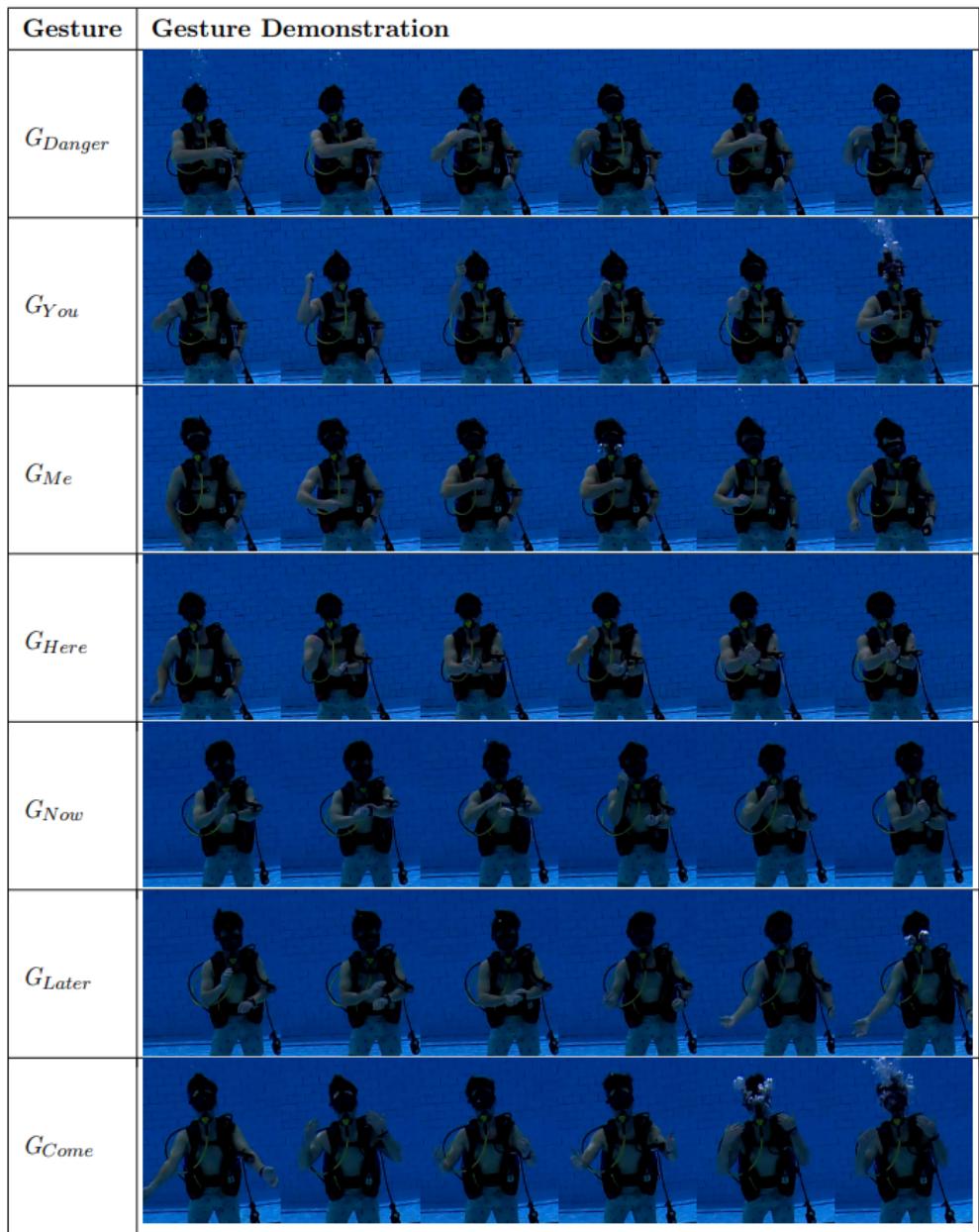


Figure 8.7: Gestures 8-14 of the Protean Gesture Language.

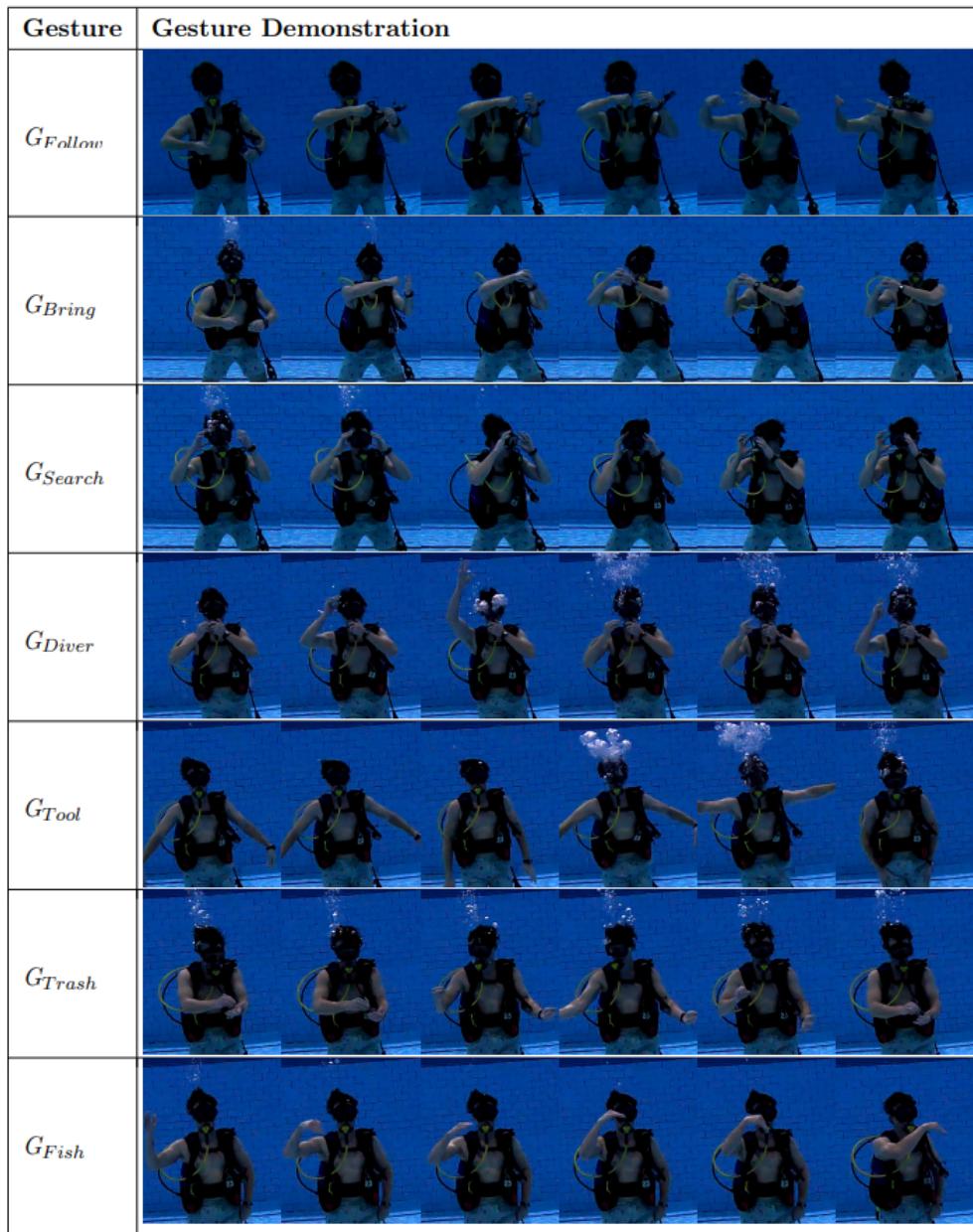


Figure 8.8: Gestures 15-21 of the Protean Gesture Language.

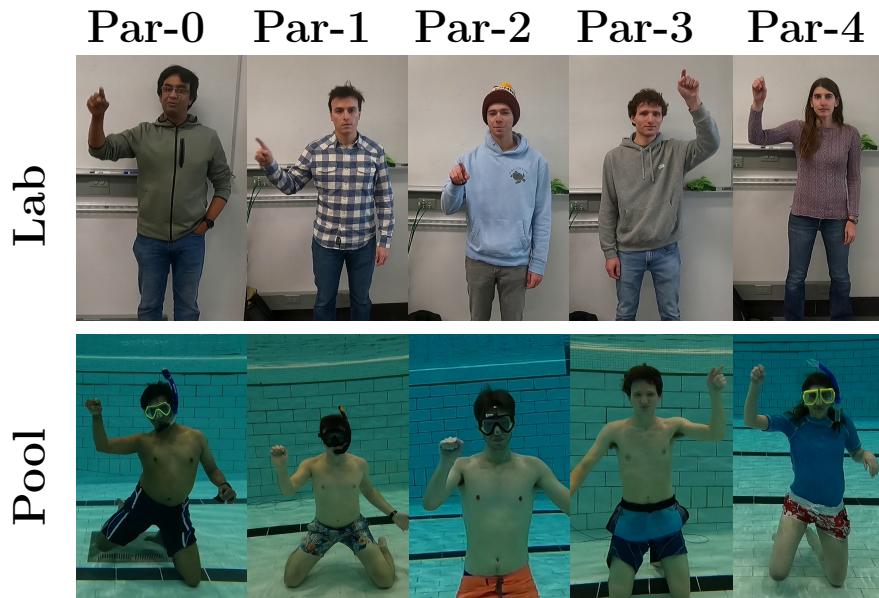


Figure 8.9: The divers and environments present in the PGR dataset. Par-1 and Par-4 also have videos wearing scuba gear. Note that Par-3 is left-handed.

8.3.3 The Protean Generation and Recognition Datatset

For the purpose of creating a one-shot recognition system for Protean, we must create a dataset of gesture demonstrations. The Protean Generation and Recognition (PGR) dataset consists of demonstrations of each Protean gesture performed by five participants. Each participant demonstrated all twenty-two gestures 3-5 times in a lab environment, then 3-5 times in a pool environment (example frames can be seen in Figure 8.9). This enables cross-domain learning of gestures (a topic not explored in this chapter, see Section 8.7), where a demonstration from a lab environment could be used to learn underwater gestures, further reducing the difficulty data collection. The demonstrations were recorded using a GoPro Hero6, then split into individual video clips, each containing a single gesture. Demonstrations are not predefined as being training or testing data, as the dataset is intended for one-shot learning. During data generation, one demonstration per participant will be selected as the generation source, with the remaining demonstrations being used as the evaluation samples. In addition to these demonstrations, two participants recorded demonstrations of Protean while wearing scuba equipment, these demonstrations can be used for evaluation. Lastly, while

all gestures were performed as correctly as possible, par-3 is left-handed, and performs every gesture in a mirrored manner, using the left instead of the right and reversing left/right movement directions. This provides an opportunity to explore ways to adapt gesture recognition to different handedness and varying users' ways of gesturing.

8.4 Generating Gesture Data

Based on the method presented by Cabrera *et al.*, [230], we now present a method for generating gestures based on a single demonstration. Like Cabrera, our method finds inflection points in trajectories, samples new points around them, and generates a trajectory through them. However, unlike Cabrera, our method utilizes input from a monocular vision-based pose estimation network on underwater data, which requires a number of small modifications throughout. The overall generation process can be seen in Figure 8.10.

8.4.1 Collecting Trajectory Data

As an initial step, the data from PGR must be transformed into keypoint trajectories over time. We use DLC-Live [258] to process the images of each video in PGR sequentially, recording the estimated keypoints in Python pickle files.

Pose Filters

For the task of generating training data, even small disturbances in the input data can have a large effect, so filtering or interpolating data can be quite useful. While we do interpolate trajectories in the data generation phase, we chose to also apply filters directly to pose estimation. Four filter configurations are utilized: no filter, a one ϵ filter [259], a moving average, and a Kalman filter [260]. These filters are applied by DLC-Live directly, meaning that they can be used in our collection step for the generation process, or used in real-time inference for input to gesture recognition.

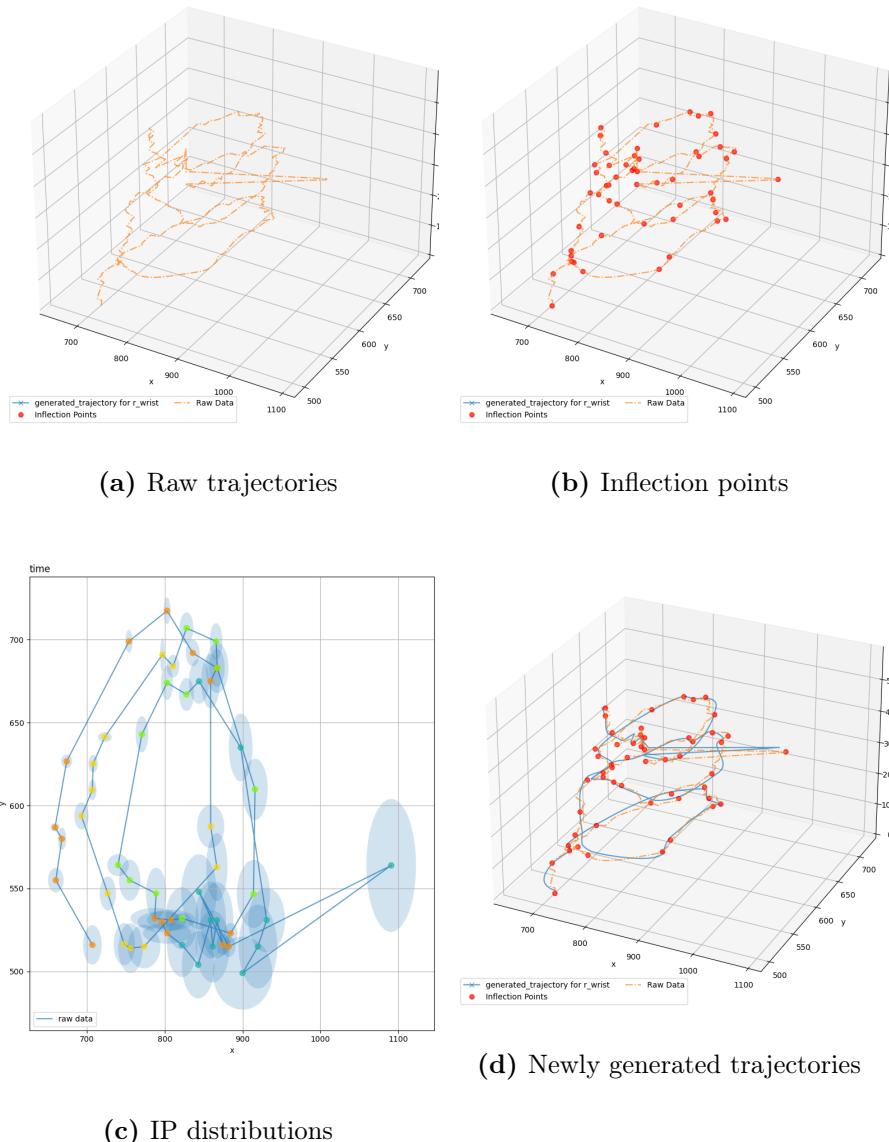


Figure 8.10: The generation process employed by POSH-G: finding inflection points in trajectories, sampling new points from a distribution around them, and fitting a spline to those newly sampled points.

8.4.2 Data Generation Algorithm

Once gesture videos have been processed and stored as trajectory data, the next step is to generate synthetic gestures based on the demonstrations in the PGR dataset. This process has five distinct parts: data preprocessing, calculating inflection points, filtering inflection points, sampling new inflection points, and generating new examples.

Data Preprocessing

Upon loading trajectory data, we search for gaps in the trajectory and attempt to fill them as a preprocessing step. This step is performed on a per-keypoint basis: each body part is treated individually. Some gaps are present in the original data, but we also introduce new gaps by rejecting points with a probability of less than 0.5 and eliminating points that have a distance from previous frames greater than 20 pixels. Once these gaps have all been identified, we interpolate the trajectory using the Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) method [261]. Any gaps larger than a configurable limit are simply left empty.

Calculating Inflection Points

Once body keypoint trajectories have been filtered and interpolated, the next step is selecting a set of representative points for the trajectory from which to reconstruct new ones. The inflection points of a trajectory, the points at which concavity changes, are a natural choice for this. We calculate inflection points using the second derivative method, by calculating the gradient of the keypoint trajectory, fitting a cubic spline to the gradient, and finding the roots of the polynomial which describes the spline. With the roots calculated, we can select the points of the real trajectory which correspond to them, selecting those points as our inflection points.

Filtering Inflection Points

After calculating inflection points, we filter them to ensure high quality for each point and remove unnecessary inflection points. This additional step is required due to the increased noisiness of our input compared to Cabrera *et al.*, . For simplicity, we assume that the first inflection point is high quality, and iterate over all subsequent inflection

points to create our final set. We drop inflection points that are determined to be simple linear midpoints between other inflection points, as long as they fall within a distance threshold. Additionally, any point that is too close to a inflection point that has been marked for retention in the final set is dropped. Midpoint inflection points that have sufficient distance between themselves and the points with which they form a line are retained, as the length of the trajectory requires more points to define it.

Sampling New Inflection Points

With our inflection points calculated, we have a representation of a keypoint's trajectory through time. To create new, synthetic data, we must create distributions around each inflection point from which to sample the points of a new trajectory. This is achieved in two steps. First, we cluster inflection points using DBSCAN [262] and estimate the variance of said clusters. Each inflection point receives the cluster to which it belongs. However, if an inflection point is not clustered using DBSCAN, we calculate a manual variance by setting the bounds of distribution around the inflection point to the limits of the preceding and following inflection points, then scaling the variance by a manual factor. Regardless of how the variance is estimated, once it has been estimated per inflection point, we can generate a new set of inflection points by sampling the normal distributions around each point defined by the inflection point and its calculated variance.

Generating New Gesture Examples

The final step in generating a new gesture example is producing a full trajectory from the newly sampled inflection points. We achieve this using the PCHIP interpolation algorithm, creating a new trajectory for body part in turn. These new trajectories preserve the concavity of the original trajectory, but pass through new inflection points, shifting and transforming them slightly to create a wide variety of synthetic gesture demonstrations.

8.4.3 Generation Parameters

For each gesture video, we process seven keypoints: the shoulder, elbow, and wrist of each arm, as well as the sternum, which is used during recognition to calculate relative coordinates. Each of these keypoints has its trajectory preprocessed at which time it is saved as the “real” trajectory. Following preprocessing, we calculate and filter inflection points, then generate one thousand synthetic trajectories, which are saved as the “generated” trajectories. Having generated training data, we can now move to the problem of gesture recognition.

8.5 Recognizing Protean Gestures

For the purpose of classifying Protean gestures, we train four different types of neural networks using hyperparameter optimization to select the best network configuration. In this section, we describe the data configurations, network types, and training methodology utilized.

8.5.1 Data Configurations

We collected data from four different pose filter configurations: no filter, a one €filter, a moving average filter, and a Kalman filter. Using all participants from the PGR dataset except Par-3, we generate data using two sets of generation parameters filter: one with a variance scaling factor of 0.25 and a maximum distance of 10 pixels between two adjacent points (for the distance aspect of the preprocessing step), and another with a variance scaling factor of 0.4 and a maximum distance of 20 pixels. We define the first set (with variance scaling 0.25 and a maximum gap of 10 pixels) as Set A and the second set (with variance scaling 0.4 and a maximum gap of 20 pixels) as set B. After the generation step of our process, we have eight pairs (four pose filters X two generation parameter sets) comprised of a real trajectory and 1,000 synthetic trajectories for each gesture video. Each combination of pose filter and generation parameters is trained and evaluated separately. When training neural networks on these configurations, we randomly select one gesture demonstration per gesture, per participant as the source for our generated training data. Of the remaining examples, one real trajectory is used

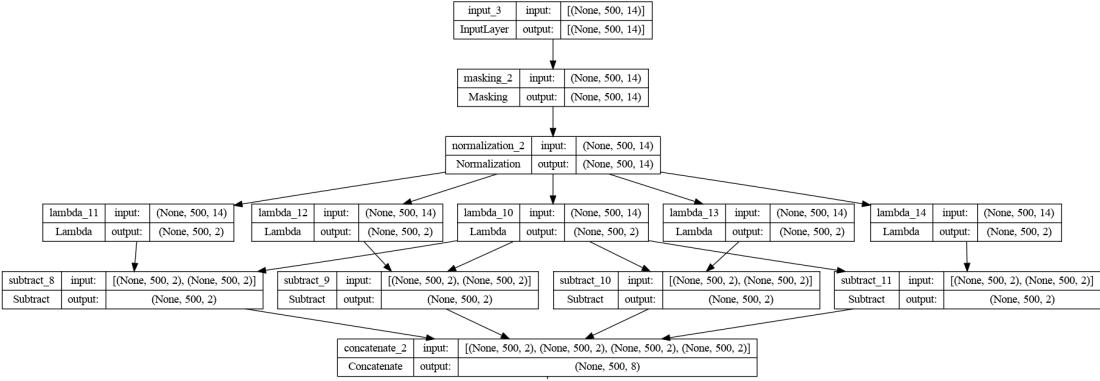


Figure 8.11: The preprocessing layers which are attached to the beginning of every network.

as validation data for hyperparameter search and the remaining trajectories are used as our test data. This training/validation/test split is consistent across all of the models trained for a specific pose filter and generation parameter combination.

8.5.2 Hyperparameter Search With KerasTuner

As we have no established network for one-shot gesture recognition underwater, we perform hyperparameter search using KerasTuner’s [263] Hyperband Tuner [264] to create dynamically configured neural networks. Each network can have a large number of its internal parameters and even structure modified by the Hyperband Tuner, which searches for good configurations by evaluating against the validation data previously mentioned. When KerasTuner selects a hyperparameter set, the network is trained for 10 epochs, evaluated against previous runs, and updates are made to the current best hyperparameters. Once a set of hyperparameters has been selected, the network is trained for 50 epochs, then retrained to the epoch which had the highest performance on validation data.

8.5.3 Data Preprocessing Layers

Every network used for POSH-G recognition has a common preprocessing module prior to its inputs. The input to the network is a 500 data point window of body keypoint trajectories for the shoulder, elbow, and wrist of each arm, as well as the sternum.

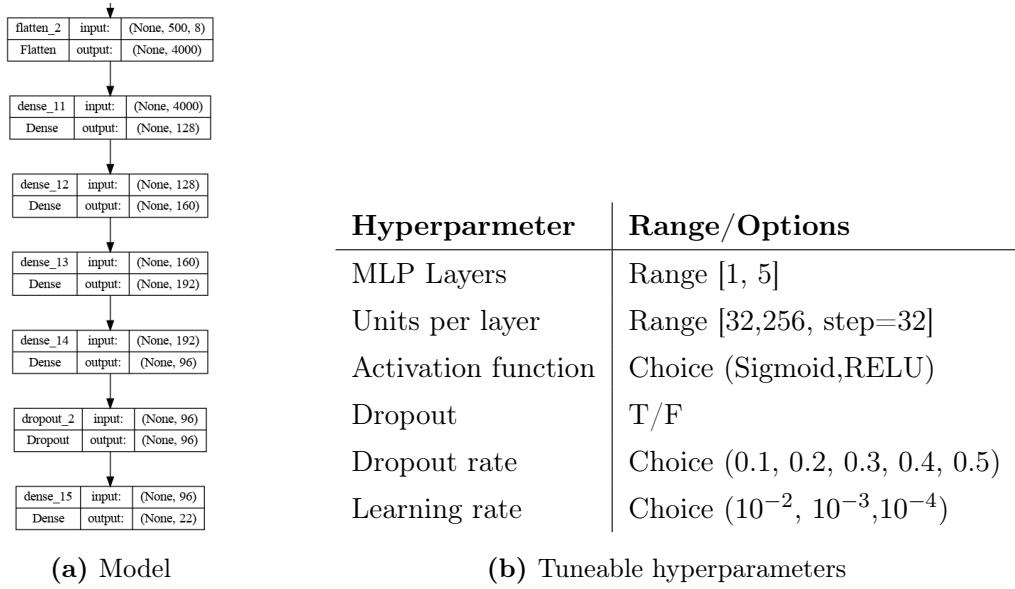


Figure 8.12: An example of the multi-layer perceptron networks trained for gesture recognition, with the hyperparameters that can be changed by KerasTuner.

The preprocessing layers make the coordinates of the arm keypoints relative to the sternum, then normalize them. While the overall input to the network is absolute coordinate values for seven keypoints, the actual recognition portion of each network receives normalize, sternum-relative values for the arm keypoints. The layers which perform these operations are visualized in Figure 8.11.

8.5.4 Neural Network Types Utilized

We train four different types of neural networks for gesture recognition: a multi-layer perceptron [265], a convolutional neural network [266], an LSTM [209], and a transformer-based network [267].

Multi-Layer Perceptron

The multi-layer perceptron is an early form of deeper neural networks, well suited to small classification problems. When tuning our MLPs, we vary the number of layers, the number of units within each layer, the activation function for the neurons, the presence

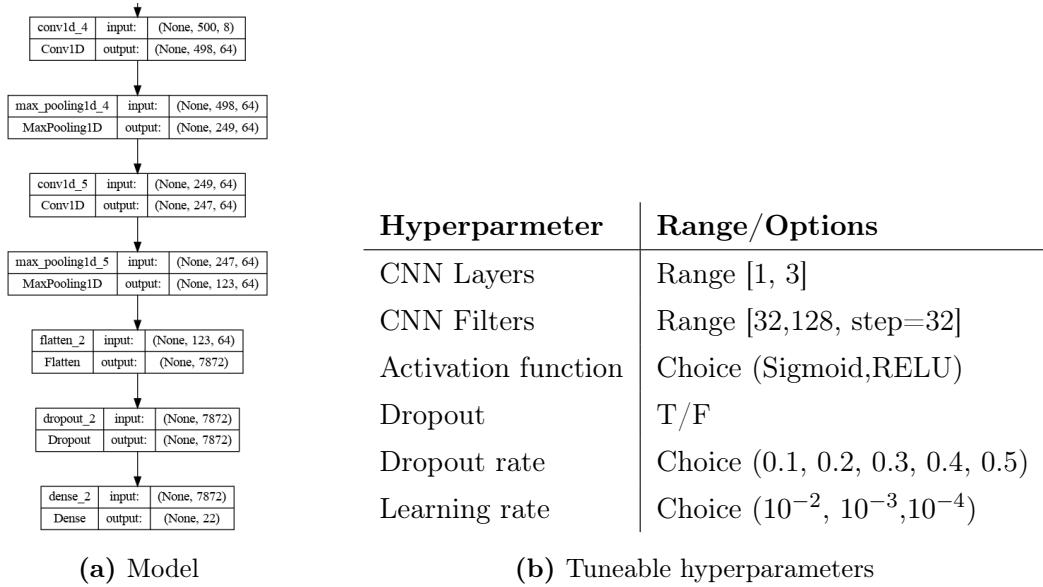
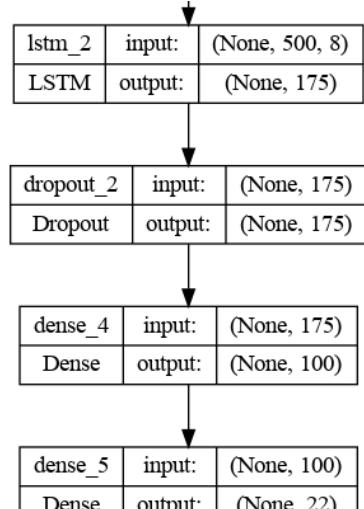


Figure 8.13: An example of the convolutional neural networks trained for gesture recognition, with the hyperparameters that can be changed by KerasTuner.

of a dropout layer, the dropout values, and the learning rate. An example of a tuned MLP structure along with the hyperparameters which can be tuned for it can be found in Figure 8.12.

Convolutional Neural Network

Two-dimensional convolution neural networks have demonstrated success in image classification and object detection tasks. One-dimensional CNNs, on the other hand, are often used for activity recognition and classification of motion data. When tuning our CNNs, we vary the number of layers, the number of filters, the activation function, the presence of a dropout layer, the dropout value, and the learning rate. An example of a tuned CNN structure along with the hyperparameters which can be tuned for it can be found in Figure 8.13.



(a) Model

Hyperparameter	Range/Options
LSTM Units	Range [25, 250, step=25]
Dropout rate	Range [0.1, 0.7, step=0.1]
Dense units	Range(25, 150, step=25)
Activation function	Choice (Sigmoid, RELU)
Learning rate	Choice (10^{-2} , 10^{-3} , 10^{-4})

(b) Tuneable hyperparameters

Figure 8.14: An example of the long short-term memory networks trained for gesture recognition, with the hyperparameters that can be changed by KerasTuner.

Long Short-Term Memory Network

Long short-term memory networks are a form of recurrent neural network frequently utilized on time-series data, which our gesture trajectories are. When tuning our LSTMs, we vary the number of units within the LSTM, the dropout value, the size of the densely connected layers after the LSTM, their activation function, and the learning rate. An example of a tuned LSTM structure along with the hyperparameters which can be tuned for it can be found in Figure 8.14.

Transformer Network

Transformers are a newer form of neural network which replaces the traditional convolution and recurrence of CNNs and RNNs with a self-attention mechanism. When tuning our transformers, we vary the number of heads, the head size, the filter dimension for the head, the dropout values, the configuration of the multilayer perception after the transformer, and the learning rate. An example of a tuned transformer structure along with the hyperparameters which can be tuned for it can be found in Figure 8.15.

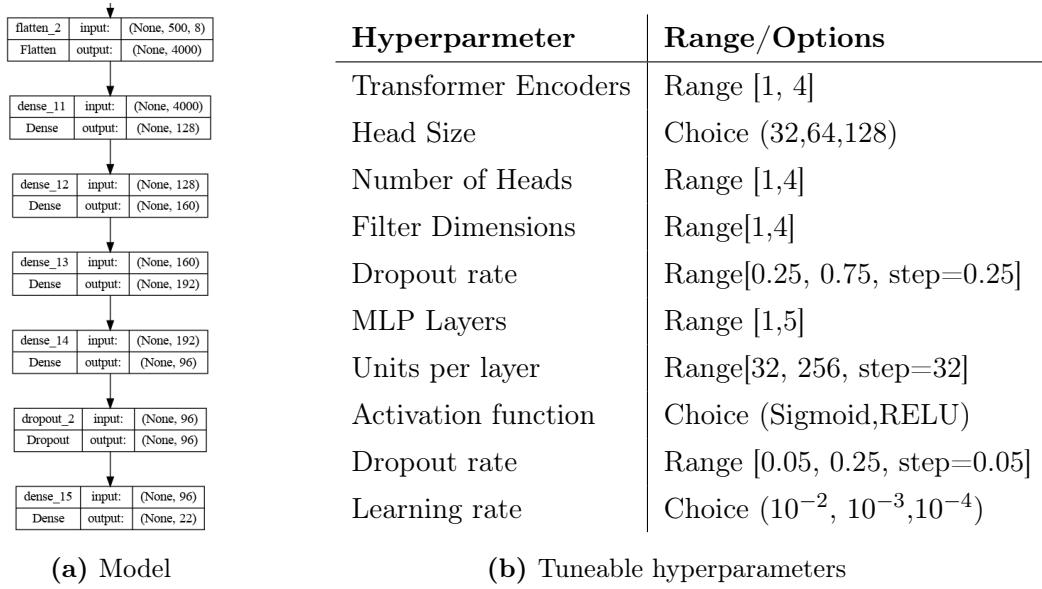


Figure 8.15: An example of the transformer networks trained for gesture recognition, with the hyperparameters that can be changed by KerasTuner.

8.6 Results: Accuracy of One-Shot-Trained Recognizers

Having trained MLP, CNN, LSTM, and Transformer networks on data from the eight generated datasets we created, we must evaluate the networks to determine the viability of the POSH-G system. We evaluate each model on the designated test data for its combination of generation parameters and pose filters, comparing against ground truth gesture labels from the PRG dataset. The overall results of these evaluations can be seen in Figure 8.16, with confusion matrices for the two top-performing models in Figure 8.17. In this section, we will discuss the results of our tuning, training, and evaluation process: which models perform best on our data, and how different generation parameters and pose filters affected model accuracy.

8.6.1 Overall Results

The accuracy of all models is significantly higher than the expected accuracy of a random guess out of 22 (4.5%), indicating that the data generation is producing datasets that can be learned. In fact, with some models reaching an accuracy over 50%, model accuracies

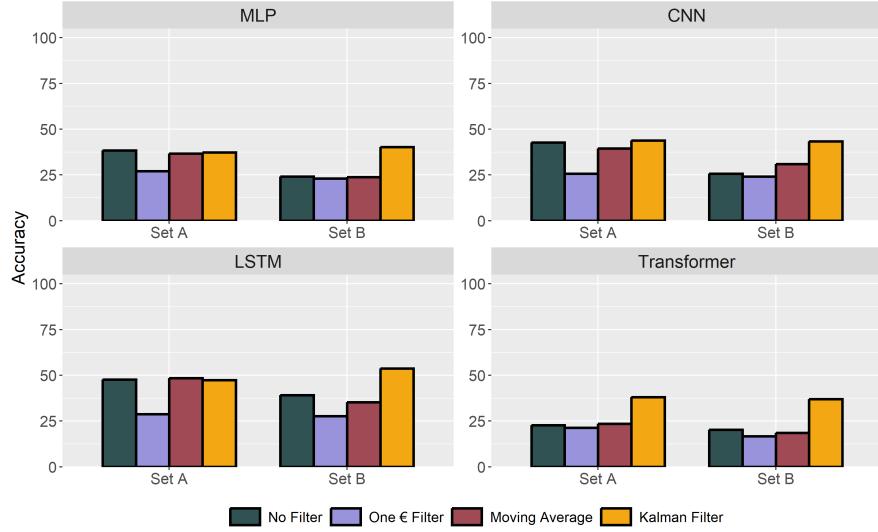
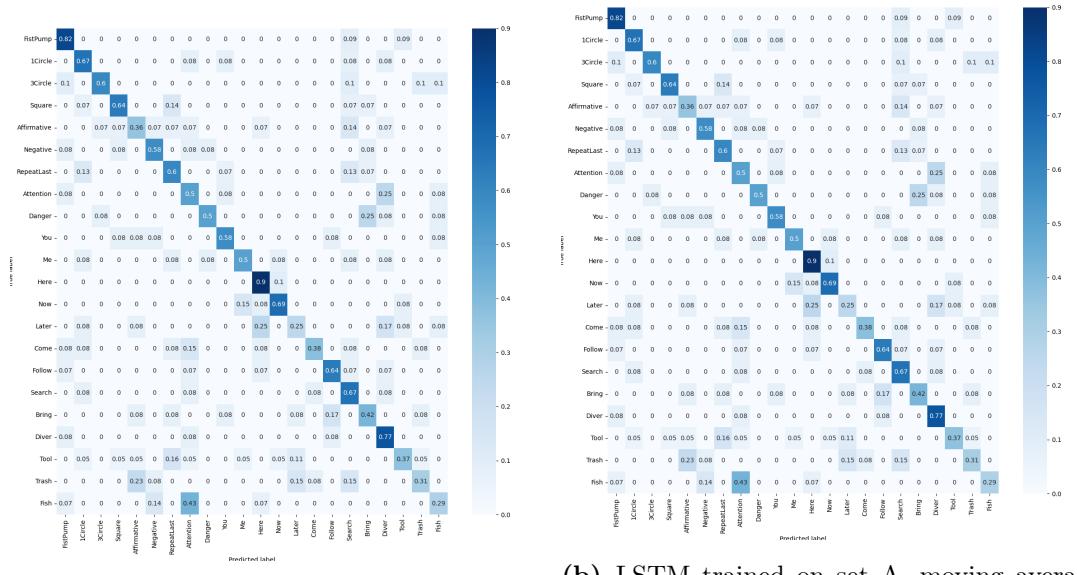


Figure 8.16: Recognition results for MLP, CNN, LSTM, and Transformer networks trained on Set A or B, with four pose filter options.

are somewhat higher than might be expected given that they are trained on synthetic data generated from an extremely small set of demonstrations. However, no models performed well enough to be viable for deployment in a real-world environment. Further development is necessary, to achieve a level of accuracy more reasonable for deployment ($\geq 80\%$).

8.6.2 Which Models Perform Best?

By far the best-performing models were the LSTMs, with some versions reaching a test accuracy of 53%. While there is some variation from the generation parameters and pose filters, no other models come close to reaching the accuracy of the LSTM-based networks. All of the other model types performed relatively similarly, with the transformer models performing the worst. This does not necessarily mean that these models are inappropriate for the task, it may be due to issues in data generation that these models have not achieved higher accuracy. Nonetheless, from the current information available, LSTMs appear to be the best option for training POSH-G recognizers.



(a) LSTM trained on set B, Kalman filter.

(b) LSTM trained on set A, moving average filter.

Figure 8.17: Confusion matrices from the two best-performing network configurations of the POSH-G recognition networks.

8.6.3 Effect of Generation Parameters

The generation parameters had a much less significant effect on recognition accuracy than the model type. Set B appears to reduce the accuracy of models, which may reflect the fact that the variance scaling factor is higher, leading to a more widely distributed set of generated trajectories. However, we have thus far only evaluated two sets of generation parameters. There are many other possible options for these parameters and other parameters or options during generation that have not been discussed. Now that it has been shown that the overall approach can work in underwater environments, further investigation into the generation process is necessary.

8.6.4 Effect of Pose Filters

Pose filters have a more pronounced effect on recognition results. The one ϵ filter unfortunately reduces recognition accuracy over using no filter in every case, providing a strong argument for not using this filter. On the other hand, the Kalman filter version of the dataset frequently outperforms the unfiltered version, with the highest accuracy model being trained on this version. This may not be definitive proof that this pose filter is worth implementing, however. There is a certain degree of stochasticity in the success of model training due to the random selection of the demonstration use to generate data. Replication of this result with different demonstration would be required to show that the Kalman filtered pose data improves recognition, but that fact that we see this pattern repeated in this evalaution is promising. Lastly, the moving average filter does not degrade performance significantly, neither does it improve it.

8.7 Conclusion and Future Directions

In this chapter, we presented Protean One-Shot Hand Gestures or POSH-G. We began by introducing a diver body pose estimation dataset (OceanPose-Training and OceanPose-Evaluation) and training pose estimation models using DeepLabCut on the said dataset. With an accurate pose estimation network created for divers, we next developed a synthetic gesture generation algorithm based on Cabrer *et al.*, , which allows us to generate thousands of variations on a gesture from a single demonstration. Lastly, we trained

four varieties of gesture recognition networks using a hyperparameter search method to determine the best network structures and parameters. The LSTM networks perform the best overall, reaching a recognition accuracy of 53%. While our results do not reach a level of success sufficient for POSH-G recognition methods to be deployed in the field, the three parts of this method all have significant room for improvement, and we believe that this method will eventually be suited for deployment. Regardless, this research has made significant advancements in the state of the art to underwater gesture recognition, bringing methods that had previously only been applied in terrestrial environments into the world of AUVs.

Possible Areas of Future Exploration

As in previous chapters, we provide a few concepts for future exploration along this line. Unlike previous chapters, however, our method has not yet achieved sufficient success to be field-deployable, so our future exploration suggestions are somewhat more mundane.

Improvement of Pose Estimation

The quality of the pose estimation inputs to the POSH-G method is of paramount importance. Spurious keypoints at the pose estimation level cause issues during data generation as well as during recognition. Improvement of pose estimation algorithms for divers is therefore a useful area of focus. The error level of our selected pose estimation network is quite low, however, there are still issues to be solved. Single-frame jumps in keypoints, misclassifications, and similar issues could likely be solved with further investigation of filtering methods.

Evaluation of Data Generation

Our data generation method has thus far demonstrated impressive results, generating thousands of feasible gesture examples from a single demonstration. However, we currently have no way of evaluating either the similarity or distinctiveness of generated gestures to determine the success of the generation process quantitatively.

Multi-Participant Gesture Analysis

Additionally, it is likely that there are improvements that can be made to our generation algorithm, primarily on the topic of improving the variety of generated gestures. One possible way to achieve this would be to analyze multiple participants' demonstrations of a gesture together, creating a representation and new gesture trajectories based off of all participants' demonstrations of a gesture, instead of one by one.

Improvement of Recognition

Our recognition methods thus far are relatively standard approaches but have not achieved sufficient accuracy for field deployment. While it is unclear where exactly the fault lies, it is entirely possible that the recognition algorithms themselves are the primary issue. To this end, it would be useful to explore other options for recognition algorithms, including traditional machine learning methods such as hidden Markov models, dynamic time warping, etc.

Part III: Underwater Human-Robot Interaction

Underwater human-robot interaction research is a field in its infancy. The platforms are still immature, few researchers have noted UHRI as a topic of interest, and the majority of research focuses on “first-order” questions of how to perceive and communicate with humans underwater. Studying such questions is of obvious value; indeed, research on underwater communication and perception comprises the first two parts of this thesis. However, for the field to progress, we must move past the foundational questions of communication and perception to begin exploring questions of interaction and collaboration and discover how the capabilities that the community creates can impact the larger picture of UHRI. In this final part of the thesis, we present two pieces of work in service of this goal. Chapter 9 presents PROTEUS-HRI, a modular, extensible system for underwater HRI using ROS which is intended as a foundation for future UHRI software. Then, in Chapter 10, we discuss a piece of “second-order” research: Autonomous Communication Vector Selection, an algorithm for adapting AUV-to-human communication methods to the context of an interaction. Our study of context-adaptive communication is a first for the underwater environment, and one of the first explorations of these second order questions, which all fundamentally return to the same question: how can we best enable interaction and collaboration among AUVs and divers? ACVS attempts to address one aspect of that question by adapting communication to its context in order to improve understanding and improve downstream impacts on task success.

Chapter 9

PROTEUS-HRI: Natural, Adaptive, Underwater HRI

Throughout this thesis, we have presented AUV-to-human communication methods, human perception methods, and a system for gestural control of an AUV. In a deployed co-AUV, these systems would have to work not only alongside each other, but with other yet-to-be-developed systems for communication, perception, and interaction. As the number and complexity of these interrelated systems grows, the need for a framework connecting them increases. The Robot Operating System (ROS), which is typically used for robotics programming provides important interprocess communication structures, but no such structures exist for defining UHRI languages and symbols, for sharing common diver information, and so on. To enable future research utilizing the communication and perception methods that we introduced in this thesis, we present a software framework for underwater HRI: PROTEUS. No specific research goal is evaluated in this chapter. Rather, this chapter serves to describe a system that enables UHRI research, including the ACVS system described in Chapter 10.

9.1 Purpose and Scope of PROTEUS

The name PROTEUS comes from an old man of Greek myth, a shapeshifter and soothsayer associated with the ocean. Proteus is often associated with the concepts of adaptability and change, hence the adjective *protean*, which the gesture language presented

in the preceding chapter is named for. In naming this UHRI software framework PROTEUS, we reflect our goals for the framework: adaptability, many-formed, and intelligent HRI in the ocean or any other body of what.

9.1.1 What Is PROTEUS?

PROTEUS is a ROS-based framework for underwater HRI, comprised of six parts:

- A communication language definition format encompassing communication vectors and symbols.
- Implementations of five communication vectors: RCVM (for Aqua and LoCO), HREyes, SIREN-TTS, SIREN-Tonal, and an OLED display.
- A diver context module that combines input from a diver detector and body pose estimator to provide a representation of recently seen divers for other algorithms to utilize.
- A set of ROS messages and services which service both the communication and perception modules.
- A set of object-oriented Python structures for representing communication symbols, to aid in implementing new communication vectors.
- An overall structure of software elements for integration into a full HRI control system.

Other components were planned for inclusion in PROTEUS, such as a system for mission planning, environmental context, atomic AUV behaviors, and more. The currently implemented portions of PROTEUS support multimodal AUV-to-DIVER communication and rich diver perception and are prepared for integration into the other planned portions of PROTEUS, if they are ever completed.

9.1.2 Why Not Just ROS?

Readers familiar with the Robot Operating System might reasonably ask what PROTEUS is good for, as ROS handles interprocess communication (IPC) for the AUV, and

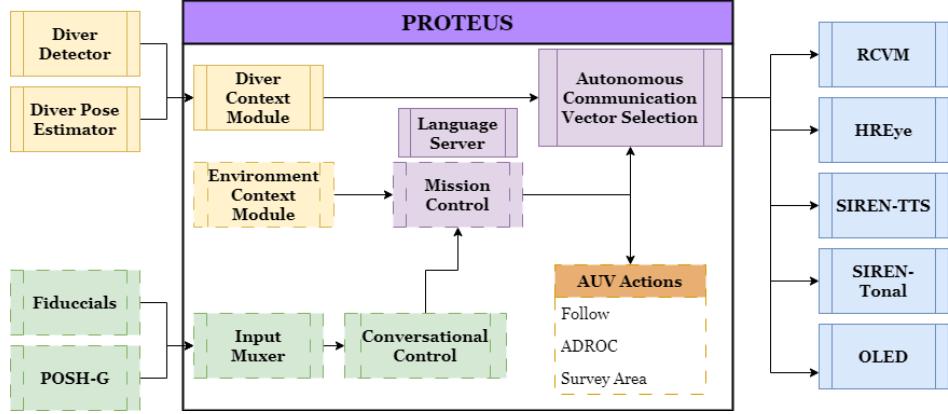


Figure 9.1: The PROTEUS UHRI system. Boxes drawn with dashed lines are not integrated or implemented.

users are intended to write modular software packages. For this exact reason, PROTEUS does not implement new IPC methods and is built within the ROS package system. PROTEUS exists due to lived experience developing UHRI software for AUVs. Over time, systems built for the same AUV begin to develop in ways that make them incompatible with one another. Without an intentional structure and process for organizing UHRI capabilities, packages written by successive generations of researchers will continue to grow like a plant that is insufficiently cared for. Entire branches of work will wither and die, simply because they do not work alongside or communicate with other methods. By creating a ROS-based framework for UHRI systems, we attempt to provide a start at bringing structure and well-managed growth to UHRI software.

9.1.3 Scope and Limitations

As mentioned in the previous section, PROTEUS does not implement new IPC methods. Additionally, we do not claim that PROTEUS can currently solve any particular UHRI task autonomously, or that it is bug-free. PROTEUS is a research software system, developed over the course of several years of spare research time. Its only current capability is providing multi-modal communication via ROS Action and rich diver perception via a ROS message. While we will briefly discuss the other planned aspects of PROTEUS's structure, they are not currently implemented and may be modified significantly if they are ever completed.

9.2 Language Definitions and Common Structures

First among the components of PROTEUS is an XML structure for defining communication languages, vectors, and symbols.

9.2.1 Language Definitions

Every PROTEUS language definition file (LDF) is an XML file that specifies the possible communication vectors and symbols for a robot's UHRI configuration. Languages are platform specific due to the platform-specific nature of communication vector implementations. The definition of a language is mostly platform agnostic, except for the vector endpoints. The PROTEUS language server processes language definitions and makes the majority of the information available on the ROS parameter server for other software to access. For instance, when a communication vector node starts, it can query the ROS parameter server for the information from the language definition on where its symbol definitions may be found. An example of a language definition follows below.

LoCO Diver-Signal Language Definition

```

1 <language>
2   <meta>
3     <name>PROTEUS Diver-Signal Based Language</name>
4     <robot>LoCO-AUV</robot>
5     <description>A PROTEUS language designed for testing PROTEUS on Aqua, with
6       symbols based on iterative coding of diver hand signal languages.</
7       description>
8     <directory>/home/michael/proteus_ws/src/proteus/language_definitions/loco_diver-signal
9       /</directory>
10    </meta>
11
12    <output>
13      <symbols>
14        <symbol name="Affirmative" id="affirmative" call_type="trigger" description="
15          A confirmation response indicating Yes, I will, etc." tags="
16          conversational,control"></symbol>
```

```

12   <symbol name="Negative" id="negative" call_type="trigger" description="A
      declining response indicating No, I will not, etc." tags="conversational
      ,control"></symbol>
13   <symbol name="Attention" id="attention" call_type="trigger" description="A
      command telling the interactant to pay attention" tags="conversational">
      </symbol>
14   <symbol name="Wait For Command" id="wait_for_command" call_type="trigger"
      description="A question asking the interactant for input of some kind."
      tags="conversational,question"></symbol>
15   <symbol name="Follow Me" id="follow_me" call_type="trigger" description="A
      command requesting the interactant to follow the AUV" tags="commands"></
      symbol>
16   <symbol name="Follow You" id="follow_you" call_type="trigger" description="An
      informational phrase telling the interactant that the AUV will follow
      them" tags="information"></symbol>
17   <symbol name="Come To Me" id="come" call_type="trigger" description="A
      command telling the interactnat to come to the AUV" tags="commands"></
      symbol>
18   <symbol name="Danger" id="danger" call_type="trigger" description="An
      informational phrase telling interactants that there is danger nearby."
      tags="information"></symbol>
19   <symbol name="Malfunction" id="malfunction" call_type="trigger" description=""
      An informational phrase informing interactants of an internal
      malfunction." tags="information"></symbol>
20   <symbol name="Go To Direction" id="go_direction" call_type="directional"
      description="A response requesting the interactant to move in a
      direction" tags="commands"></symbol>
21   <symbol name="Stay" id="stay" call_type="trigger" description="A command
      telling the interactant to stay where they are." tags="commands"></
      symbol>
22   <symbol name="Which Way" id="which_way" call_type="trigger" description="A
      question asking the interactant which way to go." tags="question"></
      symbol>
23   <symbol name="Remaining Battery" id="battery_remaining" call_type="quantity"
      description="An informational phrase telling the interactant how much
      battery the AUV has." tags="information"></symbol>
24   </symbols>
25
26   <vectors>
```

```

27     <vector name="DigitalDisplay" type="explicit_communication" pkg=""
28         proteus_loco_oled" file="loco_diver-signal_digitaldisplay.sdf"></vector>
29     <vector name="TTSSiren" type="explicit_communication" pkg="proteus_siren"
30         file="loco_diver-signal_tts-soneme.sdf"></vector>
31     <vector name="TonalSiren" type="explicit_communication" pkg="proteus_siren"
32         file="loco_diver-signal_tone-soneme.sdf"></vector>
33     <vector name="ActiveHREye" type="explicit_communication" pkg="proteus_hreye"
34         file="loco_diver-signal_active-luceme.sdf"></vector>
35   </vectors>
36
37 </language>
```

Communication Vectors

The vectors defined in a PROTEUS language represent the different ways that an AUV can communicate. The example above contains only output vectors, but input vectors are possible. A vector definition contains the name and the type of the vector, as well as listing the appropriate ROS package and symbol definition file (SDF).

Communication Symbols

Communication symbols in a PROTEUS language definition represent vector-agnostic communication symbols. Any vector may choose to implement a symbol or not, though it is recommended that vectors implement every symbol which makes sense for the medium. The “Affirmative” symbol, for example, is implemented as $K_{Affirmative}$, $L_{Affirmative}$, and $S_{Affirmative}$. These implementations are managed by the specific vector’s package, but each package reads symbol definition files to create the motion, light, sound, or digital display that provides the symbol to that vector. Each symbol definition in the LDF provides a display name, a description, a unique id string, a list of content tags, and a call type. Call types indicate what type of input is required for the symbol: “Trigger” call types require no input, “Directional” symbols take a ROS *geometry_msgs/Transform* message as input, and “Quantity” symbols require a floating point value which is interpreted as a percentage.

9.2.2 Vector-Specific Symbol Definitions

In order to implement a symbol, the various packages which provide communication vectors read symbol definition files, which provide a platform-agnostic definition of a symbol for that vector. For demonstration, a portion of the SDF for the DigitalDisplay vector above is shown below. Note the differing text data for the symbols with directional and quantity call types.

Digital Display Symbol Definition

```

1 <display_phrases>
2   <display_phrase name="Affirmative" id="affirmative">
3     <dnode step="0">
4       <text data="Affirmative"></text>
5       <duration seconds="10"></duration>
6     </dnode>
7   </display_phrase>
8   <display_phrase name="Negative" id="negative">
9     <dnode step="0">
10    <text data="Negative"></text>
11    <duration seconds="10"></duration>
12   </dnode>
13 </display_phrase>
14 <display_phrase name="Go To Direction" id="go_direction">
15   <dnode step="0">
16     <text data="Go {}"></text>
17     <duration seconds="10"></duration>
18   </dnode>
19 </display_phrase>
20 <display_phrase name="Remaining Battery" id="battery_remaining">
21   <dnode step="0">
22     <text data="{:.0f}% battery remaining"></text>
23     <duration seconds="10"></duration>
24   </dnode>
25 </display_phrase>
26 </display_phrases>
```

9.2.3 Common Structures

The task of processing SDFs and turning them into a proper implementation of each symbol is a difficult one. To aid in this process and provide some consistency between vector implementations, PROTEUS also contains a Python library of classes that represent quantities in language and symbol definition files. Symbols, vectors, and languages are represented, as are kinemes, lucemes, sonemes, digital displays, and all of the quantities which define them. In general, any XML tag in a LDF/SDF has an associated Python class which contains its own XML parsing code as well as any expected functions required for its use. These classes, along with the language server, are contained in the *proteus* package, which is intended as a dependency for nearly every PROTEUS-compliant package, along with *proteus_msgs*, which provides ROS messages, services, and actions.

9.3 Implementation of Communication Methods

The second component of PROTEUS is the implementation of five communication vectors: RCVM (Chapter 2), HREye (Chapter 3), a digital display, SIREN-TTS, and SIREN-Tonal (Chapter 4). Descriptions of each implementation follow below, with excerpts from their symbol definition files.

9.3.1 Digital Display

The Digital display vector's SDF was shown earlier as an example of symbol definition files in general. The implementation is straightforward, phrases are simply displayed on the OLED for the duration specified in the SDF. While the library used to write text to the display would likely change between different AUVs, the majority of the implementation would remain the same.

9.3.2 RCVM

RCVM implementations are extremely unique to their platform, due to the wide variety of motion characteristics that AUVs have. One of the benefits of using SDFs to specify symbols such as kinemes is that it eases the pain of translation between platforms.

For instance, the Aqua AUV can control roll, pitch, and yaw, while LoCO can only actuate on yaw and pitch independently. When transferring kinemes from Aqua to LoCO then, any kinemes which only use pitch or yaw can simply be copied. Unfortunately, there is no existing method for translating motion across axes yet, so kinemes that use roll must be rewritten for Aqua. Beyond that issue, kineme definitions are relatively simple. Most kinemes are simply a sequence of orientation or position changes with a duration. The exact way that these motions are executed depends on the platform-specific implementation. An excerpt of the LoCO RCVM symbol definition follows below.

LoCO RCVM SDF

```

1 <kinemes>
2   <kineme name="Affirmative" id="affirmative">
3     <knode_abs step="0" description="Upswing 1">
4       <orientation roll="0" pitch="15" yaw="0"></orientation>
5       <velocity surge="0" sway="0" heave="0"></velocity>
6       <duration seconds="2"></duration>
7     </knode_abs>
8     <knode_abs step="1" description="Downswing 1">
9       <orientation roll="0" pitch="-30" yaw="0"></orientation>
10      <velocity surge="0" sway="0" heave="0"></velocity>
11      <duration seconds="2"></duration>
12    </knode_abs>
13    <knode_abs step="3" description="Upswing 2">
14      <orientation roll="0" pitch="30" yaw="0"></orientation>
15      <velocity surge="0" sway="0" heave="0"></velocity>
16      <duration seconds="2"></duration>
17    </knode_abs>
18    <knode_abs step="4" description="Downswing 2">
19      <orientation roll="0" pitch="-15" yaw="0"></orientation>
20      <velocity surge="0" sway="0" heave="0"></velocity>
21      <duration seconds="2"></duration>
22    </knode_abs>
23  </kineme>
24  <kineme name="Negative" id="negative">
25    <knode_abs step="0" description="Right swing 1">
```

```

26     <orientation roll="0" pitch="0" yaw="15"></orientation>
27     <velocity surge="0" sway="0" heave="0"></velocity>
28     <duration seconds="2"></duration>
29   </knode_abs>
30   <knode_abs step="1" description="Left Swing 1">
31     <orientation roll="0" pitch="0" yaw="-30"></orientation>
32     <velocity surge="0" sway="0" heave="0"></velocity>
33     <duration seconds="2"></duration>
34   </knode_abs>
35   <knode_abs step="2" description="Right Swing 2">
36     <orientation roll="0" pitch="0" yaw="30"></orientation>
37     <velocity surge="0" sway="0" heave="0"></velocity>
38     <duration seconds="2"></duration>
39   </knode_abs>
40   <knode_abs step="3" description="Left Swing 2">
41     <orientation roll="0" pitch="0" yaw="-15"></orientation>
42     <velocity surge="0" sway="0" heave="0"></velocity>
43     <duration seconds="2"></duration>
44   </knode_abs>
45 </kineme>
46
47 <kineme name="Go To Direction" id="go_direction">
48   <knode_dir step="0" description="Go To target direction">
49     <duration seconds="2"></duration>
50   </knode_dir>
51   <knode_abs step="1" description="">
52     <orientation roll="0" pitch="0" yaw="0"></orientation>
53     <velocity surge="0.2" sway="0" heave="0"></velocity>
54     <duration seconds="1"></duration>
55   </knode_abs>
56   <knode_abs step="2" description="">
57     <orientation roll="0" pitch="0" yaw="0"></orientation>
58     <velocity surge="-0.2" sway="0" heave="0"></velocity>
59     <duration seconds="2"></duration>
60   </knode_abs>
61   <knode_abs step="3" description="">
62     <orientation roll="0" pitch="0" yaw="0"></orientation>
63     <velocity surge="0.2" sway="0" heave="0"></velocity>
64     <duration seconds="2"></duration>

```

```

65   </knode_abs>
66   <knode_abs step="4" description="">
67     <orientation roll="0" pitch="0" yaw="0"></orientation>
68     <velocity surge="0.2" sway="0" heave="0"></velocity>
69     <duration seconds="1"></duration>
70   </knode_abs>
71 </kineme>
72 <kineme name="Remaining Battery" id="battery_remaining">
73   <knode_abs step="0" description="">
74     <orientation roll="0" pitch="-90" yaw="0"></orientation>
75     <velocity surge="0.0" sway="0" heave="0"></velocity>
76     <duration seconds="2"></duration>
77   </knode_abs>
78   <knode_abs step="1" description="">
79     <orientation roll="0" pitch="180" yaw="0"></orientation>
80     <velocity surge="0.0" sway="0" heave="0"></velocity>
81     <duration seconds="2"></duration>
82   </knode_abs>
83   <knode_abs step="2" description="">
84     <orientation roll="0" pitch="-180" yaw="0"></orientation>
85     <velocity surge="0.0" sway="0" heave="0"></velocity>
86     <duration seconds="2"></duration>
87   </knode_abs>
88   <knode_abs step="3" description="">
89     <orientation roll="0" pitch="0" yaw="10"></orientation>
90     <velocity surge="0.0" sway="0" heave="0"></velocity>
91     <duration seconds="0.5"></duration>
92   </knode_abs>
93   <knode_abs step="4" description="">
94     <orientation roll="0" pitch="0" yaw="-10"></orientation>
95     <velocity surge="0.0" sway="0" heave="0"></velocity>
96     <duration seconds="0.5"></duration>
97   </knode_abs>
98   <knode_quant step="5" description="Go up an amount equivalent to the battery
99     remaining">
100    <quantity display_on="pitch" max="180" min="0"></quantity>
101    <duration seconds="2"></duration>
102   </knode_quant>
<knode_abs step="6" description="">
```

```

103      <orientation roll="0" pitch="0" yaw="10"></orientation>
104      <velocity surge="0.0" sway="0" heave="0"></velocity>
105      <duration seconds="0.5"></duration>
106    </knode_abs>
107    <knode_abs step="7" description="">
108      <orientation roll="0" pitch="0" yaw="-10"></orientation>
109      <velocity surge="0.0" sway="0" heave="0"></velocity>
110      <duration seconds="0.5"></duration>
111    </knode_abs>
112  </kineme>
113 </kinemes>
```

9.3.3 HREye

The implementation of HREye requires the largest amount of meta-configuration in the SDF of any of the vectors. HREye SDFs contain information about the number of light rings in use, configurations for logical sectors of lights, predefined colors, and a defined default state for the HREye rings. This amount of configuration was necessary to allow the creation of arbitrary HREye lucemes without specifying the illumination state of each of the individual LEDs. While HREye devices are not necessarily well-suited for integration into some AUVs, the implementation and definition of lucemes should translate seamlessly to new AUVs with little issue. An excerpt of the HREye symbol definition file follows below.

HREye Active Lucemes SDF

```

1 <hreye_definition>
2   <hreye_config number="2" mode="mirror" rate="10">
3     <!-- The rings meta information defines the LED indexing, including the total
        number of LEDS, the number of rings,
4       whether they are clockwise or counter clockwise, and their cardinal indexes. If
        your rings' indexing/directionality
5       is different, or cardinal indexes are different, you can easily change those
        here.-->
6   <rings num_rings="2" total_leds="40">
```

```

7      <ring id="outer_ring" start="0" end="23" dir="clockwise" top="0" right="6"
8          bot="12" left="18"></ring>
9      <ring id="inner_ring" start="24" end="39" dir="counter_clockwise" top="24"
10         right="28" bot="32" left="36"></ring>
11  </rings>
12
13  <!__ The sectors meta information provides aliases for portions of the device.
14      Instead of having to specify indexes or
15      even cardinal points, you can specify sector assignments for lnodes.__>
16  <sectors>
17      <sector id="all">
18          <sector_segment ring="outer_ring" start="start" end="end"></
19              sector_segment>
20          <sector_segment ring="inner_ring" start="start" end="end"></
21              sector_segment>
22      </sector>
23      <sector id="outer">
24          <sector_segment ring="outer_ring" start="start" end="end"></
25              sector_segment>
26      </sector>
27      <sector id="inner">
28          <sector_segment ring="inner_ring" start="start" end="end"></
29              sector_segment>
30      </sector>
31      <sector id="directional">
32          <center_relative_sector_segment ring="outer_ring" center="roll" width="12
33              "></center_relative_sector_segment>
34          <center_relative_sector_segment ring="inner_ring" center="roll" width="8"
35              "></center_relative_sector_segment>
36      </sector>
37  </sectors>
38
39  <!__ The colors meta information provides a set of colors for use on the device.
40  These color definitons are drawn directly from the original HREye code.__>
41  <colors>
42      <color id="none" r="0" g="0" b="0"></color>
43      <color id="green" r="0" g="200" b="0"></color>
44      <color id="cyan" r="0" g="175" b="175"></color>
45      <color id="blue" r="0" g="0" b="140"></color>

```

```

37      <color id="white" r="250" g="250" b="250"></color>
38      <color id="red" r="200" g="0" b="0"></color>
39      <color id="purple" r="128" g="0" b="128"></color>
40      <color id="orange" r="255" g="165" b="0"></color>
41      <color id="yellow" r="255" g="255" b="0"></color>
42  </colors>
43
44  <!-- The default state meta information provides a luceme_like definition which
     the HREyes should be set to after any luceme call in this mode.-->
45  <default_state>
46      <lnode_static step="0" sector="all" description="default state">
47          <illumination color="none" brightness="1.0"></illumination>
48      </lnode_static>
49  </default_state>
50 </hreye_config>
51
52 <lucemes>
53     <luceme name="Affirmative" id="affirmative">
54         <lnode_static step="0" sector="outer" description="Outer ring cyan">
55             <illumination color="cyan" brightness="1.0"></illumination>
56             <duration seconds="6"></duration>
57         </lnode_static>
58         <lnode_blink step="0" sector="inner" description="Inner ring blinks green">
59             <illumination id="on_state" color="green" brightness="1.0"></illumination>
60                 >
61             <illumination id="off_state" color="none" brightness="1.0"></illumination>
62                 >
63             <blink period="1" iterations="6"></blink>
64         </lnode_blink>
65     </luceme>
66     <luceme name="Go To Direction" id="go_direction">
67         <lnode_blink step="0" sector="directional" description="Blink sector around
68             angle">
69             <illumination id="on_state" color="yellow" brightness="1.0"></
70                 illumination>
71             <illumination id="off_state" color="none" brightness="1.0"></illumination>
72                 >
73             <blink period="2" iterations="3"></blink>
74         </lnode_blink>

```

```

70   </luceme>
71   <luceme name="Remaining Battery" id="battery_remaining">
72     <lnode_fill step="0" sector="outer" description="Outer ring fills to value.">
73       <illumination id="low" color="red" brightness="1.0"></illumination>
74       <illumination id="mid" color="yellow" brightness="1.0"></illumination>
75       >
76       <illumination id="high" color="green" brightness="1.0"></illumination>
77       >
78     <fill type="expand_value" direction="clockwise" start="top" range="0:100"></fill>
79     <color_map target="value" mapping="low=0:20;mid=21:50;high=51:100"></color_map>
80     <duration seconds="3"></duration>
81   </lnode_fill>
82   <lnode_fill step="1" sector="outer" description="Outer ring reverses its fill">
83     <illumination id="low" color="red" brightness="1.0"></illumination>
84     <illumination id="mid" color="yellow" brightness="1.0"></illumination>
85     >
86     <illumination id="high" color="green" brightness="1.0"></illumination>
87     >
88     <fill type="contract_value" direction="counter_clockwise" start="value" range="0:100"></fill>
89     <color_map target="value" mapping="low=0:20;mid=21:50;high=51:100"></color_map>
90     <duration seconds="3"></duration>
91   </lnode_fill>
92   </luceme>
93 </lucemes>
94 </hreye_definition>

```

9.3.4 SIREN-TTS

The SIREN-TTS implementation is quite straightforward, with configuration related to speech synthesis parameters in the header of the SDF, and sonemes describing simple speech content in the remaining body of the file. The cross-platform translation is straightforward as well, with the only concern being integrating a sound production

device into the AUV. As long as the AUV can produce sound, the SIREN-TTS implementation should work seamlessly. An excerpt of the SIREN-TTS symbol definition file follows below.

SIREN-TTS SDF

```

1 <siren_definition>
2   <siren_config>
3     <clips pkg="proteus_siren" directory="audio_files" volume="1.0"></clips>
4     <voice lang="us" id="2" wpm="150" volume="5"></voice>
5     <dynamic_input topic=" tts_input " type="String"></dynamic_input>
6   </siren_config>
7   <sonemes>
8     <soneme name="Affirmative" id="affirmative">
9       <snode_speech step="0" description="">
10      <speech volume="1.0" speed="1" text="Yes"></speech>
11    </snode_speech>
12  </soneme>
13  <soneme name="Negative" id="negative">
14    <snode_speech step="0" description="">
15      <speech volume="1.0" speed="1" text="No"></speech>
16    </snode_speech>
17  </soneme>
18  <soneme name="Go To Direction" id="go_direction">
19    <snode_speech step="0" description="">
20      <variable_speech volume="1.0" speed="1" text="Go {}"></variable_speech>
21    </snode_speech>
22  </soneme>
23  </soneme>
24  <soneme name="Remaining Battery" id="battery_remaining">
25    <snode_speech step="0" description="">
26      <variable_speech volume="1.0" speed="1" text="{} battery remaining"></
27        variable_speech>
28    </snode_speech>
29  </soneme>
30 </sonemes>
31 </siren_definition>
```

9.3.5 SIREN-Tonal

Lastly, the implementation of SIREN-TONAL is somewhat more complex. The header of the file defines a number of synth voices using waveform, vibrato, attack, and decay values. These tracks are given unique id strings which enable their use in sonemes. The sonemes are defined using tones with defined durations, with either static tones, variable tones, or tones that sweep from one note to another. Similar to SIREN-TTS, the adaption of this implementation across AUV platforms is simple, any AUV that can produce sound can use SIREN-Tonal. An excerpt of the SIREN-Tonal symbol definition file follows below.

SIREN-Tonal SDF

```

1 <?xml version="1.0" encoding="UTF_8"?>
2 <siren_definition>
3   <siren_config>
4     <clips pkg="proteus_siren" directory="audio_files" volume="1.0"></clips>
5     <synth_track id="nintendo" wave="square" vibrato="0" variance="0" attack="0.001"
6       decay="0.01"></synth_track>
7     <synth_track id="atari" wave="triangle" vibrato="0" variance="0" attack="0.01"
8       decay="0.025"></synth_track>
9     <synth_track id="atari_fade" wave="triangle" vibrato="0" variance="0" attack="0.01"
10      decay="3"></synth_track>
11    <synth_track id="nintendo_fade" wave="square" vibrato="0" variance="0" attack="0.01"
12      decay="3"></synth_track>
13    <synth_track id="fear" wave="square" vibrato="5" variance="10" attack="0.001"
14      decay="0.01"></synth_track>
15    <synth_track id="woozy" wave="square" vibrato="2" variance="25" attack="0.001"
16      decay="0.01"></synth_track>
17   </siren_config>
18   <sonemes>
19     <soneme name="Affirmative" id="affirmative">
20       <snode_tone step="0" description="">
21         <tone track="nintendo" note="c3"></tone>
22         <duration seconds="0.1"></duration>
23       </snode_tone>
24       <snode_tone step="1" description="">
25         <duration seconds="0.1"></duration>

```

```

20      </snode_tone>
21      <snode_tone step="2" description="">
22          <tone track="nintendo" note="e3"></tone>
23          <duration seconds="0.1"></duration>
24      </snode_tone>
25      <snode_tone step="3" description="">
26          <duration seconds="0.1"></duration>
27      </snode_tone>
28      <snode_tone step="4" description="">
29          <tone track="nintendo" note="g3"></tone>
30          <duration seconds="1"></duration>
31      </snode_tone>
32  </soneme>
33  <soneme name="Negative" id="negative">
34      <snode_tone step="0" description="">
35          <tone track="nintendo" note="g3"></tone>
36          <duration seconds="0.1"></duration>
37      </snode_tone>
38      <snode_tone step="1" description="">
39          <duration seconds="0.1"></duration>
40      </snode_tone>
41      <snode_tone step="2" description="">
42          <tone track="nintendo" note="eb3"></tone>
43          <duration seconds="0.1"></duration>
44      </snode_tone>
45      <snode_tone step="3" description="">
46          <duration seconds="0.1"></duration>
47      </snode_tone>
48      <snode_tone step="4" description="">
49          <tone track="nintendo" note="c3"></tone>
50          <duration seconds="1"></duration>
51      </snode_tone>
52  </soneme>
53  <soneme name="Go To Direction" id="go_direction">
54      <snode_tone step="0" description="">
55          <tone track="atari" note="c4"/>
56          <duration seconds="0.1"/>
57      </snode_tone>
58      <snode_tone step="1" description="">

```

```

59      <variable_tone track="atari" param='y_val' options="b3;c4;d4"/>
60      <duration seconds="0.1"/>
61  </snode_tone>
62  <snode_tone step="2" description="">
63      <variable_tone track="atari" param='y_val' options="a3;c4;e4"/>
64      <duration seconds="0.1"/>
65  </snode_tone>
66  <snode_tone step="3" description="">
67      <variable_tone track="atari" param='y_val' options="g3;c4;f4"/>
68      <duration seconds="0.1"/>
69  </snode_tone>
70  <snode_tone step="4" description="">
71      <tone track="atari" note="c4"/>
72      <duration seconds="0.1"/>
73  </snode_tone>
74  <snode_tone step="5" description="">
75      <variable_tone track="atari" param='y_val' options="b3;c4;d4"/>
76      <duration seconds="0.1"/>
77  </snode_tone>
78  <snode_tone step="6" description="">
79      <variable_tone track="atari" param='y_val' options="a3;c4;e4"/>
80      <duration seconds="0.1"/>
81  </snode_tone>
82  <snode_tone step="7" description="">
83      <variable_tone track="atari" param='y_val' options="g3;c4;f4"/>
84      <duration seconds="0.1"/>
85  </snode_tone>
86  <snode_tone step="8" description="">
87      <duration seconds="1.0"/>
88  </snode_tone>
89  <snode_tone step="0" description="">
90      <tone track="atari" note="g4"/>
91      <duration seconds="0.1"/>
92  </snode_tone>
93  <snode_tone step="1" description="">
94      <variable_tone track="atari" param='z_val' options="f4;g4;a4"/>
95      <duration seconds="0.1"/>
96  </snode_tone>
97  <snode_tone step="2" description="">
```

```

98      <variable_tone track="atari" param='z_val' options="e4;g4;b4"/>
99      <duration seconds="0.1"/>
100     </snode_tone>
101     <snode_tone step="3" description="">
102       <variable_tone track="atari" param='z_val' options="d4;g4;c5"/>
103       <duration seconds="0.1"/>
104     </snode_tone>
105     <snode_tone step="4" description="">
106       <tone track="atari" note="g4"/>
107       <duration seconds="0.1"/>
108     </snode_tone>
109     <snode_tone step="5" description="">
110       <variable_tone track="atari" param='z_val' options="f4;g4;a4"/>
111       <duration seconds="0.1"/>
112     </snode_tone>
113     <snode_tone step="6" description="">
114       <variable_tone track="atari" param='z_val' options="e4;g4;b4"/>
115       <duration seconds="0.1"/>
116     </snode_tone>
117     <snode_tone step="7" description="">
118       <variable_tone track="atari" param='z_val' options="d4;g4;c5"/>
119       <duration seconds="0.1"/>
120     </snode_tone>
121   </soneme>
122   <soneme name="Remaining Battery" id="battery_remaining">
123     <snode_tone step="0" description="">
124       <tone track="nintendo" note="c3"/>
125       <duration seconds="0.75"/>
126     </snode_tone>
127     <snode_tone step="1" description="">
128       <duration seconds="0.25"/>
129     </snode_tone>
130     <snode_tone step="2" description="">
131       <tone track="nintendo" note="c4"/>
132       <duration seconds="0.75"/>
133     </snode_tone>
134     <snode_tone step="3" description="">
135       <duration seconds="1"/>
136     </snode_tone>

```

```

137     <snode_tone step="4" description="">
138         <run_tone track="nintendo" start_note="c3" stop_note="c4" glissando="True
139             "/>
140         <quantity min="0.0" max="1.0" parameter="quantity" display_on="run_tone"/
141             >
142         <duration seconds="2"/>
143     </snode_tone>
144 </soneme>
145 </sonemes>
146 </siren_definition>

```

9.4 Implementation of Diver Context Module

Diver perception is an important capability for collaborative AUVs, one we have discussed extensively in this thesis. From the diver detection methods introduced in Chapter 5 to the diver-relative pose estimation used for ADROC in Chapter 7, processing images to extract information about a diver is a key capability for AUVs. The third component of PROTEUS is a diver context module (DCM), a package that processes bounding box detections of divers and associates them across frames using a spatiotemporal matching algorithm. The module then runs on-demand pose estimation using DeepLabCut-Live [258] and calculates a modified version of psuedodistance. The DCM then publishes a list of all recently seen divers, with the bounding box, pose, and diver-relative position information, available for any other ROS system to use. This system will be particularly useful in the following chapter, where it forms the basis for our adaptive communication method.

9.4.1 Spatial Identity Assignment

The diver context module core functionality revolves around an internal list of diver tracks, which represent observations of a diver across time. The primary input for these tracks is bounding boxes from a diver detector. As in previous chapters, we utilize a diver detector running a YoloV4-Tiny model trained on VDD- \bar{C} (Chapter 5) as our source of diver bounding boxes. Once the DCM has received a list of bounding boxes

from the diver detector, it must determine if they belong to previously seen divers, or if they are new to the scene. This is achieved using a spatiotemporal matching algorithm. Diver identification methods do exist [180, 181], but their accuracy is relatively low and insufficient for use in a deployed system. For this reason, we simply associate bounding boxes with the previously detected diver they most overlap with, as long as the IOU between the two bounding boxes is greater than 0.25. Any bounding box which has not been assigned to a diver track at the end of this process initiates a new diver track if the probability of the detection is greater than 0.65. Additionally, diver tracks that have not had a new bounding box added to their track for 30 seconds or longer are culled from the list. While “identities” are assigned to each track and the DCM can differentiate multiple divers in the scene, these identities are not true identities, and cannot be uniquely assigned to an individual. Further, while this algorithm can accurately track a diver as they move across the frame, it is incapable of re-identifying divers who return to the frame after leaving. Future research should focus on unique diver identification, which would solve both issues. However, for our uses, this level of identification and tracking is sufficient.

9.4.2 DeepLabCut-Live On Demand

For every recently seen diver track, the DCM requests a pose estimation from a DeepLabCut-Live model. Firstly, the DCM calculates a confidence-aware time-weighted average of the bounding boxes associated with a diver track, then sends a ROS service call containing the average bounding box. Upon receiving the service request, the pose estimation node uses the bounding box to run inference only on a cropped portion of the overall camera image. The pose estimator then responds to the service with a body pose estimation message, providing the DCM with pose data to associate with the diver track. This method was selected over running pose estimation on the entire image to reduce the inference time of pose estimation and improve accuracy at distances by focusing only on a small portion of the image. It also provides a simplification over inferring diver poses across the image by eliminating the question of assigning poses to diver tracks. At this stage, the DCM also calculates a confidence-aware time-weighted average of the diver’s recent poses.

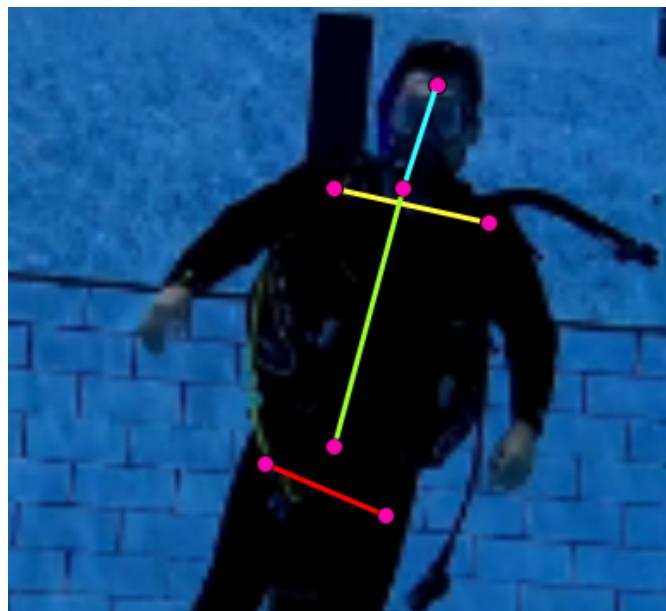


Figure 9.2: The body distances used to calculate multi-bone pseudodistance. The relevant keypoints are in pink, with the bust distance cyan, shoulder width yellow, trunk length green, and hip width red.

9.4.3 Relative Pose and Multi-Bone Pseudodistance

Now the diver context module has a list of diver tracks, each with recent and averaged bounding boxes and body poses. The only remaining step is to use this information to calculate the position of the diver relative to the AUV. We utilize the same construction of relative position we developed in Chapter 7, with some minor modifications. Firstly, to calculate the center point of the diver by processing the bounding box and pose and finding their center points. If both are available, we calculate their average, otherwise defaulting to whichever source is available.

Our psuedodistance algorithm is more complex. First, we update our definition to expand the possible values to the range between 0.0 and 5.0 (although values greater than 5.0 are possible, they have the same meaning as 5.0). Second, our method of calculating this pseudo-distance differs. Instead of simply comparing the distance between a diver’s shoulders, we consider four distances between body keypoints: the bust (head and sternum), trunk (sternum and waist), shoulder width (left and right shoulder), and hip width (left and right hip). These distances are shown in Figure 9.2. We calculate every available distance, then calculate the ratio between the image width or height and the body part distance. Multiplying this ratio by our target ratio, we calculate the pseudodistance according to each body distance. With these values determined, we calculate the variance between the psuedosdistances. If the variance is low, we average the body-distance values and use the results as our final pseudodistance. However, if the variance is greater than 2.0, we default to using a bounding box based pseudodistance, calculated by comparing the area of the bounding box to the area of the image and multiplying by a target ratio. We also default to this approach if there are fewer than 2 available body pose distances.

This “multi-bone” approach was chosen to avoid inaccurate psuedodistances caused by diver rotations. For instance, if a diver is facing their side to the AUV’s camera, a psuedodistance based only on shoulder width could incorrectly identify them as being quite far away. With the multi-bone psuedodistance however, the length of the neck and trunk will correctly identify the diver’s distance. This method is not entirely immune to these perspective problems but is more robust than the original psuedodistance introduced for ADROC.

9.4.4 Diver Context Output

Finally, with the bounding box, body pose, and relative position of each diver track collected and calculated, the DCM publishes a ROS message containing every diver track currently in the list. The message contains a list of divers, each of which contains the most recent bounding box and pose, the weighted average bounding box and pose, the relative position (including multi-bone pseudodistance), a flag indicating if they are currently visible, and timestamps of their most recent positions. Divers are kept on the diver track list for 30 seconds after their last observation. After 30 seconds, it is assumed that any information on the diver in question is sufficiently out of date to be useless.

9.4.5 Recommended Future Additions

The information published by the diver context module is useful for a variety of applications: diver following or ADROC could be reimplemented using the DCM’s outputs, and we will use it for context-adaptive communication in the following chapter. However, some improvements could be made with better input data. As previously mentioned, unique diver identification would be useful for creating more reliable tracks, as well as enabling identity-specific behaviors and memory. Additionally, the estimation of a diver’s gaze direction would provide useful information for modulating AUV communication, as well as for estimating diver attention. Lastly, while it is currently out of the reach of current diver perception methods, activity recognition for divers would be a beneficial input to the DCM, as it would allow AUVs to consider the activity that the diver is engaged in before making decisions about communication or actions relating to the diver.

9.5 Conclusion and Future Research Directions

In this chapter, we presented PROTEUS, a software framework for underwater human-robot interaction. Three components of PROTEUS were discussed: the language and symbol definition format, the implementations of AUV-to-human communication vectors, and the diver context module. As mentioned at the opening of this chapter, PROTEUS has a larger design than has currently been implemented. While the five

communication vectors and diver context module provide a strong basis for the adaptive AUV-to-human communication method we present in the next chapter, PROTEUS is far from feature-complete. In this final section, we briefly expand on five additions that should be a focus of future work. Some of these topics are simple development problems, while others are second-order HRI research, dealing with the questions of how to interact effectively and adaptively, beyond the simple matter of communicating.

Diver Input Methods

PROTEUS currently has no compatible diver input methods. It would helpful to implement fiducial marker input as a dependable backup, but if the POSH-G recognition system is ever developed to a point of proper usefulness, PROTEUS structures should be built around the system, codifying the symbols of the Protean language into language definition files and defining a full syntax for the language.

Conversational Control

A related topic is the development of conversational control. By parsing input sequences of Protean gestures, PROTEUS should be able to determine the diver's requests and create dynamic actions to fulfill them. This topic can be expanded further into questions such as when the AUV should agree to do a task and when it should refuse based on the risk involved, as well as when the AUV should ask for clarification on uncertain or confusing requests. These questions have been posed before [44], but with the more advanced perception capabilities and greater variety of communication available within the PROTEUS system, more can likely be achieved.

Mission Planning Structure

While the two previous points deal with the creation of dynamic autonomous behaviors based on diver input, PROTEUS should also be capable of working within a preplanned mission structure. Pre-existing software can likely be used [268] to actually create and manage behavior through planning and scheduling algorithms. However, this software will have to be adapted to communicate with other components of PROTEUS.

Environmental Context Node

The diver context node provides helpful information about divers in the AUV's vision. Similarly, it would be useful to have a record of other recently seen objects and entities. While a wide variety of object detection, semantic segmentation, and other perception algorithms have been developed for AUVs, selecting the most useful data and combining it into an environmental context module would be beneficial, both for the conversational control and mission planning structures mentioned earlier.

AUV Actions

A number of AUV actions such as diver following and approach are possible with the information provided by the diver context module. However, they have not yet been implemented and integrated into PROTEUS. It would be ideal for these methods to be triggered by ROS Actions, and defined in the language definition file so that the system can be aware of the actions the AUV is capable of. Other actions are not yet possible but would be valuable additions once the necessary research problems have been solved. Examples of these actions include area surveys, tool or specimen carry tasks, or leading a diver to a point of interest. AUV actions are, in many ways, the final step in creating a helpful co-AUV. Once the AUV can effectively communicate with a diver, make, modify, and understand plans, and reason about its environment, adding a greater number of complex actions will increase the AUV's usefulness.

Chapter 10

Autonomous Communication Vector Selection

Throughout this thesis, we have discussed a wide variety of capabilities intended to enable collaboration between AUVs and divers. In Part I we introduced three vectors of AUV-to-human communication, followed by diver perception capabilities in Part II, and a software architecture for UHRI in Chapter 9. Now, we introduce the last component of this thesis: Autonomous Communication Vector Selection (ACVS), a novel system for context-adaptive AUV-to-diver communication underwater. The purpose of ACVS is to improve the quality of communication underwater by utilizing the vector or vectors most appropriate to the situation. This concept was born out of the results of Study III (Section 2.8, which revealed that motion, light, and sound-based communication each had unique strengths and weaknesses, particularly depending on the relative position of interactants. We hypothesize that with an adaptive system for vector selection, AUV-to-diver communication will increase in efficacy and robustness, in turn improving outcomes of collaborative work. This serves the ultimate goal of this thesis, improving AUV-diver collaboration and interaction, as well as combining aspects of the majority of the topics and systems previously discussed including RCVM, HREye, SIREN, diver detection, diver pose estimation, psuedodistance, and PROTEUS.

In the following sections, we describe ACVS, beginning with a brief background in

adaptive communication. While HRI research in terrestrial domains has explored adapting communication to the interactant and context, ACVS is the first algorithm of its kind in underwater HRI research. Next, we provide a detailed description of the structure of ACVS, including the communication policies which define its behavior: a random policy as a baseline, and two heuristic-based policies, one of which allows the selection and combination of multiple vectors. We then describe Study VII, a case study with four well-trained AUV operators performing two tasks with the aid of communication from the LoCO AUV, managed by the ACVS system. The results of this study show improved accuracy in communication and significantly more successful task outcomes, validating our approach to adaptive communication for AUVs. Finally, we conclude with a small set of recommendations for future investigation of adaptive communication for AUVs.

10.1 Background: Adaptive Communication

Adapting robot communication to the context of the interaction is a topic that has not previously been explored for underwater robots. Indeed, the vast majority of research on underwater robot communication methods is focused simply on the efficacy of the system itself, with no complementary communication methods. For this reason, there is no relevant work to discuss in the field of UHRI, except for the different types of AUV-to-diver communication that have been developed (which we summarized in Section 1.5 and expanded in Chapters 2 - 4). However, adaptive communication and interaction has been studied for some time in the fields of human-computer interaction (HCI) [269] and human-robot interaction (HRI) when we consider terrestrial environments. The methods that have been developed for HRI most focus on user-adaptive interaction.

User-Adaptive Interaction

The bulk of adaptive interaction methods in HRI focuses on adapting communication to a user's preferences or needs, often for social or assistive robotics. Martins *et al.*, [270] provide a survey of this kind of work, separating systems into three categories: systems with no user model that only respond to immediate feedback, systems with a static user model, and systems based on dynamic user models which update over time. These types of approaches have been used to create robot interaction experiences that

more closely match the user’s personality [271] or the user’s communication frequency preferences [272], provide distinct agents to different users [273], and adapt to the interactant’s ability to see and hear [274]. A number of works have also explored the task of learning interaction patterns by observing human actions and reactions [275, 276], which has some overlap with the problem of learning from demonstrations [277]. Other methods create predefined interaction patterns which are selected by observing human behavior [278]. In general, user-adaptive methods provide more pleasant and robust interactions by matching a user’s preference. While this is an interesting area of research, it is not the focus of ACVS.

Environment-Adapative Interaction

ACVS is concerned with overcoming environmental constraints by multi-modal interaction, adapted to the environment and other factors. This has been significantly less studied than user-adaptive interaction. A few methods in terrestrial HRI have addressed adapting robot physical robot interaction to the environment [277], but more commonly, robot communication systems are designed to modify their own position and select desirable interaction distances and orientations [56, 279]. Many of these methods focus on the social aspects of communication, using Hall’s proxemic zones [280] as a guiding concept. Both Huettenrauch *et al.*, [281] and Syrdal *et al.*, [282]’s early investigations of this topic quantify user interactions and spatial relationships with robots through case studies. Safeea *et al.*, [283] explores an aspect of spatial relationships that is less social in nature: safety. Bethel [56] explores the selection of communication and interaction methods based on proxemic zones but is largely focused on the affective aspects of communication rather than communication efficacy. Overall, existing methods for environment-adaptive communication are mostly concerned with modifying robot position rather than adapting the method of communication to the current relative position of the interactant.

10.2 Design of ACVS

In this section, we present our algorithm for Autonomous Communication Vector Selection (ACVS). Working within the PROTEUS UHRI software structure, ACVS responds

to ROS Actions which request communication to a diver by selecting one or more communication vectors using a communication policy. ACVS communication policies (covered in depth in the following section) make decisions based on the estimated distance and angle to the diver, the priority applied to the message, and the message's content. Once the vectors have been selected, ACVS sends the appropriate ROS node a service request to trigger the display of the symbol.

10.2.1 Goals and Limitations

Due to the experimental nature of ACVS, there are a number of limitations of the system as it currently exists. We briefly outline the goals of the ACVS system along with these limitations before discussing the algorithm's structure in more depth.

ACVS Goals

- G1** – Given a requested communication symbol, ACVS will produce a communicative phrase (kineme, luceme, soneme, digital display) from a vector appropriate to the situation.
- G2** – If the communication policy in use allows for it, ACVS will produce phrases from multiple vectors simultaneously.
- G3** – ACVS will provide modular adaptive vector selection, with no communication vectors or perception embedded directly in the system. This enables future extension by adding new vectors or perception capabilities.

ACVS Limitations

- L1** – ACVS bases its vector selection strategies on the relative distance/angle of one diver. The diver selected is the diver who is closest to the AUV, the most recently seen, and has the highest confidence score. Any other divers in the scene are ignored.
- L2** – ACVS does not personalize vector selection by identity.
- L3** – ACVS does not modify its communication policies over time, they are statically defined.

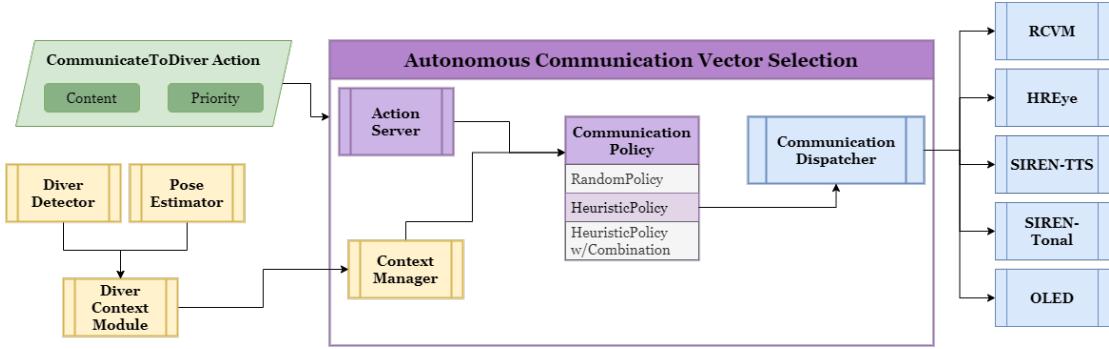


Figure 10.1: The Autonomous Communication Vector Selection system.

None of these limitations are set in stone, L1 and L2 could be achieved if true diver identification became available for use, and L3 is simply a further dimension of our current problem.

10.2.2 Overall Structure of ACVS

Our implementation of ACVS is a ROS node with four modules: the Action Server, Context Manager, Communication Dispatcher, and Communication Policy module. The structure of the system can be seen in Figure 10.1.

Action Server

The actions server is responsible for receiving and managing ROS actions that trigger communication. The *CommunicateToDiver* action, defined in the *proteus_msgs* package, consists of a request which specifies the symbol, priority, and any input data (a battery level percentage, for instance), and receives a response indicating which vectors were used to send the message. The action server module of ACVS is an implementation of a ROS ActionServer, allowing for requests to be sent and canceled, passing them on to the portions of ACVS which make the vector selection decisions and then returning a response based on the vectors selected.

Context Manager

The context manager listens to messages sent from the PROTEUS diver context module (See Section 9.4). It keeps its own record of the divers that the DCM reports, culling

them whenever they disappear from DCM messages. The most important function of the context manager is selecting an interactant from the current list of known divers. In the future, identity searching could be implemented here, but no, as per L1, we must select a diver from the list of known divers without considering identity. The context manager can do this using observation recency, relative distance, or confidence score, but by default, it uses a combination of all three. The following equations describe the interactant scoring process.

$$dist_score = 5.0 - diver.pseudodistance \quad (10.1)$$

$$recency_score = current_time - diver.last_seen \quad (10.2)$$

$$confidence_score = diver.confidence \quad (10.3)$$

$$interactant_score = dist_score * recency_score * confidence_score \quad (10.4)$$

When queried for the current interactant, the context module returns the diver with the maximum interactant score.

Communication Policy Module

The policy module is the core of ACVS. When the action server receives a message, it asks the policy module to select a set of communication vectors. The policy module does so according to one of its predefined policies (described in depth in the next section), sometimes using input from the context manager. Once the vector or vectors of communication are selected, they are passed to the communication dispatcher.

Communication Dispatcher

The communication dispatcher manages connections to the outside world of communication vector ROS nodes. For the most part, it simply sends ROS service requests to the selected vector's nodes, but in some cases, it may be required to pass dynamic input across topics. The communication dispatcher reads the list of possible vectors and

Distance	Min	Max	Angle	Min	Max	Priority	Min	Max
intimate	0.0	0.3	central	0.0	0.15	low	0	3
personal	0.3	1.5	paracentral	0.15	0.5	medium	3	5
social	1.5	3.0	peripheral	0.5	0.75	high	5	7
public	3.0	4.0	far peripheral	0.75	1.0	peril	7	10
edge	4.0	∞						

Tags	control	information	directional	interjection	dynamic
------	---------	-------------	-------------	--------------	---------

Table 10.1: The possible values for the context variables. Distance is the multi-bone pseudoistance described in Section 9.4.3, angle is a centrality ratio for the center point of the diver, and priority and symbol tags are passed with the communication request.

establishes connections to their endpoints by accessing data put on the ROS parameter server by the PROTEUS language server. Because of this, if one wanted to add a new communication vector to an AUV, the communication dispatcher would not require modification. In fact, the only part of ACVS which would require modification is the communication policy definitions, to add rules for the new vector.

10.3 Communication Policy Design

As previously alluded to, communication policies are the code of ACVS’s vector selection algorithm. We use three communication policies in this work, but more could be developed and integrated into the system seamlessly.

10.3.1 Random Communication Policy

The random communication policy is a baseline policy, randomly selecting one of the vectors advertised by the PROTEUS language server to complete each communication request. While an argument could be made for using a single, static vector policy as the baseline, we felt that the random policy was appropriate, as it represents the level of adaptation to interaction context that previous methods have had: none. The random policy is capable of returning a configurable number of random vectors, but by default

		Communication Vector				
	Context Value	Digital Display	SIREN-TTS	SIREN-Tone	HREye-Active	RCVM
Distance	<i>intimate</i>	2.0	3.0	0.5	1.0	0.1
	<i>personal</i>	0.5	2.0	1.5	1.5	0.75
	<i>social</i>	0.1	0.5	2.0	2.0	2.0
	<i>public</i>	0.0	0.25	2.0	2.0	2.0
	<i>edge</i>	0.0	0.0	1.5	1.0	2.0
Angle	<i>central</i>	1.0	1.0	1.0	1.0	1.0
	<i>paracentral</i>	0.25	1.0	1.0	1.5	1.0
	<i>peripheral</i>	0.0	1.5	1.5	0.5	2.5
	<i>far peripheral</i>	0.0	2.0	2.0	0.25	5.0
Priority	<i>low</i>	1.0	1.0	1.0	1.0	1.0
	<i>medium</i>	0.5	1.5	1.5	1.5	1.5
	<i>high</i>	0.25	2.0	3.0	2.5	2.0
	<i>peril</i>	0.0	2.0	3.0	3.0	1.5
Content	<i>control</i>	1.0	3.0	3.0	3.0	1.5
	<i>information</i>	2.0	2.0	1.5	3.0	0.75
	<i>directional</i>	1.0	1.0	0.75	3.0	2.5
	<i>interjection</i>	1.5	1.5	3.0	3.0	2.0
	<i>dynamic</i>	1.0	1.0	0.0	0.0	0.0

Table 10.2: Vector suitability heuristics for the Heuristic and Heuristic_Combo policies. Each context value is multiplied with the others, and the highest overall score is selected. Therefore, a 0 value blocks the vector from being selected, and the higher a context value is, the higher the final suitability score will be.

only returns one.

10.3.2 Heuristic Policy

The heuristic policy is the first of our experimental communication policies. It scores each communication vector's suitability for use based on four values: the pseudodistance to the interactant, the interactant's centrality in the frame (termed pseudoangle), the priority passed by the communication request, and the content tags related to the

requested symbol. These are our context variables, which describe the context of an interaction for ACVS. Content tags are applied at the PROTEUS language definition level, for precisely such a purpose as this. Once a communication request has been received, the communication policy module requests interactant data from the context manager, pseudodistance and angle to the diver being the primary focus. Using the center point of the diver, a ratio is calculated between the center of the image and the center of the diver, indicating how close to the center of the robot's view the diver is. Having received distance and angle information, the message priority, and content tags are gathered and all four values are classified by the ranges in Table 10.1 Next, the heuristic policy selects a value from its context value table (Table 10.2) by finding the correct value for each context variable and vector. If the value from the context table for a context variable *context* is:

$$CVT(context, vector) \quad (10.5)$$

the suitability score of a vector is:

$$\begin{aligned} suitability_score(vector) = & CVT(distance, vector) * CVT(angle, vector) \\ & * CVT(priority, vector) * \sum_{i=1}^{ntags} CVT(tags_i, vector) \end{aligned} \quad (10.6)$$

The vector with the highest suitability score is selected as the communication vector. The heuristic policy values are defined in a YAML file which is loaded by the communication policy module and can be seen in Table 10.2. The values of the context value table were set based on a combination of common sense and our experience in developing and evaluating these vectors for AUV communication.

10.3.3 Heuristic Policy with Combination

The heuristic policy with combination (also referred to as Herustic_Combo) is the same base policy as the heuristic policy, with one major modification. After selecting the best possible vector, the heuristic combination policy selects a configurable number of additional policies (defaulted to one). This is achieved by calculating the ratio between the selected vector's suitability score and all remaining vector's scores. For a selected vector $vector_S$ and a potential combination vector $vector_P$, the chance of combination

		Vector Selected				
		Digital Display	SIREN-TTS	SIREN-Tone	HREye-Active	RCVM
Combo Opt.	Digital Display	N/A	0.25	0.1	-0.25	-1.0
	SIREN-TTS	0.25	N/A	-1.0	0.15	-0.5
	SIREN-Tone	0.25	-1.0	N/A	0.25	0.5
	HREye-Active	-0.1	0.15	0.15	N/A	-1.0
	RCVM	-0.5	-0.5	0.3	0.3	N/A

Table 10.3: Combination rules for the Heuristic_Combos policy. These values are added to the combination probabilities calculated for each vector, then clamped to [0, 1.0]. A positive value (blue) increases the likelihood of use, a negative value (red) decreases the likelihood, and N/A (gray) indicates that the vector cannot be combined with itself.

is:

$$\text{combo_chance}(\text{vector}_P) = \frac{\text{suitability_score}(\text{vector}_S)}{\text{suitability_score}(\text{vector}_P)} + \text{combo_weight}(\text{vector}_S, \text{vector}_P) \quad (10.7)$$

The value of *combo_weight* is defined for each vector pair and can be seen in Table 10.3. Some vectors are completely disallowed from being selected as a combination vector (for instance the two types of SIREN sonemes cannot be used simultaneously) and other vectors are made more or less likely to be selected based on how they are expected to interact with the chosen vector. The combination weights are set based on our experience and research on these forms of communication.

10.4 Study VII: Adaptive Communication Evaluation

In order to evaluate our approach for context-adaptive communication, including three different communication policies, we undertake a final human study. This study is groundbreaking in terms of the complexity of the task that the diver is asked to complete with the robot's help, and the amount of training that participants must complete. Since participants must be familiar with all five PROTEUS-compliant communication vectors that ACVS can use, hours of training are required for each participant. Additionally, the pool evaluation of the ACVS took between 90 and 150 minutes for each participant. For

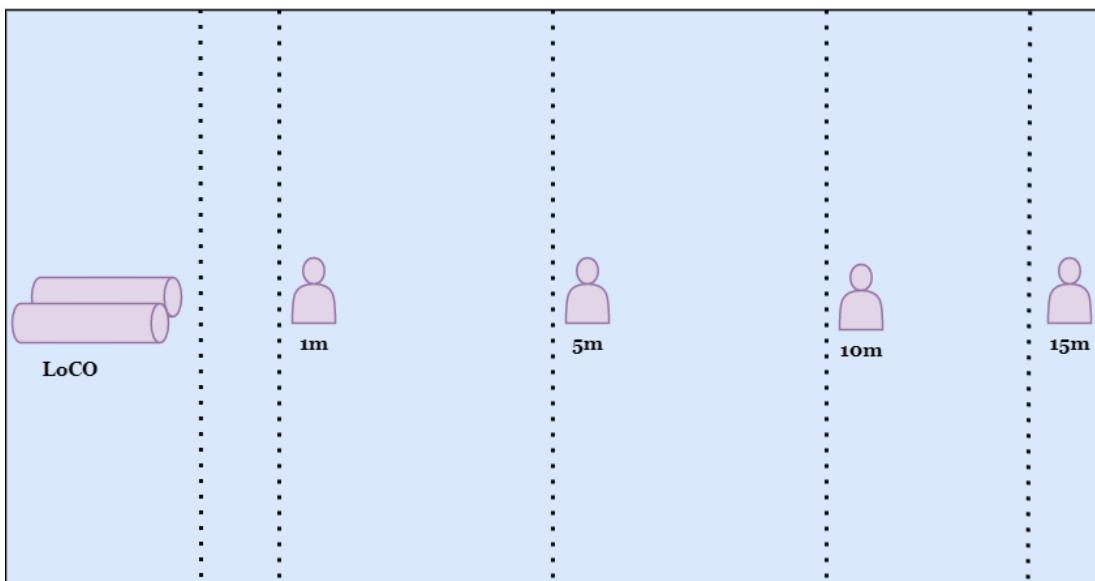


Figure 10.2: A diagram of the Underwater Telephone task for Study VII. Note the four different distances at which the diver may observe the robot.

this reason, Study VII has a much smaller participant population than previous studies presented in this document: four participants were trained, evaluated, and debriefed.

10.4.1 Study VII Design

Study VII is a small case study of four participants, organized into two separate tasks. The first task is a simple communication recognition task we refer to as the Underwater Telephone task, in which participants observe three communications from the AUV, surface, and report the symbols they observed. The second task, Oracle Search, is a task designed to be representative of searching and coverage tasks that might be undertaken by divers in the real world. In Oracle Search, participants must communicate bi-directionally with the AUV for critical information about the task. Participants were equipped with scuba gear for both tasks. While the ACVS portion of the task was managed entirely autonomously, diver communication to the robot and decision making about what messages to send via ACVS were managed by study staff, making this a partial Wizard-of-Oz study.

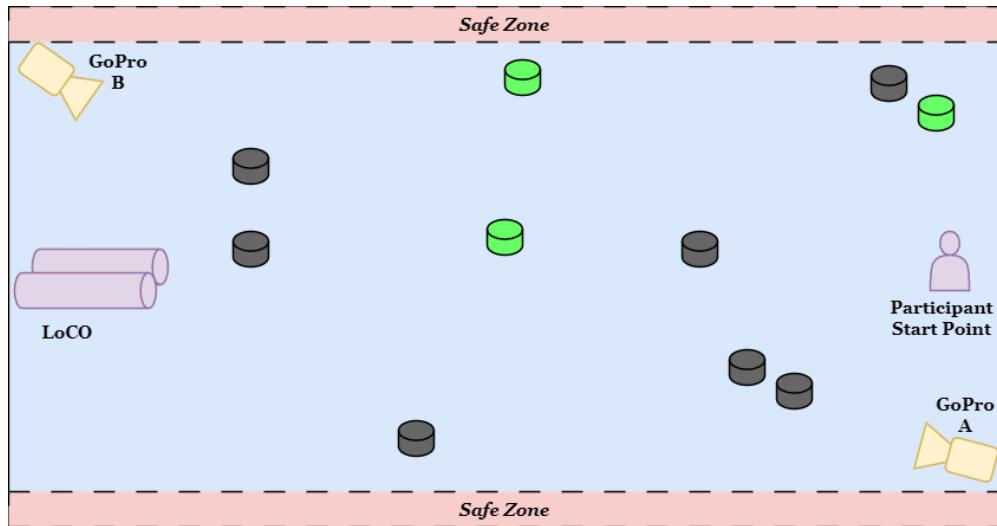


Figure 10.3: A diagram of the Worf scenario of the Oracle Search task for Study VII. The black cylinders are empty containers, while the green cylinders have a target item within them. Uncontained items are not diagrammed, as their location is set randomly.

Underwater Telephone Task

For the Underwater Telephone task, participants were asked to submerge themselves below the water and observe the robot for a few moments. The AUV was then triggered by study staff to display between one and three randomly selected phrases, using ACVS to select the communication vectors used. The participant was then signaled to surface by another member of study staff, after which they reported the phrases that they had observed. We chose to randomize the number of phrases so that the a phrase being missed by the participant would be more obvious, instead of the participant simply guessing for the phrases they did not perceive. This process was completed four times per communication policy at a distance of one meter from the robot. Once that was completed, two other distances were tested. These varied between participants, but included five meters, ten meters, and fifteen meters. The pool setup used for the Underwater Telephone task is shown in Figure 10.2.



Figure 10.4: Containers and uncontained objects used in the Oracle Search task. Note the

Oracle Search Task

The second task of Study VII, Oracle Search is centered around the collection of target items from the pool floor. These target items are small plastic fish toys, six of which were scattered across the pool floor randomly before each iteration of the task. A further three were placed inside small opaque plastic containers with color tape marking them. Examples of the containers and target items are shown in Figure 10.4. These containers, along with seven other empty containers, are placed on the pool floor in pre-defined locations, specified by a “scenario code”. The placement of the containers, as well as which color codes contain target items changes between each scenario and is kept secret from participants, being used by study staff to provide the titular oracle of the task. Once the items are placed, participants are given seven minutes to collect all of the target items that they can find, but are not permitted to open the containers. To determine which containers hold target items, the participant must communicate with the AUV, asking it for directions to the nearest container with a target, or if a container has a target within. Additionally, no less than once (and no more than twice) per scenario, the AUV randomly triggers a danger state, which it signals the participant about using ACVS. The participant must flee to a safe zone within thirty seconds of the danger warning, or they are marked as receiving a virtual “injury”. If they receive two

virtual injuries, the scenario is terminated. Lastly, the AUV can provide information about the amount of time remaining. A participant’s success in the Oracle Search task is calculated using the rules:

- +1 for each uncontained target item collected.
- +5 for each target item in a container collected .
- -1 for each empty container collected.
- -2 for each missing target item, contained or uncontained.
- If the participant takes one injury, their score is halved. If the participant takes a second, their score is zero.

The pool setup for the Oracle Search scenario coded “Worf” can be found in Figure 10.3.

Bidirectional Communication Language for ACVS

To support the bi-directional communication required for the Oracle search task, we make several modifications to our existing AUV languages for the LoCO AUV, and introduce a new language for diver-to-AUV communication using a flashlight. The communication protocol shown in Figure 10.5 lists the possible symbols that the robot and human can exchange in their communications. On the side of AUV-to-diver communication, the changes to language as compared to the kineme, luceme, and soneme sets previously described primarily consist of dropping symbols which would not be used in Oracle Search. Other changes in the language are the replacement of $K_{Directional}$ (and the equivalent lucemes and sonemes) with four cardinal direction symbols for left, right, forward, and back, and the re-purposing of $K_{RemainingBattery}$ (and the equivalent lucemes and sonemes) as $K_{RemainingTime}$. This language is used for the Oracle Search task, but is also used for Underwater Telephone task so that the participants only have to learn one version of the AUV-to-human communication symbols.

When it comes to the human-to-AUV communication for the Oracle Search task, we introduce a set of four gestures which can be performed with a waterproof flashlight. These gestures $FG_{WhereTarget?}$, $FG_{ContainsTarget?}$, $FG_{TimeLeft?}$, and $FG_{RepeatLast?}$ are pictured in Figure 10.6. These flashlight gestures are not recognized by an automated

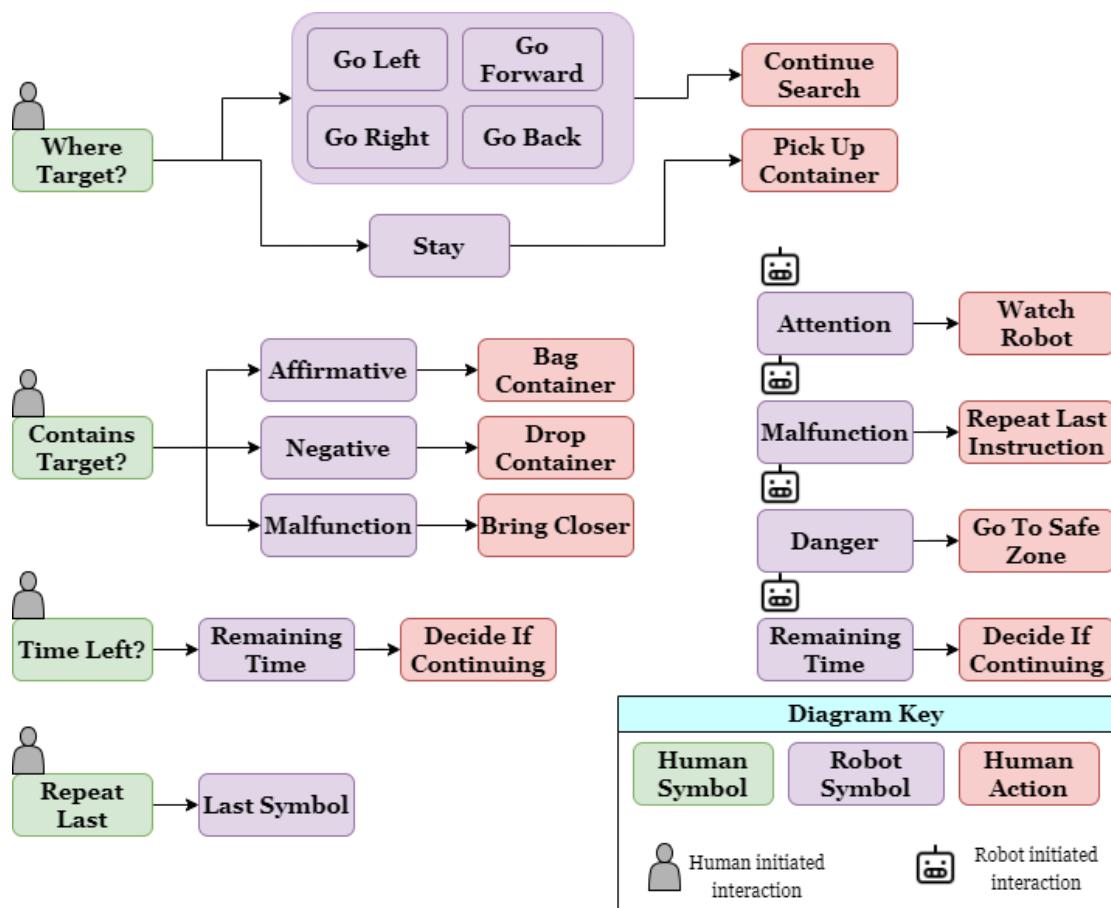


Figure 10.5: The AUV-diver communication protocol used for the Oracle Search task.

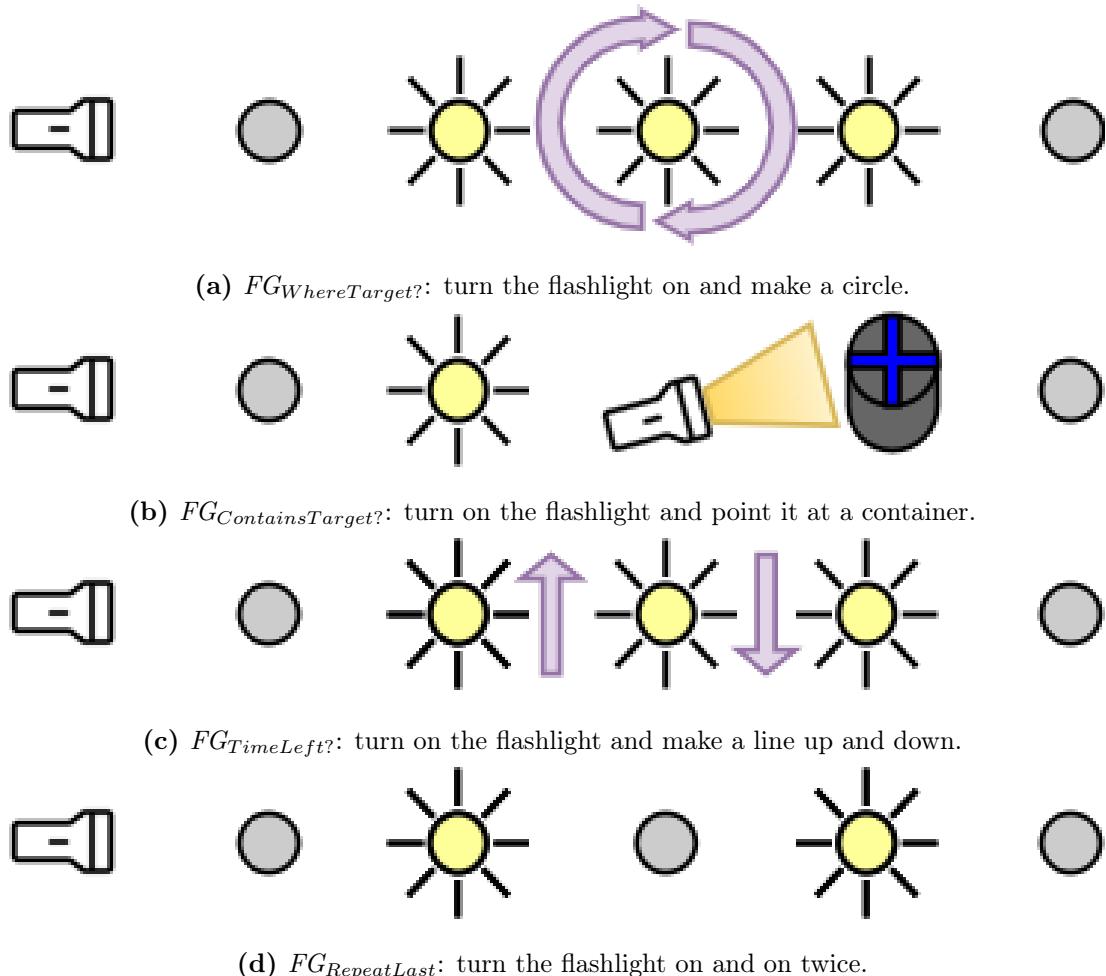


Figure 10.6: Depictions of the flashlight gestures used for human input in the Oracle search task.

system; rather the study staff operating on the poolside observe the robot's cameras, recognize the gestures, and input them to the control software for the Study VII tasks. This control software is responsible for selecting random times to trigger Danger during Oracle Search, randomly replacing the intended symbol input to ACVS with Malfunction (to simulate situations in which the AUV is unsure of its environment), and providing a structure for the study staff to manage the task. Study staff are responsible for recognizing flashlight gestures and determining the answers to questions about the direction to the nearest target item and whether or not a container has a target within it.

10.4.2 Study Administration

This study was submitted to the University of Minnesota's Institutional Review Board and determined to be human research. After completing a full protocol submission, it was approved under the study reference number 00017755. The study has four stages: recruitment, training, pool evaluations, and debrief.

Recruitment Procedures

Since this study requires a significant amount of time and effort, we chose to recruit participants from within our own group of underwater robotics researchers. This has obvious downsides: the risk of participant contamination, the risk of feelings of pressure on participants, and so on. In order to avoid the first issue, participants were kept from learning almost anything about the study until their training began. While many of them had been exposed to the communication methods that would be utilized by LoCO during the study, the participants will be trained to a point of competency on those very communication methods, making this a non-issue. So as to avoid placing undue pressure on participants, we carefully crafted the consent and recruitment forms they were provided with to specify protections on their privacy, independence, and ability to freely leave the study at any point.

Training Procedures

Once our four participants had been recruited from within our research group, we began the training process. One week prior to the date of their pool evaluation, participants

received the first of three training surveys. The first survey collected demographic information from the participants and taught them the five AUV-to-human communication systems they would be using: a digital display, RCVM, HREye active lucemes, SIREN TTS-Sonemes, and SIREN Tonal-Sonemes. Once they had completed the training, participants took a small competency test in which they identified a small number of symbols randomly selected from all of the AUV-to-human systems. The second survey, delivered three days before the pool date, continued this training, along with introducing the Underwater Telephone and Oracle Search tasks to the participants as well as the flashlight gesture language, finishing with a competency test which now included the flashlight gestures. Lastly, the third survey was delivered to participants twenty-four hours before their pool time and simply reminded participants of the languages they had been taught thus far, as well as providing them with a final competency test.

Pool Evaluation

Once participants had been fully trained, they were brought to a University pool for their evaluation step. The evaluation step for each participant took between 90 and 150 minutes. After preparing their scuba gear and beginning to dive, participants completed the Underwater Telephone task. For the Underwater Telephone task, participant answers were recorded in duplicate, along with their confidence in their answers, and the time that it took for them to answer. Once participants had completed their first task, participants were asked to complete three runs of the Oracle Search task, one for each communication policy. In between each Oracle Search scenario, a NASA TLX survey [110] was administered by study staff, evaluating the effort required by the participant for the task. For the Oracle Search task, the LoCO AUV recorded one of its cameras, the outputs of the diver context module, and all of the operations of the ACVS system, including the manually recognized flashlight gestures, the symbols requested from the ACVS system, the vectors selected, and the selection weights. Two GoPros also recorded each Oracle Search run. There were a number of small issues that resulted in some data loss during pool evaluations. Due to time constraints, one participant was unable to complete both of their tasks, so it was decided that they would not complete an Underwater Telephone run. Additionally, portions of some robot data for Oracle Search runs was lost due to transmission issues. Fortunately, the loss was rather minimal, and

the majority of scenarios were completed without incident.

Debrief Interviews

Each participant was given an Amazon gift card valued at 75 USD upon completing their pool evaluation procedures. Additionally, participants were asked to sit for a short interview, in which they discussed their experience during the study and provided insights into the aspects of LoCO’s communication which did and did not work for them.

10.4.3 Analysis Methods

The analysis of this study differs somewhat from that of our previous studies. The population size is small enough that statistical testing is of little use, and our traditional metrics such as accuracy and operational accuracy, which depend on participant-provided responses cannot be used to evaluate Oracle Search. Instead, we introduce three new metrics which allow us to provide a quantitative layer on top of a more qualitative approach to our data analysis: symbol perception, symbol understanding, and interaction quality.

Symbol Perception

Symbol perception is a score which ranges from 0.0 to 10.0, representing the extent to which a participant was aware that a communication symbol was performed for them. In the future, symbol perception could be calculated by using a diver attention estimation system, but at the current time, we evaluate it according to the following rubric:

- $S_{percept} = 0.0$ if there is no evidence that the participant perceived the symbol.
- $S_{percept} = 2.5$ if there is evidence that the participant perceived the symbol, but may have missed part of it.
- $S_{percept} = 10.0$ if the participant definitely perceived the symbol, regardless of their understanding of it.

Symbol Understanding

Symbol understanding is a similar metric to symbol perception, ranging from 1.0 to 10.0 instead of 1.0 to 10.0. It represents the accuracy of a participant's understanding, and has slightly more values, as it can often be evaluated by considering the participant's response to the symbol. Once again, we evaluate it according to the following rubric:

- $S_{understand} = 1.0$ if the participant did not perceive the symbol, or if their understanding is clearly incorrect.
- $S_{understand} = 2.5$ if the participant perceived the symbol, but their understanding cannot be determined.
- $S_{understand} = 7.5$ if the participant understood the symbol to some extent, but their actions leave doubt. This value is not used if a direct report of the participant's understanding of the symbol is available, and is rarely used if no such report is available.
- $S_{understand} = 10.0$ if the participant definitely understood the symbol based on their actions or reports.

Interaction Quality

Lastly, interaction quality is a metric which combines the values assigned to an interaction for perception and understanding scores with factors based on the time that the interaction took and the number of repetitions that were requested by the participant. Interaction quality ranges between 0 and 100, and is calculated using the following formulae:

$$Q_{int} = \frac{(p_r * r)(S_{percept} * S_{understand})}{\max(1, |T_{exp} - T_{actual}|)} \quad (10.8)$$

$$T_{exp} = T_{input} + T_{symbol} + T_{proc} \quad (10.9)$$

where $p_r = 0.15$ is the penalty for repetitions, r is the number of repetitions, T_{actual} is the duration of the interaction, $T_{input} = 3.0$ is the average length of an input symbol, T_{symbol} is the average length of whatever symbol the AUV is sending, and $T_{proc} = 1.0$

is the processing time allowance. Expressed in plain language, interaction quality is the combination of the perception and understanding scores, divided by the number of excess seconds that the interaction takes and reduced by 15% for each repetition that is requested by the participant.

Processing Interaction Data

We evaluate the collected data from the Underwater Telephone and Oracle Search tasks according to the rubrics described above. This process was time consuming and complex, particularly for Oracle Search. In the case of the Underwater Telephone task, the diver reports the phrases that they witnessed, making it relatively easy to evaluate if they perceived and understood a phrase. When evaluating Oracle Search however, we have no direct report. The three recorded video streams (from LoCO and both GoPros) were synchronized and analyzed one by one, determining for each communication attempted by the ACVS system whether or not the diver perceived it and if they understood it. This was done by the author rather than by raters, as the complexities of the system, the task, and the nature of the data make it extremely difficult to analyze. Care was taken, however, to ensure that no leniency was given in rating perception or understanding.

10.5 Results of Study VII

Having completed pool session for our four participants and analyzed the resulting data, we turn now to the results of this analysis. In particular: how well did the ACVS system perform in the Under Telephone and Oracle Search tasks, what was the effect of the different communication policies on quantities such as perception, understanding, and interaction quality scores, and how do these effects impact task success?

10.5.1 Underwater Telephone

We begin by looking at the results of the Underwater Telephone task. Our general results indicate strong, but not perfect perception and understanding of the communication attempted by ACVS. The average perception score is 6.1 overall, with an average understanding score of 4.7. This clearly indicates that ACVS-adapted communication is

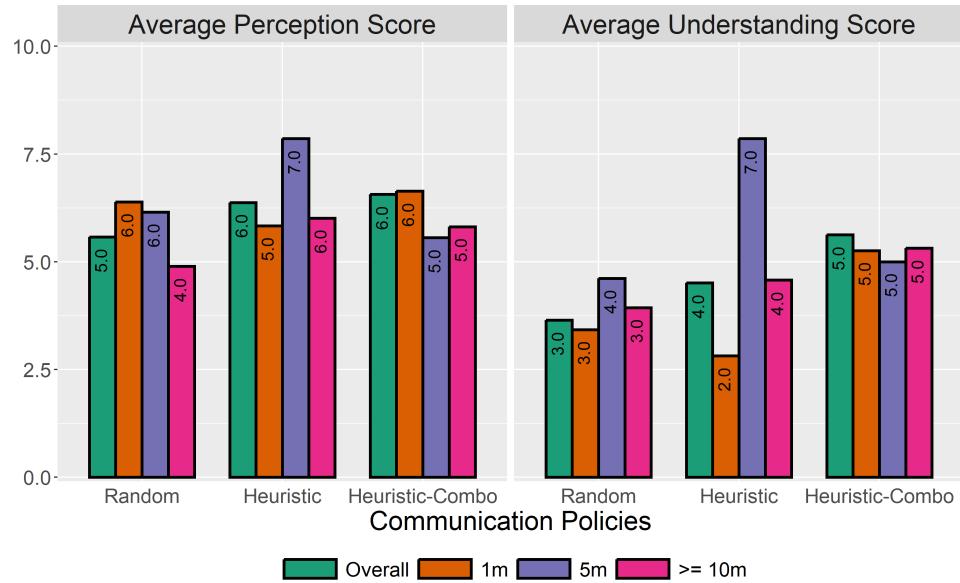


Figure 10.7: Perception and understanding of symbols communicated to divers during the Underwater Telephone task.

being perceived and understood, but there is room for improvement. The average perception score and average understanding score can be found in Figure 10.7, separated by communication policy and distance.

Effect of Policy

The three evaluated communication policies perform similarly in terms of perception, with the heuristic policies having a slight edge over the random policy. When consider understanding scores, we effect is more stark: the random policy has an average understanding score of 3, compared to the heuristic policy's 4, and the 5 achieved by the heuristic_combo policy. These improvements in understanding, achieved by adapting communication vector to the interaction context, should lead to improved task outcomes in Oracle Search and other communication heavy tasks.

PID	Scenario	Policy	Containers			Injuries	Duration	Task Score
			Correct	Empty	Missing			
918	La Forge	Random	2	0	1	1	7:06	7
918		Heuristic_Combo	2	1	1	1	7:23	6.5
918		Heuristic	1	1	2	1	5:35	2.5
73	Riker	Heuristic_Combo	2	0	1	0	7:11	14
73		Random	1	2	3	1	6:52	1
73		Heuristic	3	2	1	1	7:01	8
73		Heuristic_Combo	2	1	0	0	7:09	15
983	Troi	Random	2	0	1	1	7:12	7
983		Heuristic	3	0	0	0	4:42	21
983		Heuristic_Combo	3	0	0	1	5:10	10.5
616	Worf	Random	2	2	1	0	8:00	6
616		Heuristic	3	0	1	1	7:00	9.5
616		Heuristic_Combo	2	0	2	0	7:00	11

Table 10.4: Results of the Oracle Search task per attempt.

Effect of Distance

Considering the effect of distance on communication, we see the typical effect of distance on the random policy: perception and understanding both fall. The heuristic and heuristic_combo policies, other other hand, remain somewhat more stable at distance, particularly the heuristic_combo policy. We suggest that this may be due to the use of multiple, heuristically selected vectors, which improves the chances that a communication will be perceived and understood.

10.5.2 Oracle Search

The results of the Underwater Telephone task are encouraging, as they suggest that ACVS is selecting appropriate communication vectors for the context of the interactions. However, the real test is the effect of ACVS communication on a task such as Oracle Search. Table 10.4 shows the 13 completed Oracle Search scenarios, with data on the number of containers collected, virtual injuries sustained, and task score. Figure 10.8 also shows average perception and understanding score, interaction duration, and interaction quality. From the data in this Figure, it is apparent that the heuristic

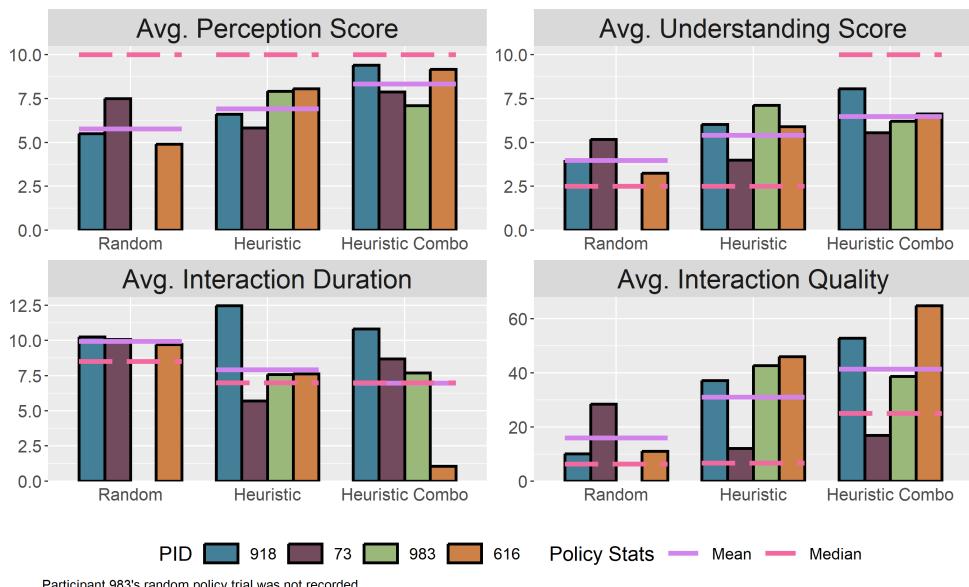


Figure 10.8: Summary of the performance of ACVS communication policies in Oracle search in terms of perception, understanding, interaction duration, and average interaction quality.

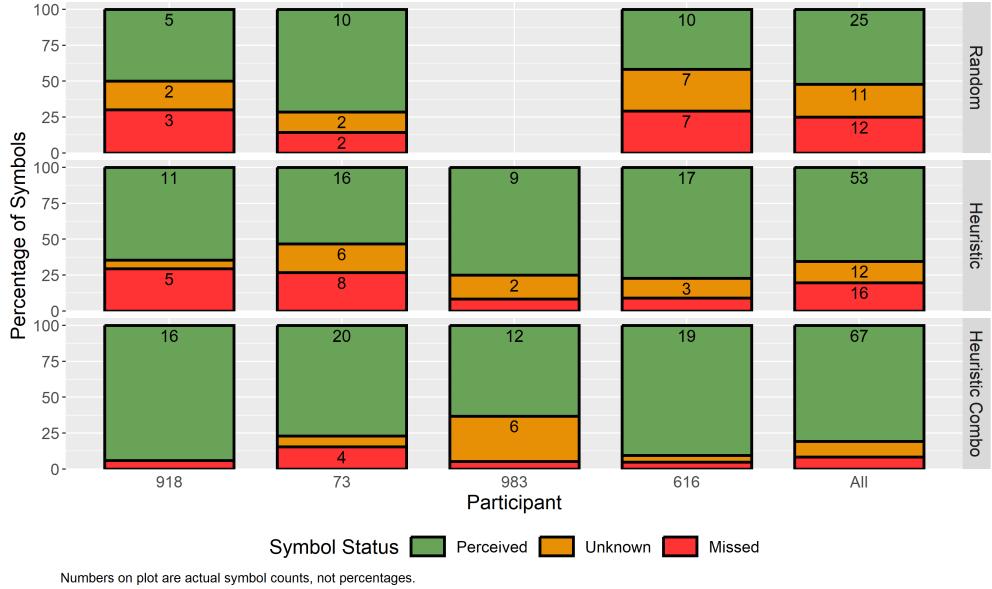


Figure 10.9: A per-interaction breakdown of participant perception of communication during Oracle Search.

policies outperform the random policy: perception and understanding both rise, interaction duration reduces, and average interaction qualities rises. The results of the Oracle Search runs in Table 10.4 validate this, with higher scores for runs using a heuristic policy.

Effect of Policy on Perception and Understanding

Figure 10.9 and Figure 10.10 show a per-interaction count of the understanding and perception of different participants when using different communication policies. The perception values in Figure 10.9 show a high number of missed symbols for runs with the random policy than for the heuristic_combo policy. While the random policy has a lower number of missed symbols than the heuristic policy, fewer total symbols were communicated with the random policy, so the missed symbols make up a larger percentage. Per-interaction understanding results in Figure 10.10 are more definitive: while every policy has some misunderstood or unknown symbols, the heuristic policies vastly outpace the random policy both in number and proportion of symbols that were fully

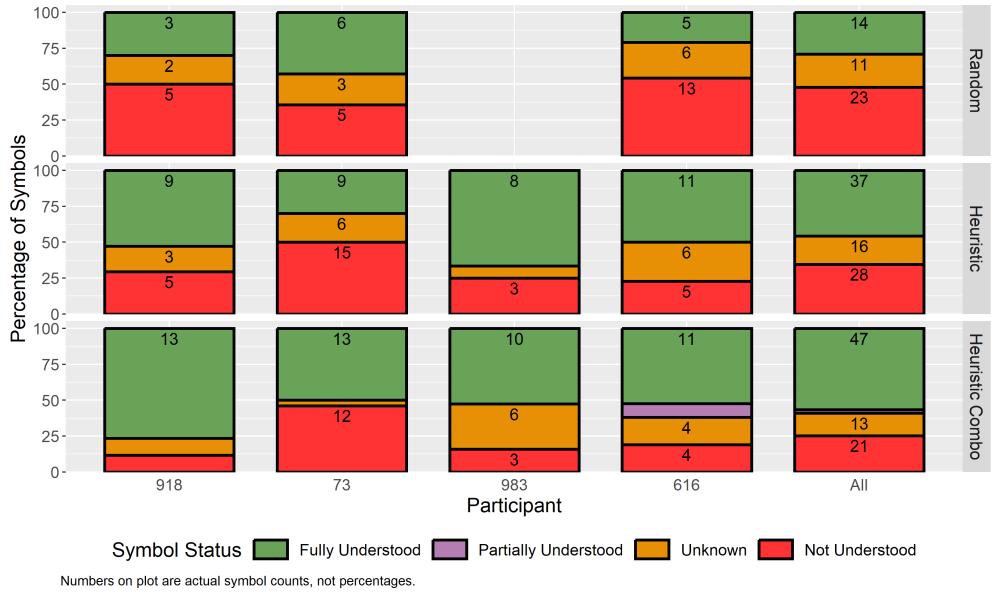


Figure 10.10: A per-interaction breakdown of participant understanding during Oracle Search.

understood.

Effect of Policy on Interaction Quality

The improved perception and understanding of heuristically-selected communication contributes to the improved interaction quality for these policies. Figure 10.11 shows that the average interaction quality of the heuristic policy is nearly twice that of the random policy. The heuristic_combo policy's interaction quality rises higher still, though not by two times.

Effect of Policy on Oracle Search Score

The heuristic policy-based communication also leads to higher task scores, though the task score is achieved by both versions of the heuristic policy is nearly equal. In Figure 10.12, we can see the components which lead to this: the heuristic policies lead to a lower number of incorrect items being picked up, and a higher number of correct items. Because the communications of the AUV are more frequently and clearly perceived and

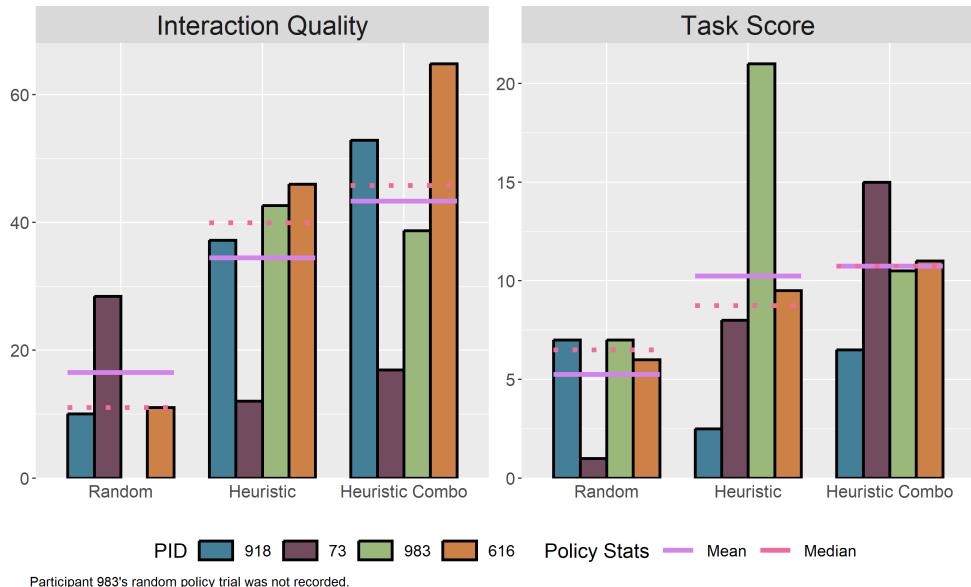


Figure 10.11: The effects of communication policy on average interaction quality and task score during Oracle Search.

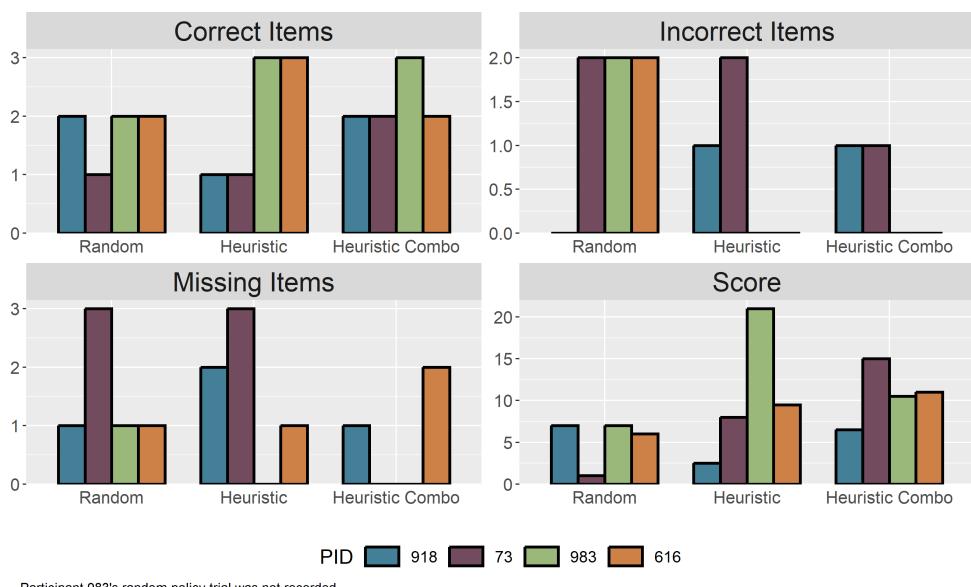


Figure 10.12: Average number of correct containers and incorrect containers collected, along with missing items in oracle search.

Survey Question	Random		Heuristic		Heuristic_Combo	
	<i>Mean</i>	<i>Median</i>	<i>Mean</i>	<i>Median</i>	<i>Mean</i>	<i>Median</i>
How mentally demanding was the task?	34	24	32	29	30	19
How demanding was the task in terms of time?	65	78	47	53	58	74
How physically demanding was the task?	34	32	43	50	58	69
How hard did you have to work to accomplish your level of performance?	73	86	70	70	71	70
How successful were you in doing what you were asked to do?	60	65	50	60	50	60
How insecure, discouraged, irritated, stressed, and annoyed were you?	41	35	29	20	22	10

Table 10.5: Results of the NASA TLX surveys administered after Oracle Search scenarios

understood, the participant achieves better results in the task. This, combined with the previous result, validates our core hypothesis: adapting communication vector selection to the context of the interaction results in higher performance on collaborative work underwater.

Effect of Policy on Effort and Difficulty

The NASA TLX surveys conducted after each search scenario reveal interesting facts about the perceived difficulty of the task, shown in Table 10.5. While participants rated the level of effort they had to commit to the task more or less the same regardless of policy, we see higher levels of frustration for the Random policy, along with higher

ratings for the demand on time. Interestingly, the rating of physical demand increases for the heuristic policies. We suggest that this may be due to the shorter interaction times and fewer repetitions caused by high quality communication. When the diver is not station-keeping, waiting for a response that they may not perceive, they are doing less work than if the robot responds promptly and perceptibly.

10.5.3 Analysis of Interviews

The interviews conducted with participants after their pool sessions covered a wide variety of topics, but a two salient points are summarized below:

Scuba Interferes With SIREN

Every participant mentioned having difficult hearing sonemes, particular TTS-Sonemes, over the sound of their breathing apparatus. Several even mentioned holding their breath in an attempt to better hear the robot – participant 73 said the following: “I can’t hear very well underwater. So any of the sound based systems were impossible to hear, especially because I found myself having to stop breathing a bunch of times to try and hear.” This is concerning, and requires further investigation. One concept that emerged during these interviews was the idea of synchronize sonemes to fall between breaths, but this is a difficult method with uncertain viability. Others possible approach’s to address this problem include frequency analysis to determine which frequencies are easiest to hear over the noise of a scuba device, or attempting to use longer sonemes, which can be heard between breaths by virtue of simply continuing through multiple breath cycles.

HREyes Are Highly Preferred

Participant 616 said the following: “I’d say the lights for sure that that seemed the best” (for determining where a container was). Other participants echoed this sentiment for a variety of tasks, indicating a high preference for HREyes overall. This is likely impacted by the fact that when the participant is looking at the robot, the HREyes can operate normally, without the issues that were plaguing the SIREN systems.

10.6 Conclusion and Future Directions

In this chapter, we presented ACVS: a novel method for selecting communication vectors based on the relative position of an interactant and communication priority and content in underwater environments. This system utilizes the perception and communication capabilities introduced earlier in this document, making it a piece of “second-order” UHRI research, one of the few which have ever been conducted. To evaluate our adaptive communication system and the three communication policies we developed for it, we also presented a seventh human study. This study was distinct from the others presented in this thesis with a smaller population, a more complex task, and a more qualitative method of analysis. Nonetheless, the results from Study VII are no less impactful than the previous chapters’ results: heuristic based ACVS communication policies outperform the random choice policy baseline. Not only do they perform better in terms of producing perceptible and understandable communication, they increase interaction quality overall, which leads to improved results in the collaborative Oracle Search task of Study VII. Having demonstrated ACVS to improve interactions and collaboration in underwater environments, we can conclude the final piece of research in this thesis with a positive result.

Possible Areas of Future Exploration

As with previous chapters, we provide a small number of possible directions for future research on this topic.

Adapting Communication Policies To Environmental Issues

One aspect of adapting communication vector selection policies that will certainly become of importance in the underwater HRI community is adapting to new and different environments. With the current approach, a high-visibility and low-visibility environment will be treated the same when it comes to selecting communication vectors. Adding structures for adaption to differing levels of visibility would be a good first step, but for this addition to ACVS to truly shine, it would require a sensor-based estimation of the visibility of the current environment.

Adapting Communication to Participant Attention

As previously mentioned in the discussion of the diver context module, attention estimation is an extremely useful type of data to have about divers. This is especially true for ACVS, where the extent to which a diver is paying attention could not only affect the selection of vectors, but be used as a way to autonomously determine if a diver has perceived a communication. The ability to determine if the interactant has perceived and understood the communication dispatched by ACVS would not only be helpful for further analysis, but for modifying communication policies over time.

Modifying Communication Policies Over Time

Our current heuristic policies are static, as we do not currently have any input to modify based off of. However, with properly integrated visual attention estimation and gestural control, the ACVS system should be able to reason about the probability that a diver has perceived and understood communications, enabling a feedback loop which modifies the communication policy based on success rates. This active reinforcement learning method could be applied to learn communication policies from scratch, or merely modify them to improve them for lifelong operation.

Conclusion

Summary of Presented Research

Over the course of the ten chapters that make up this thesis, we have introduced the components of a new paradigm for underwater human-robot interaction. In Part I, we presented three groundbreaking methods for natural AUV-to-human communication, each of which added an entirely new level of natural communication capabilities to AUVs. Robot Communication Via Motion (Chapter 2) is a first in the field, using the motion of an AUV to communicate information to divers using natural and intuitive body language gestures called kinemes. Chapter 3 presented the HREye, a biomimetic light-based communication device for AUVs that is not only capable of replicating the meanings of RCVM's kinemes as light codes called lucemes but also of communicating gaze direction through ocular mimicry. Lastly, Chapter 4 discussed SIREN, a novel sound-based communication system that also replicates the meanings of the kinemes, this time in two forms of sonemes: one based on synthetic speech and another based on musical tones. Human studies evaluating each of these systems were reported, showing strong recognition and natural understanding of each type of communication.

We continued in Part II, advancing the state of the art in diver detection, developing a novel capability of diver motion prediction, a monocular-vision-based method for diver-relative distance estimation and diver approach, and a novel gesture recognition system for AUVs. The diver detectors and dataset presented in Chapter 5 outperform previous approaches and explore a new aspect of the performance of a diver detector: temporal stability of detections. This dataset also enables the creation of the diver motion prediction method discussed in Chapter 6, which is an adaption of pedestrian motion predictions to divers underwater, the first method for diver motion prediction

ever proposed. ADROC, the algorithm for diver approach covered by Chapter 7, expands on the world of diver-relative navigation by enabling an accurate and robust diver approach. Core to ADROC is the method of monocular diver distance estimation we invented using biological priors: pseudodistance estimation. The last chapter of Part II introduces POSH-G, a one-shot gesture learning method for recognizing dynamic gestures underwater. A first for the world of UHRI, POSH-G uses diver pose estimation models trained to capture body keypoint trajectories, generates thousands of examples from each one, and then autonomously tune recognition algorithms, which have achieved accuracies as high as 50%.

Part III brings these components together under the structure of PROTEUS (Chapter 9), a UHRI software framework that contains implementations of the communication methods of Part I and utilizes the perception capabilities of Part II. The final chapter of Part III covers Autonomous Communication Vector Selection, a novel method for context-adaptive communication built with PROTEUS. The heuristic-based communication vector selection policies we created for ACVS succeed in producing communication that is more easily perceived and understood than communication governed by a random vector selection policy. Furthermore, this improved quality of communication leads to improved task success rates in the study of ACVS we present. Neither PROTEUS nor ACVS could exist without the foundation of the previous two parts.

How Does This Thesis Reflect The Field?

When considered together, the methods presented in this thesis reflect the epochal shift that is beginning to happen in underwater robotics and underwater HRI in particular. This shift is comprised of three parts: a growth of applications, a much-needed fusion between two long-separate parts of the community, and an expansion in the focus of underwater HRI research from simple to complex research questions.

When it comes to the growth of applications of HRI, this is a long-term process that is finally coming to fruition. Roboticists and early masters of the field of HRI were responsible for bringing HRI to the public's attention and establishing the field. The creators of the PR2, Cynthia Breazeal, the Honda Robotics team which created ASIMO, and the organizers of the first RO-MAN and HRI conferences were among these early founders. In recent years, this field which started some 40 years ago (32

RO-MANs and 18 HRI's ago), has finally begun to become of importance in industry and more intensely studied in academia. Not only are humanoid robots reaching a point of commercialization that necessitates them having some form of HRI, but applications that would have been impossible to study (such as underwater HRI, for instance) are now being explored. This increase in applications is allowing the greater proliferation of HRI concepts and capabilities but is also intensifying the division in the field.

For many years, there have been two primary camps of researchers in the HRI field. There are researchers who publish primarily at RO-MAN and HRI, who focus on human studies and social impacts and somewhat less on engineering. Then, there are those who publish more at robotics (ICRA, IROS), vision (CVPR, IMCL), or learning (NeurIPS, CoRL) venues, who focus more on engineering questions and often do much less in the way of human studies. These two camps are not mutually exclusive. Both camps cross-publish and have skills in human studies as well as engineering, but the distribution of researchers centers around these two poles. This thesis is intended to be an example of the multidisciplinary work that should be the standard in the field. Additionally, it is an element of the work that must be done to form a consensus on methods of HRI research. Engineering is important, human studies are important, and both should be given adequate effort and time. The recommendations found in Appendix II are primarily focused on providing suggestions for improving the human research capabilities of a computer scientist such as myself, based on the learning to do. done towards ensuring that the methods presented in this thesis are rigorous and methodologically sound. This is an intentional inclusion, based on the philosophical opinion of the author that HRI research is at its best when it includes both humans and robots in equal measure in the methods and the work.

Finally, this thesis reflects the expansion of focus in UHRI research from first-level to second-level questions. The first two parts of this thesis deal with simple questions: “can motion be used to communicate from AUV to human” or “can an AUV approach a human effectively”, for example. We termed these topics first-order questions, as they focus on atomic capabilities and actions. In the third part of this thesis, we addressed a single second-order question: “How should an AUV adapt its interaction with a human for maximum interaction effectiveness?” Second-order questions go beyond the atomic actions of interaction and explore aspects of the interaction treated as a separate concept.

This shift from first to second-order researchers in UHRI has not been achieved without support. That is to say, the work of those who came before me in the field has enabled this thesis. Additionally, many first-order questions remain to be addressed and nothing in UHRI can truly be considered a solved problem at this point. However, due in part to the contributions of this thesis, we hope that further research can address second-order questions with greater ease, even as first-order research continues.

Final Thoughts

Long a domain of user-hostile robots and interfaces, marine robots are becoming inexpensive and highly capable, opening the domain to a large set of users who had never ventured below the waves. The researchers whose work inspired this thesis, including my advisor, saw this shift beginning in the early 2000s and began to explore the idea of building better user interfaces for AUVs. Their work produced a truly amazing set of capabilities for the time, which, in the inevitable march of computerized progress, have been antiquated in less than two decades. This work will likely last an even shorter time given the rate of progress in robotics and artificial intelligence, but I would be pleased if some aspects of the interaction paradigms remain. Along with the unprecedented sample sizes and complexities of the underwater human research studies herein, this thesis contains a wide variety of novel communication patterns which break significantly from the established norm. It is my hope that this thesis will help to usher in the next wave of AUVs: natural, multi-modal, adaptive, intelligent, and collaborative robotic companions underwater. In any event, the time of the co-AUV is coming, a time when the work that marine robotics researchers have been doing for decades will come to fruition, expanding humankind's mastery and understanding of our aquatic resources. As with all robotics research, it is important that we, the writer and the reader, remember that whether these robotic advances improve the world around us or simply increase the profits of the rich is not predetermined. We decide this. Let us work together to make sure that robots, even underwater robots, all do their part to usher in a better world for all, not simply a richer world for some.

Bibliography

- [1] Gregory Dudek, Philippe Giguère, Chris Prahacs, Shane Saunderson, Junaed Sattar, L. Abril Torres-Méndez, Michael Jenkin, Andrew German, Andrew Hogue, Arlene Ripsman, James Zacher, Evangelos Milios, Hui Liu, Pifu Zhang, Martin Buehler, and Christina Georgiades. AQUA: An amphibious autonomous robot. *Computer*, 40:46–53, 2007.
- [2] M. J. Islam, M. Ho, and J. Sattar. Dynamic Reconfiguration of Mission Parameters in Underwater Human-Robot Collaboration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2018.
- [3] Arturo Gomez Chavez, Christian A. Mueller, Andreas Birk, Anja Babic, and Nikola Miskovic. Stereo-vision based diver pose estimation using LSTM recurrent neural networks for AUV navigation guidance. *OCEANS 2017 - Aberdeen*, pages 1–7, June 2017. Conference Name: OCEANS 2017 - Aberdeen ISBN: 9781509052783 Place: Aberdeen, United Kingdom Publisher: IEEE.
- [4] J. Sattar, G. Dudek, O. Chiu, I. Rekleitis, P. Giguere, A. Mills, N. Plamondon, C. Prahacs, Y. Girdhar, M. Nahon, and J. Lobos. Enabling autonomous capabilities in underwater robotics. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3628–3634, 2008.
- [5] M. L. Walters, D. S. Syrdal, K. L. Koay, K. Dautenhahn, and R. te Boekhorst. Human approach distances to a mechanical-looking robot with different robot voice styles. In *RO-MAN 2008 - The 17th IEEE International Symposium on Robot and Human Interactive Communication*, pages 707–712, 2008. ISSN: 1944-9437.

- [6] J. Sattar and G. Dudek. Underwater human-robot interaction via biological motion identification. In *Robotics: Science and Systems V*. Robotics: Science and Systems Foundation, June 2009.
- [7] Xian Wu, Rachel E. Stuck, Ioannis Rekleitis, and Jenay M. Beer. Towards a Human Factors Model for Underwater Robotics. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts*, HRI'15 Extended Abstracts, pages 159–160. Association for Computing Machinery, 2015.
- [8] Arturo Gomez Chavez, Andrea Ranieri, Davide Chiarella, Enrica Zereik, Anja Babić, and Andreas Birk. CADDY Underwater Stereo-Vision Dataset for Human-Robot Interaction (HRI) in the Context of Diver Activities. *arXiv CS*, 2018, 1807.04856.
- [9] Nikola Mišković, Marco Bibuli, Andreas Birk, Massimo Caccia, Murat Egi, Karl Grammer, Alessandro Marroni, Jeff Neasham, Antonio Pascoal, Antonio Vasiljević, and others. CADDY—cognitive autonomous diving buddy: Two years of underwater human-robot interaction. *Marine Technology Society Journal*, 50(4):54–66, 2016.
- [10] G. Dudek, J. Sattar, and A. Xu. A Visual Language for Robot Control and Programming: A Human-Interface Study. In *Proceedings 2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2507–2513, 2007. ISSN: 1050-4729.
- [11] Arturo Gomez Chavez, Christian A. Mueller, Tobias Doernbach, Davide Chiarella, and Andreas Birk. Robust Gesture-Based Communication for Underwater Human-Robot Interaction in the context of Search and Rescue Diver Missions. *arXiv CS*, 2018, 1810.07122.
- [12] J. Sattar, E. Bourque, P. Giguere, and G. Dudek. Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction. In *Fourth Canadian Conference on Computer and Robot Vision (CRV '07)*, pages 165–174, 2007.

- [13] Md. Jahidul Islam, Michael Fulton, and Junaed Sattar. Toward a Generic Diver-Following Algorithm: Balancing Robustness and Efficiency in Deep Visual Detection. *IEEE Robotics and Automation Letters*, 4(1):113–120, 2019. Conference Name: IEEE Robotics and Automation Letters.
- [14] Đula Nadđ, Filip Mandić, and Nikola Mišković. Using Autonomous Underwater Vehicles for Diver Tracking and Navigation Aiding. *Journal of Marine Science and Engineering*, 8(6):413, June 2020. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.
- [15] Y. Ukai and J. Rekimoto. Swimoid: Interacting with an underwater buddy robot. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 423–423, 2013. ISSN: 2167-2148.
- [16] K. J. DeMarco, M. E. West, and A. M. Howard. Underwater human-robot communication: A case study with human divers. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3738–3743, 2014. ISSN: 1062-922X.
- [17] Bart Verzijlenberg and Michael Jenkin. Swimming with robots: Human robot communication at depth. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4023–4028, 2010. ISSN: 2153-0866.
- [18] Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie W. Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 2018.
- [19] G. C. Dismukes, V. V. Klimov, S. V. Baranov, Yu. N. Kozlov, J. DasGupta, and A. Tyryshkin. The origin of atmospheric oxygen on Earth: The innovation of oxygenic photosynthesis. *Proceedings of the National Academy of Sciences*, 98(5):2170–2175, February 2001. Publisher: Proceedings of the National Academy of Sciences.

- [20] Y. R. Petillot, S. R. Reed, and J. M. Bell. Real time AUV pipeline detection and tracking using side scan sonar and multi-beam echo-sounder. In *OCEANS '02 MTS/IEEE*, pages 217–222 vol.1, Oct 2002.
- [21] Brendan P. Foley, Katerina Dellaporta, Dimitris Sakellariou, Brian S. Bingham, Richard Camilli, Ryan M. Eustice, Dionysis Evangelistis, Vicki Lynn Ferrini, Kostas Katsaros, Dimitris Kourkoumelis, Aggelos Mallios, Paraskevi Micha, David A. Mindell, Christopher Roman, Hanumant Singh, David S. Switzer, and Theotokis Theodoulou. The 2005 Chios Ancient Shipwreck Survey: New Methods for Underwater Archaeology. *Hesperia: The Journal of the American School of Classical Studies at Athens*, 78(2):269–305, 2009.
- [22] Andreea Lup, Mihai Gorea, Denisa Bruhs, Oana Căpriă, and Patricia Vajda. Current perspectives on the remediation methods of marine plastic pollution: a review. *Studia Universitatis Babes-Bolyai, Biologia*, 65(2), 2020.
- [23] JW Nicholson and AJ Healey. The present state of autonomous underwater vehicle (auv) applications and technologies. *Marine Technology Society Journal*, 42(1):44–51, 2008.
- [24] Stefan B. Williams, Oscar Pizarro, Michael Jakuba, and Neville Barrett. AUV Benthic Habitat Mapping in South Eastern Tasmania. In Andrew Howard, Karl Iagnemma, and Alonzo Kelly, editors, *Field and Service Robotics*, pages 275–284, Berlin, Heidelberg, 2010. Springer.
- [25] N. Stilinović, Đ. Nađ, and N. Mišković. AUV for diver assistance and safety — Design and implementation. In *OCEANS 2015 - Genova*, pages 1–4, 2015.
- [26] Ammar Amory and Erik Maehle. SEMBIO - a small energy-efficient swarm AUV. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–7, 2016. ISSN: null.
- [27] Amy Underwood and Chris Murphy. Design of a micro-AUV for autonomy development and multi-vehicle systems. In *OCEANS 2017 - Aberdeen*, pages 1–6, 2017. ISSN: null.

- [28] Marc Carreras, Juan David Hernández, Eduard Vidal, Narcís Palomeras, David Ribas, and Pere Ridao. Sparus II AUV—A Hovering Vehicle for Seabed Inspection. *IEEE Journal of Oceanic Engineering*, 43(2):344–355, 2018.
- [29] Chelsey Edge, Sadman Sakib Enan, Michael Fulton, Jungseok Hong, Jiawei Mo, Kimberly Barthelemy, Hunter Bashaw, Berik Kallevig, Corey Knutson, Kevin Orpen, and Junaed Sattar. Design and Experiments with LoCO AUV: A Low Cost Open-Source Autonomous Underwater Vehicle. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1761–1768, 2020. ISSN: 2153-0866.
- [30] Salimzhan A. Gafurov and Evgeniy V. Klochkov. Autonomous Unmanned Underwater Vehicles Development Tendencies. *Procedia Engineering*, 106:141 – 148, 2015.
- [31] H. R. Widditsch. SPURV - The First Decade:. Technical report, Defense Technical Information Center, 1973.
- [32] W. Nodland, T. Ewart, W. Bendiner, J. Miller, and E. Aagaard. SPURV II-An Unmanned, Free-Swimming Submersible Developed for Oceanographic Research. In *OCEANS 81*, pages 92–98, 1981.
- [33] B. Allen, R. Stokey, T. Austin, N. Forrester, R. Goldsborough, M. Purcell, and C. von Alt. REMUS: a small, low cost AUV; system description, field trials and performance results. In *Oceans '97. MTS/IEEE Conference Proceedings*, volume 2, pages 994–1000 vol.2, 1997.
- [34] R. P. Stokey, A. Roup, C. von Alt, B. Allen, N. Forrester, T. Austin, R. Goldsborough, M. Purcell, F. Jaffre, G. Packard, and A. Kukulya. Development of the REMUS 600 autonomous underwater vehicle. In *Proceedings of OCEANS 2005 MTS/IEEE*, pages 1301–1304 Vol. 2, 2005.
- [35] C. C. Eriksen, T. J. Osse, R. D. Light, T. Wen, T. W. Lehman, P. L. Sabin, J. W. Ballard, and A. M. Chiodi. Seaglider: a long-range autonomous underwater vehicle for oceanographic research. *IEEE Journal of Oceanic Engineering*, 26(4):424–436, 2001.

- [36] T. J. Osse and C. C. Eriksen. The Deepglider: A Full Ocean Depth Glider for Oceanographic Research. In *OCEANS 2007*, pages 1–12, 2007.
- [37] Axel Hackbart, Edwin Kreuzer, and Eugen Solowjow. HippoCampus: A micro underwater vehicle for swarm applications. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2258–2263, 2015. ISSN: null.
- [38] Carlos S. Gonçalves, Bruno M. Ferreira, and Aníbal C. Matos. Design and development of SHAD - a Small Hovering AUV with Differential actuation. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–4, 2016. ISSN: null.
- [39] Akihiro Okamoto, Kenkichi Tamura, Masahiko Sasano, Kenichi Sawada, Takahiro Seta, Shogo Inaba, Tamaki Ura, Yuya Nishida, Junichi Kojima, and Yuzuru Itoh. Development of hovering-type AUV “HOBALIN” for exploring seafloor hydrothermal deposits. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–4, 2016. ISSN: null.
- [40] Junhao Wu, Shilin Peng, Teqi Xu, Raofeng Hu, Shuo Wang, Mian Pan, and Xiaowei Weng. Test Bed AUV for Docking Algorithm Research. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–6, 2018. ISSN: 0197-7385.
- [41] Jaime Lloret, Sandra Sendra, Miguel Ardid, and Joel J. P. C. Rodrigues. Underwater Wireless Sensor Communications in the 2.4 GHz ISM Frequency Band. *Sensors (Basel, Switzerland)*, 12(4):4237–4264, 2012.
- [42] Andreas Birk. A Survey of Underwater Human-Robot Interaction (U-HRI). *Current Robotics Reports*, 3(4):199–211, December 2022.
- [43] Robert K. Katzschmann, Joseph DelPreto, Robert MacCurdy, and Daniela Rus. Exploration of underwater life with an acoustically controlled soft robotic fish. *Science Robotics*, 3(16), 2018.
- [44] J. Sattar and J. J. Little. Ensuring safety in human-robot dialog — A cost-directed approach. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6660–6666, 2014.

- [45] Anqi Xu, Gregory Dudek, and Junaed Sattar. A natural gesture interface for operating robotic systems. In *2008 IEEE International Conference on Robotics and Automation*, pages 3557–3563, May 2008. ISSN: 1050-4729.
- [46] Michael Fulton, Chelsey Edge, and Junaed Sattar. Robot Communication Via Motion: A Study on Modalities for Robot-to-Human Communication in the Field. *ACM Transactions on Human-Robot Interaction*, 11(2):1–40, 2022.
- [47] Mark L. Knapp, Judith A. Hall, and Terrence G. Horgan. *Nonverbal Communication in Human Interaction*. Cengage Learning, 2013.
- [48] Guy Hoffman and Cynthia Breazeal. Collaboration in Human-Robot Teams. In *AIAA 1st Intelligent Systems Technical Conference*. American Institute of Aeronautics and Astronautics, 2004.
- [49] John-John Cabibihan, Wing-Chee So, and Soumo Pramanik. Human-Recognizable Robotic Gestures. *IEEE Transactions on Autonomous Mental Development*, 4(4):305–314, December 2012. Conference Name: IEEE Transactions on Autonomous Mental Development.
- [50] Chien-Ming Huang and Bilge Mutlu. Modeling and Evaluating Narrative Gestures for Humanlike Robots. In *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation, June 2013.
- [51] Cynthia L. Breazeal. *Designing Sociable Robots*. MIT Press, 2004.
- [52] Chaona Chen, Oliver G.B. Garrod, Jiayu Zhan, Jonas Beskow, Philippe G. Schyns, and Rachael E. Jack. Reverse Engineering Psychologically Valid Facial Expressions of Emotion into Social Robots. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 448–452, May 2018.
- [53] Mauricio E Reyes, Ivan V Meza, and Luis A Pineda. Robotics facial expression of anger in collaborative human–robot interaction. *International Journal of Advanced Robotic Systems*, 16(1):1729881418817972, January 2019. Publisher: SAGE Publications.
- [54] Pepper the humanoid and programmable robot | SoftBank robotics.

- [55] Zhijun Zhang, Yaru Niu, Shangen Wu, Shuyang Lin, and Lingdong Kong. Analysis of Influencing Factors on Humanoid Robots' Emotion Expressions by Body Language. In Tingwen Huang, Jiancheng Lv, Changyin Sun, and Alexander V. Tuzikov, editors, *Advances in Neural Networks – ISNN 2018*, Lecture Notes in Computer Science, pages 775–785, Cham, 2018. Springer International Publishing.
- [56] Cindy L Bethel. *Dissertation: Robots without faces: Non-verbal social human-robot interaction*. University of South Florida, 2009.
- [57] C. L. Bethel and R. R. Murphy. Survey of Non-facial/Non-verbal Affective Expressions for Appearance-Constrained Robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(1):83–92, 2008.
- [58] Cindy L. Bethel and Robin R. Murphy. Non-Facial and Non-Verbal Affective Expression for Appearance-Constrained Robots Used in Victim Management*. *Paladyn, Journal of Behavioral Robotics*, 1(4):219–230, 2011.
- [59] L. Moshkina and R. C. Arkin. Human perspective on affective robotic behavior: a longitudinal study. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1444–1451, 2005.
- [60] T. Shimokawa and T. Sawaragi. Acquiring communicative motor acts of social robot using interactive evolutionary computation. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*, volume 3, pages 1396–1401 vol.3, 2001.
- [61] Joshua Wainer, David J. Feil-seifer, Dylan A. Shell, and Maja J. Mataric. The role of physical embodiment in human-robot interaction. In *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication*, pages 117–122, 2006. ISSN: 1944-9437.
- [62] Justin Hart, Reuth Mirsky, Xuesu Xiao, Stone Tejeda, Bonny Mahajan, Jamin Goo, Kathryn Baldauf, Sydney Owen, and Peter Stone. Using human-inspired signals to disambiguate navigational intentions. In *Proceedings of the 12th International Conference on Social Robotics (ICSR)*. Springer Publishing, 2020.

- [63] Rachel Holladay, Anca Dragan, and Siddhartha Srinivasa. Legible Robot Pointing. In *Legible Robot Pointing*, volume 2014, 2014.
- [64] Sandy H. Huang, Isabella Huang, Ravi Pandya, and Anca D. Dragan. Nonverbal robot feedback for human teachers. In *Conference on Robot Learning*, 2019.
- [65] A. Zhou, D. Hadfield-Menell, A. Naaabandi, and A. D. Dragan. Expressive robot motion timing. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 22–31, 2017.
- [66] Anca D. Dragan, Shira Bauman, Jodi Forlizzi, and Siddhartha S. Srinivasa. Effects of robot motion on human-robot collaboration. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 51–58, 2015. ISSN: 2167-2121.
- [67] Daniel Szafir, Bilge Mutlu, and Terrence Fong. Communication of intent in assistive free flyers. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, HRI ’14, pages 358–365. Association for Computing Machinery, 2014.
- [68] B. A. Duncan, E. Beachly, A. Bevins, S. Elbaum, and C. Detweiler. Investigation of Communicative Flight Paths for Small Unmanned Aerial Systems * This work was supported by NSF NRI 1638099. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 602–609, 2018.
- [69] J. R. Cauchard, K. Y. Zhai, M. Spadafora, and J. A. Landay. Emotion encoding in Human-Drone Interaction. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 263–270, 2016.
- [70] Dante Arroyo, Cesar Lucho, Silvia Julissa Roncal, and Francisco Cuellar. Daedalus: A sUAV for Human-robot Interaction. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, HRI ’14, pages 116–117. ACM, 2014.
- [71] Rachel Holladay, Anca Dragan, and Siddhartha Srinivasa. Legible robot pointing. In *Legible Robot Pointing*, volume 2014, 08 2014.

- [72] Elena Andonova and Holly A Taylor. Nodding in dis/agreement: a tale of two cultures. *Cognitive processing*, 13(1):79–82, 2012.
- [73] Norman S. Nise. *Control Systems Engineering*. Wiley, 7 edition, 2015.
- [74] Klaus Krippendorff. Agreement and Information in the Reliability of Coding. *Communication Methods and Measures*, 5(2):93–112, 2011.
- [75] Henry B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.
- [76] Carlo Bonferroni. Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62, 1936.
- [77] John Brown. Some tests of the decay theory of immediate memory. *Quarterly Journal of Experimental Psychology*, 10(1):12–21, 1958, <https://doi.org/10.1080/17470215808416249>.
- [78] Otto Waris, Anna Soveri, Miikka Ahti, Russell Hoffing, Daniel Ventus, Susanne Jaeggi, Aaron Seitz, and Matti Laine. A latent factor analysis of working memory measures using large-scale data. *Frontiers in Psychology*, 8, 2017.
- [79] S. S. Shapiro and M. B. Wilk. An Analysis of Variance Test for Normality (Complete Samples). *Biometrika*, 52(3/4):591–611, 1965. Publisher: [Oxford University Press, Biometrika Trust].
- [80] William H. Kruskal and W. Allen Wallis. Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952. Publisher: [American Statistical Association, Taylor & Francis, Ltd.].
- [81] Amit Kumar Pandey and Rodolphe Gelin. A Mass-Produced Sociable Humanoid Robot: Pepper: The First Machine of Its Kind. *IEEE Robotics & Automation Magazine*, 25(3):40–48, 2018. Conference Name: IEEE Robotics & Automation Magazine.

- [82] JingGuang Han, Nick Campbell, Kristiina Jokinen, and Graham Wilcock. Investigating the use of Non-verbal Cues in Human-Robot Interaction with a Nao robot. In *2012 IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 679–683, 2012.
- [83] Kotaro Funakoshi, Kazuki Kobayashi, Mikio Nakano, Seiji Yamada, Yasuhiko Kitamura, and Hiroshi Tsujino. Smoothing human-robot speech interactions by using a blinking-light as subtle expression. In *Proceedings of the 10th International Conference on Multimodal Interfaces*, ICMI '08, pages 293–296. Association for Computing Machinery, 2008.
- [84] Mingyu Kim, Hui Sung Lee, Jeong Woo Park, Su Hun Jo, and Myung Jin Chung. Determining color and blinking to support facial expression of a robot for conveying emotional intensity. In *Determining color and blinking to support facial expression of a robot for conveying emotional intensity*, 2008.
- [85] Daniel Szafir, Bilge Mutlu, and Terry Fong. Communicating Directionality in Flying Robots. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '15, pages 19–26. Association for Computing Machinery, 2015.
- [86] Kim Baraka, Stephanie Rosenthal, and Manuela Veloso. Enhancing human understanding of a mobile robot’s state and actions using expressive lights. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 652–657, 2016. ISSN: 1944-9437.
- [87] Kim Baraka and Manuela M. Veloso. Mobile Service Robot State Revealing Through Expressive Lights: Formalism, Design, and Evaluation. *International Journal of Social Robotics*, 10(1):65–92, 2018.
- [88] Sichao Song and Seiji Yamada. Bioluminescence-Inspired Human-Robot Interaction : Designing Expressive Lights that Affect Human’s Willingness to Interact with a Robot. In *2018 13th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 224–232, 2018. ISSN: 2167-2148.

- [89] Sichao Song and Seiji Yamada. Effect of Expressive Lights on Human Perception and Interpretation of Functional Robot. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, pages 1–6. Association for Computing Machinery, 2018.
- [90] George Butterworth and Nicholas Jarrett. What minds have in common is space: Spatial mechanisms serving joint visual attention in infancy. *British Journal of Developmental Psychology*, 9(1):55–72, 1991.
- [91] Debra Zeifman, Sarah Delaney, and Elliott M. Blass. Sweet taste, looking, and calm in 2- and 4-week-old infants: The eyes have it. *Developmental Psychology*, 32(6):1090–1099, 1996.
- [92] Jonathan S. Beier and Elizabeth S. Spelke. Infants’ Developing Understanding of Social Gaze. *Child Development*, 83(2):486–496, 2012.
- [93] Nikolaos Mavridis. A review of verbal and non-verbal human–robot interactive communication. *Robotics and Autonomous Systems*, 63:22–35, 2015.
- [94] Brian Scassellati. Imitation and Mechanisms of Joint Attention: A Developmental Structure for Building Social Skills on a Humanoid Robot. In *Computation for Metaphors, Analogy, and Agents*, volume 1562, pages 176–195. Springer Berlin Heidelberg, 1999.
- [95] B Scassellati. Mechanisms of Shared Attention for a Humanoid Robot. *Embodied Cognition and Action: Papers from the 1996 AAAI Fall Symposium*, page 5, 1996.
- [96] Henny Admoni, Anca Dragan, Siddhartha S. Srinivasa, and Brian Scassellati. Deliberate delays during robot-to-human handovers improve compliance with gaze communication. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, HRI ’14, pages 49–56. Association for Computing Machinery, 2014.
- [97] AJung Moon, Daniel M. Troniak, Brian Gleeson, Matthew K.X.J. Pan, Minhua Zheng, Benjamin A. Blumer, Karon MacLean, and Elizabeth A. Croft. Meet me where I’m gazing: how shared attention gaze affects human-robot handover timing.

- In *Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '14, pages 334–341. Association for Computing Machinery, 2014.
- [98] C. Breazeal, C.D. Kidd, A.L. Thomaz, G. Hoffman, and M. Berlin. Effects of nonverbal communication on efficiency and robustness in human-robot teamwork. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 708–713, 2005. ISSN: 2153-0866.
 - [99] Bilge Mutlu, Toshiyuki Shiwa, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. Footing in human-robot conversations: How robots might shape participant roles using gaze cues. In *2009 4th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 61–68, 2009. ISSN: 2167-2148.
 - [100] Vijay Chidambaram, Yueh-Hsuan Chiang, and Bilge Mutlu. Designing persuasive robots: How robots might persuade people using vocal and nonverbal cues. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 293–300, 2012. ISSN: 2167-2148.
 - [101] C. Breazeal. Social interactions in HRI: the robot view. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2):181–186, 2004. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).
 - [102] Tomomi Onuki, Takafumi Ishinoda, Yoshinori Kobayashi, and Yoshinori Kuno. Design of robot eyes suitable for gaze communication. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 203–204, 2013. ISSN: 2167-2148.
 - [103] Simon Schulz, Sebastian Meyer zu Borgsen, and Sven Wachsmuth. See and Be Seen – Rapid and Likeable High-Definition Camera-Eye for Anthropomorphic Robots. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2524–2530, year. ISSN: 2577-087X.

- [104] Davide Chiarella, Marco Bibuli, Gabriele Bruzzone, Massimo Caccia, Andrea Ranieri, Enrica Zereik, Lucia Marconi, and Paola Cutugno. A Novel Gesture-Based Language for Underwater Human–Robot Interaction. *Journal of Marine Science and Engineering*, 6(3):91, 2018.
- [105] Michael Fulton, Chelsey Edge, and Junaed Sattar. Robot Communication Via Motion: Closing the Underwater Human-Robot Interaction Loop. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4660–4666, 2019. ISSN: 2577-087X.
- [106] Michael Fulton, Muntaqim Mehtaz, Junaed Sattar, and Owen Queeglay. Underwater Robot-To-Human Communication Via Motion: Implementation and Full-Loop Human Interface Evaluation. In *Robotics: Science and Systems XVIII*. Robotics: Science and Systems Foundation, 2022.
- [107] Christoph Bartneck, Dana Kulić, Elizabeth Croft, and Susana Zoghbi. Measurement Instruments for the Anthropomorphism, Animacy, Likeability, Perceived Intelligence, and Perceived Safety of Robots. *International Journal of Social Robotics*, 1(1):71–81, 2009.
- [108] Astrid Weiss and Christoph Bartneck. Meta analysis of the usage of the Godspeed Questionnaire Series. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 381–388, 2015.
- [109] Sandra G. Hart and Lowell E. Staveland. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In Peter A. Hancock and Najmedin Meshkati, editors, *Advances in Psychology*, volume 52 of *Human Mental Workload*, pages 139–183. North-Holland, 1988.
- [110] Jan M. Noyes and Daniel P. J. Bruneau. A self-analysis of the NASA-TLX workload measure. *Ergonomics*, 50(4):514–519, 2007.
- [111] Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.
- [112] J. Richard Landis and Gary G. Koch. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174, 1977.

- [113] Jennifer Birch. Efficiency of the Ishihara test for identifying red-green colour deficiency. *Ophthalmic and Physiological Optics*, 17(5):403–408, 1997.
- [114] Robert L. Adams. Can You Communicate? Underwater? In *Can You Communicate? Underwater?* OnePetro, 1971.
- [115] Meera M. Blattner, Denise A. Sumikawa, and Robert M. Greenberg. Earcons and icons: Their structure and common design principles. *Human–Computer Interaction*, 4(1):11–44, 1989, https://www.tandfonline.com/doi/pdf/10.1207/s15327051hci0401_1.
- [116] Michael Fulton, Aditya Prabhu, and Junaed Sattar. Hreyes: Design, development, and evaluation of a novel method for auvs to communicate information and gaze direction.
- [117] Andrea Bonarini. Communication in Human-Robot Interaction. *Current Robotics Reports*, 1(4):279–285, 2020.
- [118] Cynthia Breazeal. Toward sociable robots. *Robotics and Autonomous Systems*, 42(3):167–175, 2003.
- [119] Terrence Fong, Illah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3):143–166, 2003.
- [120] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter*, 19(2):25–35, 2017, 1711.01731 [cs].
- [121] I. R. Murray and J. L. Arnott. Toward the simulation of emotion in synthetic speech: a review of the literature on human vocal emotion. *The Journal of the Acoustical Society of America*, 93(2):1097–1108, 1993.
- [122] Joe Crumpton and Cindy L Bethel. A survey of using vocal prosody to convey emotion in robot speech. *International Journal of Social Robotics*, 8(2):271–285, 2016.
- [123] Friederike Eyssel, Laura de Ruiter, Dieta Kuchenbrandt, Simon Bobinger, and Frank Hegel. ‘If you sound like me, you must be more human’: On the interplay

- of robot and user features on human-robot acceptance and anthropomorphism. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 125–126, 2012. ISSN: 2167-2148.
- [124] Hamish Tennent, Dylan Moore, Malte Jung, and Wendy Ju. Good vibrations: How consequential sounds affect perception of robotic arms. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 928–935, 2017. ISSN: 1944-9437.
 - [125] Dylan Moore, Hamish Tennent, Nikolas Martelaro, and Wendy Ju. Making Noise Intentional: A Study of Servo Sound Perception. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI ’17, pages 12–21. Association for Computing Machinery, 2017.
 - [126] Frederic Anthony Robinson, Mari Velonaki, and Oliver Bown. Smooth Operator: Tuning Robot Perception Through Artificial Movement Sound. In *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, HRI ’21, pages 53–62. Association for Computing Machinery, 2021.
 - [127] Elizabeth Cha, Naomi T. Fitter, Yunkyoung Kim, Terrence Fong, and Maja J. Matarić. Effects of Robot Sound on Auditory Localization in Human-Robot Collaboration. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, HRI ’18, pages 434–442. Association for Computing Machinery, 2018.
 - [128] Eun-Sook Jee, Yong-Jeon Jeong, Chong Hui Kim, and Hisato Kobayashi. Sound design for emotion and intention expression of socially interactive robots. *Intelligent Service Robotics*, 3(3):199–206, 2010.
 - [129] Markus Schwenk and Kai O. Arras. R2-D2 Reloaded: A flexible sound synthesis system for sonic human-robot interaction design. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 161–167, 2014. ISSN: 1944-9437.
 - [130] Dayton audio - DAEX25w-8 waterproof 25mm exciter 10w 8 ohm.

- [131] Erik Nyquist. eriknyquist/tones. original-date: 2018-07-29T01:19:54Z.
- [132] hadware. Voxpopuli. original-date: 2017-01-22T18:09:56Z.
- [133] eSpeak: Speech synthesizer.
- [134] T. Dutoit, V. Pagel, N. Pierret, F. Bataille, and O. van der Vrecken. The MBROLA project: towards a set of high quality speech synthesizers free of use for non commercial purposes. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP '96*, volume 3, pages 1393–1396 vol.3, 1996.
- [135] Charles Edward Spearman. *The proof and measurement of association between two things*. [n.p.], 1904.
- [136] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. Publisher: [Board of the Foundation of the Scandinavian Journal of Statistics, Wiley].
- [137] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019.
- [138] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.
- [139] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.
- [140] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.
- [141] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

- [142] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 379–387, 2016.
- [143] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.
- [144] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.
- [145] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.
- [146] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning Research*, 20(184):1–25, 2019.
- [147] Vaishaal Shankar, Achal Dave, Rebecca Roelofs, Deva Ramanan, Benjamin Recht, and Ludwig Schmidt. Do image classifiers generalize across time? *arXiv preprint arXiv:1906.02168*, 2019.
- [148] Keren Gu, Brandon Yang, Jiquan Ngiam, Quoc Le, and Jonathon Shlens. Using videos to evaluate image model robustness. *arXiv preprint arXiv:1904.10076*, 2019.
- [149] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *arXiv preprint arXiv:2007.00644*, 2020.
- [150] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

- [151] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [152] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pages 1802–1811, 2019.
- [153] Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4845–4854, 2019.
- [154] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.
- [155] Mason Liu and Menglong Zhu. Mobile video object detection with temporally-aware feature maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5686–5695, 2018.
- [156] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3038–3046, 2017.
- [157] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 408–417, 2017.
- [158] Gedas Bertasius, Lorenzo Torresani, and Jianbo Shi. Object detection in video with spatiotemporal sampling networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 331–346, 2018.
- [159] Mason Liu, Menglong Zhu, Marie White, Yinxiao Li, and Dmitry Kalenichenko. Looking fast and slow: Memory-guided mobile video object detection. *arXiv preprint arXiv:1903.10172*, 2019.

- [160] Yihong Chen, Yue Cao, Han Hu, and Liwei Wang. Memory enhanced global-local aggregation for video object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10337–10346, 2020.
- [161] Xizhou Zhu, Jifeng Dai, Lu Yuan, and Yichen Wei. Towards high performance video object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7210–7218, 2018.
- [162] Hong Zhang and Naiyan Wang. On the stability of video detection and tracking. *arXiv preprint arXiv:1903.10172*, 2017.
- [163] Xingyu Chen, Zhengxing Wu, Junzhi Yu, and Li Wen. Rethinking temporal object detection from robotic perspectives. *arXiv preprint arXiv:1912.10406*, 2020, 1912.10406.
- [164] Duc Thanh Nguyen, Wanqing Li, and Philip O. Ogunbona. Human detection from images and videos: A survey. *Pattern Recognition*, 51:148–175, March 2016.
- [165] Md Jahidul Islam, Jungseok Hong, and Junaed Sattar. Person-following by autonomous robots: A categorical overview. *The International Journal of Robotics Research*, 38(14):1581–1618, 2019.
- [166] K. J. DeMarco, M. E. West, and A. M. Howard. Sonar-Based Detection and Tracking of a Diver for Underwater Human-Robot Interaction Scenarios. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2378–2383, 2013.
- [167] Nabin S. Sharma, Alexander M. Yakubovskiy, and Matthew J. Zimmerman. SCUBA diver detection and classification in active and passive sonars — A unified approach. In *2013 IEEE International Conference on Technologies for Homeland Security (HST)*, pages 189–194, November 2013.
- [168] Kil Woo Chung, Hongbin Li, and Alexander Sutin. A Frequency-Domain Multi-Band Matched-Filter Approach to Passive Diver Detection. In *2007 Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers*, pages 1252–1256, November 2007. ISSN: 1058-6393.

- [169] Anton Molyboha and Michael Zabarankin. Stochastic Optimization of Sensor Placement for Diver Detection. *Operations Research*, 60(2):292–312, April 2012. Publisher: INFORMS.
- [170] Hannan Lohrasbippeydeh, Tom Dakin, T. Aaron Gulliver, and Claire De Grasse. Passive energy based acoustic signal analysis for diver detection. In *2014 Oceans - St. John's*, pages 1–5, September 2014. ISSN: 0197-7385.
- [171] Qiang Tu, Fei Yuan, Weidi Yang, and En Cheng. An Approach for Diver Passive Detection Based on the Established Model of Breathing Sound Emission. *Journal of Marine Science and Engineering*, 8(1):44, January 2020. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.
- [172] Junaed Sattar and Gregory Dudek. Robust servo-control for underwater robots using banks of visual filters. In *2009 IEEE International Conference on Robotics and Automation*, pages 3583–3588, May 2009. ISSN: 1050-4729.
- [173] Heiko Buelow and Andreas Birk. Diver detection by motion-segmentation and shape-analysis from a moving vehicle. In *OCEANS'11 MTS/IEEE KONA*, pages 1–7, September 2011. ISSN: 0197-7385.
- [174] Kevin J. DeMarco, Michael E. West, and Ayanna M. Howard. Autonomous robot-diver assistance through joint intention theory. In *2014 Oceans - St. John's*, pages 1–5, September 2014. ISSN: 0197-7385.
- [175] Junaed Sattar and Gregory Dudek. Where is your dive buddy: tracking humans underwater using spatio-temporal features. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3654–3659, October 2007. Conference Name: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems ISBN: 9781424409112 9781424409129 Place: San Diego, CA, USA Publisher: IEEE.
- [176] Md Jahidul Islam and Junaed Sattar. Mixed-domain biological motion tracking for underwater human-robot interaction. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4457–4464, 2017. ISSN: null.

- [177] A. G. Chavez, C. A. Mueller, A. Birk, A. Babic, and N. Miskovic. Stereo-vision based diver pose estimation using LSTM recurrent neural networks for AUV navigation guidance. In *OCEANS 2017 - Aberdeen*, pages 1–7, 2017.
- [178] Arturo Gomez Chavez, Max Pfingsthorn, Andreas Birk, Ivor Rendulić, and Nikola Misković. Visual diver detection using multi-descriptor nearest-class-mean random forests in the context of underwater Human Robot Interaction (HRI). In *OCEANS 2015 - Genova*, pages 1–7, May 2015.
- [179] Igor Kvasić, Nikola Mišković, and Zoran Vukić. Convolutional Neural Network Architectures for Sonar-Based Diver Detection and Tracking. In *OCEANS 2019 - Marseille*, pages 1–6, June 2019.
- [180] Youya Xia and Junaed Sattar. Visual Diver Recognition for Underwater Human-Robot Collaboration. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6839–6845, May 2019. ISSN: 2577-087X.
- [181] Karin de Langis and Junaed Sattar. Realtime Multi-Diver Tracking and Re-identification for Underwater Human-Robot Collaboration. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11140–11146, May 2020. ISSN: 2577-087X.
- [182] Ericsson Open Source Software. Github - ericsson/eva. <https://github.com/Ericsson/eva>. (Accessed on 10/26/2020).
- [183] João F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *Proceedings of the 12th European Conference on Computer Vision (ECCV)*, page 702–715, Berlin, Heidelberg, 2012.
- [184] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.
- [185] TensorFlow. *tf.train.Example*. <https://bit.ly/3j06tuR>. Accessed 10-31-2020.

- [186] TensorFlow. *tf.train.SequenceExample*. <https://bit.ly/3oMDdYX>. Accessed 10-31-2020.
- [187] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.
- [188] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7310–7311, 2017.
- [189] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4278–4284, 2017.
- [190] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.
- [191] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017, 1704.04861.
- [192] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.
- [193] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint ArXiv:2004.10934*, 2020, 2004.10934.

- [194] Mason Liu and Menglong Zhu. Mobile video object detection with temporally-aware feature maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5686–5695, 2018.
- [195] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.
- [196] Hong Zhang and Naiyan Wang. On the stability of video detection and tracking. *arXiv preprint arXiv:1903.10172*, 2017.
- [197] Xingyu Chen, Zhengxing Wu, Junzhi Yu, and Li Wen. Rethinking temporal object detection from robotic perspectives. *arXiv preprint arXiv:1912.10406*, 2020, 1912.10406.
- [198] Xingyu Chen, Junzhi Yu, and Zhengxing Wu. Temporally identity-aware SSD with attentional LSTM. *IEEE Transactions on Cybernetics*, 50(6):2674–2686, 2019.
- [199] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.
- [200] Payam Nikdel, Rakesh Shrestha, and Richard Vaughan. The Hands-Free Push-Cart: Autonomous Following in Front by Predicting User Trajectory Around Obstacles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4548–4554, 2018. ISSN: 2577-087X.
- [201] Geoff Nagy and Richard Vaughan. Flying Face Engagement: Aligning a UAV to Directly Face a Moving Uninstrumented User.
- [202] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. In *Physical Review E*, volume 51, pages 4282–4286, May 1995.
- [203] S. Pellegrinelli, H. Admoni, S. Javdani, and S. Srinivasa. Human-robot shared workspace collaboration via hindsight optimization. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 831–838, 2016.

- [204] H. S. Koppula and A. Saxena. Anticipating Human Activities Using Object Affordances for Reactive Robotic Response. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):14–29, 2016.
- [205] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *European Conference on Computer Vision*, pages 201–214. Springer, 2012.
- [206] Haifeng Gong, Jack Sim, M. Likhachev, and Jianbo Shi. Multi-hypothesis motion planning for visual object tracking. In *2011 International Conference on Computer Vision*, pages 619–626, 2011.
- [207] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by hierarchical association of detection responses. In *European Conference on Computer Vision*, pages 788–801. Springer, 2008.
- [208] MarcAurelio Ranzato, Arthur Szlam, Joan Bruna, Michael Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014.
- [209] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.
- [210] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with LSTM. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855, 1999.
- [211] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You’ll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 261–268, 2009.
- [212] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. *Computer graphics forum*, 26(3):655–664, 2007.
- [213] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

- [214] Christopher Brandl, Alexander Mertens, and Christopher Schlick. Human-Robot Interaction in Assisted Personal Services: Factors Influencing Distances That Humans Will Accept between Themselves and an Approaching Service Robot: Human-Robot Interaction in Assisted Personal Services. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 26, 2016.
- [215] K. Dautenhahn, M. Walters, S. Woods, K. L. Koay, C. L. Nehaniv, A. Sisbot, R. Alami, and T. Siméon. How may I serve you? a robot companion approaching a seated person in a helping context. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, HRI '06, pages 172–179. Association for Computing Machinery, 2006.
- [216] D. Syrdal, K. Dautenhahn, S. Woods, M. L. Walters, and K. Koay. 'Doing the right thing wrong' - Personality and tolerance to uncomfortable robot approaches. *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication*, 2006.
- [217] Leila Takayama and Caroline Pantofaru. Influences on proxemic behaviors in human-robot interaction. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5495–5502. IEEE, 2009.
- [218] 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 *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4410–4417, 2016.
- [219] Walid Remmas, Ahmed Chemori, and Maarja Kruusmaa. Diver tracking in open waters: A low-cost approach based on visual and acoustic sensor fusion. *Journal of Field Robotics*, 38(3):494–508, 2021, <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21999>.
- [220] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

- [221] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 593–600, 2005.
- [222] David Gallup, Jan-Michael Frahm, Philippas Mordohai, and Marc Pollefeys. Variable baseline/resolution stereo. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.
- [223] https://github.com/NVIDIA-AI-IOT/trt_pose.
- [224] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.
- [225] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 466–481, 2018.
- [226] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 593–600, 2005.
- [227] David Gallup, Jan-Michael Frahm, Philippas Mordohai, and Marc Pollefeys. Variable baseline/resolution stereo. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.
- [228] Margaret McDowell, Cheryl Fryar, and Cynthia Ogden. Anthropometric reference data for children and adults: United states, 2003–2006. *Vital and health statistics. Series 11, Data from the national health survey*, 10:1–68, 05 2009.
- [229] Nathan Koenig and Andrew Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154, 2004.
- [230] Maria Eugenia Cabrera and Juan Pablo Wachs. A Human-Centered Approach to One-Shot Gesture Learning. *Frontiers in Robotics and AI*, 4, 2017.

- [231] Razieh Rastgoo, Kourosh Kiani, and Sergio Escalera. Sign Language Recognition: A Deep Survey. *Expert Systems with Applications*, 164:113794, February 2021.
- [232] Munir Oudah, Ali Al-Naji, and Javaan Chahl. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques. *Journal of Imaging*, 6(8):73, August 2020. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute.
- [233] Jesus Suarez and Robin R. Murphy. Hand gesture recognition with depth images: A review. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, pages 411–417, September 2012. ISSN: 1944-9437.
- [234] Laura Dipietro, Angelo M. Sabatini, and Paolo Dario. A Survey of Glove-Based Systems and Their Applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(4):461–482, July 2008. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).
- [235] Derek W. Orbaugh Antillon, Christopher Walker, Samuel Rosset, and Iain A. Anderson. The challenges of hand gesture recognition using dielectric elastomer sensors. In *Electroactive Polymer Actuators and Devices (EAPAD) XXII*, volume 11375, pages 231–241. SPIE, May 2020.
- [236] Sushmita Mitra and Tinku Acharya. Gesture Recognition: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, May 2007. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).
- [237] Lingchen Chen, Feng Wang, Hui Deng, and Kaifan Ji. A Survey on Hand Gesture Recognition. In *2013 International Conference on Computer Sciences and Applications*, pages 313–316, December 2013.
- [238] Harpreet Kaur and Jyoti Rani. A review: Study of various techniques of Hand gesture recognition. In *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, pages 1–5, July 2016.

- [239] Maria Cabrera Ubaldi. Gist of a Gest: Learning Gestures for the First Time. *Open Access Dissertations*, August 2018.
- [240] Md Jahidul Islam, Marc Ho, and Junaed Sattar. Understanding human motion and gestures for underwater human–robot collaboration. *Journal of Field Robotics*, 36(5):851–873, 2019. _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21837>.
- [241] D. Chiarella, M. Bibuli, G. Bruzzone, M. Caccia, A. Ranieri, E. Zereik, L. Marconi, and P. Cutugno. Gesture-based language for diver-robot underwater interaction. In *OCEANS 2015 - Genova*, pages 1–9, May 2015.
- [242] Nikola Mišković, Marco Bibuli, Andreas Birk, Massimo Caccia, Murat Egi, Karl Grammer, Alessandro Marroni, Jeff Neasham, Antonio Pascoal, Antonio Vasiljević, et al. Caddy—cognitive autonomous diving buddy: Two years of underwater human-robot interaction. *Marine Technology Society Journal*, 50(4):54–66, 2016.
- [243] Arturo Gomez Chavez, Andrea Ranieri, Davide Chiarella, and Andreas Birk. Underwater Vision-Based Gesture Recognition: A Robustness Validation for Safe Human–Robot Interaction. *IEEE Robotics & Automation Magazine*, 28(3):67–78, September 2021. Conference Name: IEEE Robotics & Automation Magazine.
- [244] Mygel Andrei Martija, Jakov Ivan Dumbrique, and Prospero Naval. Underwater Gesture Recognition Using Classical Computer Vision and Deep Learning Techniques. *Mathematics Faculty Publications*, March 2020.
- [245] Yu Jiang, Minghao Zhao, Chong Wang, Fenglin Wei, Kai Wang, and Hong Qi. Diver’s hand gesture recognition and segmentation for human–robot interaction on AUV. *Signal, Image and Video Processing*, 15(8):1899–1906, November 2021.
- [246] Yu Jiang, Xianglong Peng, Mingzhu Xue, Chong Wang, and Hong Qi. An Underwater Human–Robot Interaction Using Hand Gestures for Fuzzy Control. *International Journal of Fuzzy Systems*, 23(6):1879–1889, September 2021.
- [247] Đula Nad, Christopher Walker, Igor Kvasić, Derek Orbaugh Antillon, Nikola Mišković, Iain Anderson, and Ivan Lončar. Towards Advancing Diver-Robot Interaction Capabilities. *IFAC-PapersOnLine*, 52(21):199–204, January 2019.

- [248] Derek Orbaugh, Christopher R. Walker, Samuel Rosset, and Iain A. Anderson. From New Zealand to Croatia: hand gesture control of an underwater robot using dielectric elastomer sensors. In *Electroactive Polymer Actuators and Devices (EAPAD) XXIV*, volume 12042, pages 194–206. SPIE, April 2022.
- [249] Shengshun Duan, Yucheng Lin, Chenyu Zhang, Yinghui Li, Di Zhu, Jun Wu, and Wei Lei. Machine-learned, waterproof MXene fiber-based glove platform for underwater interactivities. *Nano Energy*, 91:106650, January 2022.
- [250] Chelsey Edge and Junaed Sattar. Diver Interest via Pointing: Human-Directed Object Inspection for AUVs, December 2022. arXiv:2212.01205 [cs], Accepted at ICRA 2023.
- [251] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
- [252] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. Blazepose: On-device real-time body pose tracking, 2020.
- [253] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [254] W. He, P. Motlicek, and J. Odobez. Deep Neural Networks for Multiple Speaker Detection and Localization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 74–79, 2018.

- [255] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *MobileNetV2: Inverted Residuals and Linear Bottlenecks*, pages 4510–4520, 2018.
- [256] Mingxing Tan and Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114. PMLR, May 2019. ISSN: 2640-3498.
- [257] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [258] Gary Kane, Gonçalo Lopes, Jonny Sanders, Alexander Mathis, and Mackenzie Mathis. Real-time, low-latency closed-loop feedback using markerless posture tracking. *eLife*, 2020.
- [259] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1 € filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’12, pages 2527–2530, New York, NY, USA, May 2012. Association for Computing Machinery.
- [260] Greg Welch and Gary Bishop. An introduction to the kalman filter. *Proc. Siggraph Course*, 8, 01 2006.
- [261] F. N. Fritsch and J. Butland. A method for constructing local monotone piecewise cubic interpolants. *SIAM Journal on Scientific and Statistical Computing*, 5(2):300–304, 1984, <https://doi.org/10.1137/0905021>.
- [262] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD’96, page 226–231. AAAI Press, 1996.
- [263] Francois Chollet et al. Keras, 2015.

- [264] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.
- [265] Hassan Ramchoun, Mohammed Amine, Janati Idrissi, Youssef Ghanou, and Mohamed Ettaouil. Multilayer perceptron: Architecture optimization and training. *International Journal of Interactive Multimedia and Artificial Intelligence*, 4:26–30, 01 2016.
- [266] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, December 2022. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.
- [267] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [268] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnaud Carrera, Narcís Palomeras, Natàlia Hurtós, and Marc Carreras. Rosplan: Planning in the robot operating system. In *International Conference on Automated Planning and Scheduling*, 2015.
- [269] Somnath Arjun. Personalizing data visualization and interaction. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP ’18*, pages 199–202, New York, NY, USA, July 2018. Association for Computing Machinery.
- [270] Gonçalo S. Martins, Luís Santos, and Jorge Dias. User-adaptive interaction in social robots: A survey focusing on non-physical interaction. *International Journal of Social Robotics*, 11(1):185–205, 2019.

- [271] Adriana Tapus, Cristian Tăpuş, and Maja J. Matarić. User—robot personality matching and assistive robot behavior adaptation for post-stroke rehabilitation therapy. *Intelligent Service Robotics*, 1(2):169–183, April 2008.
- [272] Kim Baraka and Manuela Veloso. Adaptive Interaction of Persistent Robots to User Temporal Preferences. In Adriana Tapus, Elisabeth André, Jean-Claude Martin, François Ferland, and Mehdi Ammi, editors, *Social Robotics*, Lecture Notes in Computer Science, pages 61–71, Cham, 2015. Springer International Publishing.
- [273] Samantha Reig, Michal Luria, Janet Z. Wang, Danielle Oltman, Elizabeth Jeanne Carter, Aaron Steinfeld, Jodi Forlizzi, and John Zimmerman. Not some random agent: Multi-person interaction with a personalizing service robot. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, HRI ’20, pages 289–297. Association for Computing Machinery, 2020.
- [274] Edward Simon John, Sandro José Rigo, and Jorge Barbosa. Assistive Robotics: Adaptive Multimodal Interaction Improving People with Communication Disorders. *IFAC-PapersOnLine*, 49(30):175–180, January 2016.
- [275] Ali Sekmen and Prathima Challa. Assessment of adaptive human–robot interactions. *Knowledge-Based Systems*, 42:49–59, April 2013.
- [276] Erdem Biyik. Learning from humans for adaptive interaction. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 1152–1154, 2022.
- [277] Yunduan Cui, James Poon, Jaime Valls Miro, Kimitoshi Yamazaki, Kenji Sugimoto, and Takamitsu Matsubara. Environment-adaptive interaction primitives through visual context for human–robot motor skill learning. *Autonomous Robots*, 43(5):1225–1240, 2019.
- [278] Stavroula-Evita Fotinea, Eleni Efthimiou, Maria Koutsombogera, Athanasia-Lida Dimou, Theodore Goulas, Petros Maragos, and Costas Tzafestas. The mobot human-robot communication model. In *2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 201–206, 2015.

- [279] Silvia Rossi, Mariacarla Staffa, Luigi Bove, Roberto Capasso, and Giovanni Ercolano. User's Personality and Activity Influence on HRI Comfortable Distances. In Abderrahmane Kheddar, Eiichi Yoshida, Shuzhi Sam Ge, Kenji Suzuki, John-John Cabibihan, Friederike Eyssel, and Hongsheng He, editors, *Social Robotics*, Lecture Notes in Computer Science, pages 167–177, Cham, 2017. Springer International Publishing.
- [280] Edward T Hall. The hidden dimension, 1966.
- [281] Helge Huettenrauch, Kerstin Severinson Eklundh, Anders Green, and Elin A. Topp. Investigating Spatial Relationships in Human-Robot Interaction. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5052–5059, October 2006. ISSN: 2153-0866.
- [282] Dag Sverre Syrdal, Kheng Lee Koay, Michael L. Walters, and Kerstin Dautenhahn. A personalized robot companion? - The role of individual differences on spatial preferences in HRI scenarios. In *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication*, pages 1143–1148, August 2007. ISSN: 1944-9437.
- [283] Mohammad Safeea, Nuno Mendes, and Pedro Neto. Minimum Distance Calculation for Safe Human Robot Interaction. *Procedia Manufacturing*, 11:99–106, January 2017.
- [284] L. Meier, D. Honegger, and M. Pollefeys. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6235–6240, May 2015.
- [285] ArduPilot. <https://ardupilot.org/>. (Accessed on 03/01/2020).
- [286] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. International Society for Optics and Photonics, 1997.
- [287] robot_localization. https://github.com/cra-ros-pkg/robot_localization.

- [288] M. Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596, June 2005.
- [289] Christoph Bartneck, Tony Belpaeme, Friederike Eyssel, Takayuki Kanda, Merel Keijzers, and S. Sabanovic. *Human-Robot Interaction: An Introduction*. Cambridge University Press, 02 2020.
- [290] Céline Jost, Brigitte Le Pévédic, Tony Belpaeme, Cindy Bethel, Dimitrios Chrysostomou, Nigel Crook, Marine Grandgeorge, and Nicole Mirnig. *Human-Robot Interaction: Evaluation Methods and Their Standardization*. Springer Publishing, 05 2020.
- [291] Benedikt Leichtmann, Verena Nitsch, and Martina Mara. Crisis ahead? why human-robot interaction user studies may have replicability problems and directions for improvement. *Frontiers in Robotics and AI*, 9:838116, 03 2022.
- [292] Japan Agency for Marine Earth Science and Technology. *Deep-sea Debris Database*. <http://www.godac.jamstec.go.jp/catalog/dsdebris/e/index.html>. Accessed 02-10-2018.
- [293] Jose GB Derraik. The pollution of the marine environment by plastic debris: a review. *Marine pollution bulletin*, 44(9):842–852, 2002.
- [294] Michael Fulton, Jungseok Hong, Md Jahidul Islam, and Junaed Sattar. Robotic Detection of Marine Litter Using Deep Visual Detection Models. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5752–5758, 2019. ISSN: 2577-087X.
- [295] Supervisely. *Supervise.ly*, 2020. <https://supervise.ly/>. Accessed 06-30-2020.
- [296] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

- [297] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [298] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.
- [299] Paul A Miller, Jay A Farrell, Yuanyuan Zhao, and Vladimir Djapic. Autonomous underwater vehicle navigation. *IEEE Journal of Oceanic Engineering*, 35(3):663–678, 2010.
- [300] Pedro Batista, Carlos Silvestre, and Paulo Oliveira. A sensor-based controller for homing of underactuated AUVs. *IEEE Transactions on Robotics*, 25(3):701–716, 2009.
- [301] Ryan M Eustice. *Large-area visually augmented navigation for autonomous underwater vehicles*. PhD thesis, Massachusetts Institute of Technology and Woods Hole Oceanographic Institution, 2005.
- [302] Liam Paull, Sajad Saeedi, Mae Seto, and Howard Li. AUV navigation and localization: A review. *IEEE Journal of Oceanic Engineering*, 39(1):131–149, 2014.
- [303] Robert Panish and Mikell Taylor. Achieving high navigation accuracy using inertial navigation systems in autonomous underwater vehicles. In *OCEANS, 2011 IEEE-Spain*, pages 1–7. IEEE, 2011.
- [304] M Karimi, M Bozorg, and AR Khayatian. A comparison of DVL/INS fusion by UKF and EKF to localize an autonomous underwater vehicle. In *Robotics and Mechatronics (ICRoM), 2013 First RSI/ISM International Conference on*, pages 62–67. IEEE, 2013.
- [305] José Melo and Aníbal Matos. Survey on advances on terrain based navigation for autonomous underwater vehicles. *Ocean Engineering*, 139:250–264, 2017.
- [306] Alexander Ph Scherbatyuk. The AUV positioning using ranges from one transponder LBL. In *OCEANS'95. MTS/IEEE. Challenges of Our Changing Global Environment. Conference Proceedings.*, volume 3, pages 1620–1623. IEEE, 1995.

- [307] Marco Morgado, Paulo Oliveira, and Carlos Silvestre. Experimental evaluation of a USBL underwater positioning system. In *ELMAR, 2010 proceedings*, pages 485–488. IEEE, 2010.
- [308] Sebastian Carreno, Philip Wilson, Pere Ridao, and Yvan Petillot. A survey on terrain based navigation for AUVs. In *OCEANS 2010*, pages 1–7. IEEE, 2010.
- [309] Thomas B Schon, Rickard Karlsson, and Fredrik Gustafsson. The marginalized particle filter in practice. In *Aerospace Conference, 2006 IEEE*, pages 11–pp. IEEE, 2006.
- [310] Francisco Curado Teixeira, João Quintas, Pramod Maurya, and António Pascoal. Robust particle filter formulations with application to terrain-aided navigation. *International Journal of Adaptive Control and Signal Processing*, 31(4):608–651, 2017.
- [311] Nathaniel Fairfield and David Wettergreen. Active localization on the ocean floor with multibeam sonar. In *OCEANS 2008*, pages 1–10. IEEE, 2008.
- [312] Tamaki Ura, Takeshi Nakatani, and Yoshiaki Nose. Terrain based localization method for wreck observation auv. In *OCEANS 2006*, pages 1–6. IEEE, 2006.
- [313] Stefan Williams and Ian Mahon. A terrain-aided tracking algorithm for marine systems. In *Field and Service Robotics*, pages 93–102. Springer, 2003.
- [314] Deborah K Meduna, Stephen M Rock, and Rob McEwen. Low-cost terrain relative navigation for long-range AUVs. In *OCEANS 2008*, pages 1–7. IEEE, 2008.
- [315] Taeyun Kim and Jinwhan Kim. Nonlinear filtering for terrain-referenced underwater navigation with an acoustic altimeter. In *OCEANS 2014-TAIPEI*, pages 1–6. IEEE, 2014.
- [316] Takeshi Nakatani, Tamaki Ura, Takashi Sakamaki, and Junichi Kojima. Terrain based localization for pinpoint observation of deep seafloors. In *OCEANS 2009-EUROPE*, pages 1–6. IEEE, 2009.
- [317] Bo He, Ke Yang, Shuai Zhao, and Yitong Wang. Underwater simultaneous localization and mapping based on EKF and point features. In *Mechatronics and*

- Automation, 2009. ICMA 2009. International Conference on*, pages 4845–4850. IEEE, 2009.
- [318] Ba-Da Yoon, Ha-Nul Yoon, Sung-He Choi, and Jang-Myung Lee. UKF Applied for Position Estimation of Underwater-Beacon Precision. In *Intelligent Autonomous Systems 12*, pages 501–508. Springer, 2013.
 - [319] Sebastian Thrun. Particle filters in robotics. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 511–518. Morgan Kaufmann Publishers Inc., 2002.
 - [320] Rickard Karlsson, Fredrik Gustafsson, and Tobias Karlsson. Particle filtering and Cramer-Rao lower bound for underwater navigation. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 6, pages VI–65. IEEE, 2003.
 - [321] Ioannis Rekleitis. A Particle Filter Tutorial for Mobile Robot Localization. Technical Report TR-CIM-04-02, Centre for Intelligent Machines, McGill University, 3480 University St., Montreal, Québec, CANADA H3A 2A7, Jan. 2004.
 - [322] David Salmond and Neil Gordon. An introduction to particle filters. *State space and unobserved component models theory and applications*, pages 1–19, 2005.
 - [323] Francesco Maurelli, Angelos Mallios, David Ribas, Pere Ridao, and Yvan Petillot. Particle filter based auv localization using imaging sonar. *IFAC Proceedings Volumes*, 42(18):52–57, 2009.
 - [324] Fredrik Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, 2010.
 - [325] Thomas B Schön, Fredrik Gustafsson, and Rickard Karlsson. The particle filter in practice. In Boris Rozovskii Dan Crisan, editor, *The Oxford Handbook of Nonlinear Filtering*, pages 741–767. Oxford University Press, 2011.
 - [326] Thomas Schon, Fredrik Gustafsson, and P-J Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53(7):2279–2289, 2005.

- [327] Rickard Karlsson and Fredrik Gustafsson. Bayesian surface and underwater navigation. *IEEE Transactions on Signal Processing*, 54(11):4204–4213, 2006.
- [328] Eric W Weisstein. Euler angles. <https://mathworld.wolfram.com/>, 2009.
- [329] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.
- [330] Fred Daum. Nonlinear filters: beyond the Kalman filter. *IEEE Aerospace and Electronic Systems Magazine*, 20(8):57–69, 2005.
- [331] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.
- [332] Thomas B Schön, Rickard Karlsson, and Fredrik Gustafsson. The marginalized particle filter—analysis, applications and generalizations. In *ESAIM: Proceedings*, volume 19, pages 53–64. EDP Sciences, 2007.
- [333] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.
- [334] Natural Resources Department. Lake Bathymetric Outlines, Contours, Vegetation, and DEM, June 2018. <https://gisdata.mn.gov/dataset/water-lake-bathymetry>.
- [335] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.
- [336] Kristopher Krasnosky, Christopher Roman, and David Casagrande. A bathymetric mapping and SLAM dataset with high-precision ground truth for marine robotics. *The International Journal of Robotics Research*, 41(1):12–19, 2022.

Glossary and Acronyms

Care has been taken in this thesis to minimize the use of jargon and acronyms, but this cannot always be achieved. This appendix defines jargon terms in a glossary, and contains a table of acronyms and their meaning.

Glossary

- **AUV** – Autonomous underwater vehicle, an underwater robot.
- **Co-AUV** – Collaborative autonomous underwater vehicle, an underwater robot focused on human collaboration.
- **ROV** – Remotely operated vehicle, a remote-controlled underwater robot.
- **Communication Vector** – A general term for a method of communicating information from an AUV to a diver.
- **Communication Phrase** – A general term for a self-contained sound, light, motion, or text display from a communication vector with a meaning.
- **RCVM** – Robot communication via motion, a motion-based communication method for AUVs.
- **Kineme** – A motion sequence produced by RCVM with a specific meaning.
- **HREye** – A light-based communication method for AUVs.
- **Luceme** – A sequence of lights produced by HREye or another light-based communication method with a specific meaning.

- **SIREN** – Sound Indicators via Resonance Exciters uNderwater, a sound-based communication method for AUVs.
- **Soneme** – A sound sequence produced by SIREN or another sound-based communication method with a specific meaning.
- **ADROC** – Autonomous Diver-Relative Operator Configuration, a method for diver approach.
- **ACVS** – Autonomous Communication Vector Selection, an adaptive method of selecting communication vectors based on the context of an interaction.
- **PROTEUS** – A software framework for underwater HRI.
- **Protean** – A gesture language for AUVs.
- **POSH-G** – Protean One-Shot Hand Gestures, a method for learning gestures based on the generation of synthetic gesture data from a single demonstration.
- **VDD- \bar{C}** – The Video Diver Dataset, a dataset of over 100,000 images from videos of divers.
- **OceanPose-Training** – A training dataset for pose estimation of underwater swimmers.
- **OceanPose-Evaluation** – An evaluation dataset for pose estimation of underwater swimmers.
- **PGR** – Protean Generation and Recognition dataset, a dataset for generating and recognizing Protean gestures.

Acronyms

Acronym	Meaning
AUV	Autonomous Underwater Vehicle
Continued on next page	

Table G1 – continued from previous page

Acronym	Meaning
Co-AUV	Collaborative Autonomous Underwater Vehicle
ROV	Remotely Operated Vehicle
HRI	Human-Robot Interaction
UHRI	Underwater Human-Robot Interaction
RCVM	Robot Communication Via Motion
SIREN	Sound Indicators via Resonance Exciters uNderwater
ACVS	Autonomous Communication Vector Selection
ADROC	Autonomus Diver-Relative Operation Configuration
DCM	Diver Context Module
LDF	Language Definition File
SDF	Symbol Definition File
MLP	Multi-layer Perceptron
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory network
POSH-G	Protean One-Shot Hand Gestures
OPT	OceanPose-Trainging
OPE	OceanPose-Evaluation
PGR	Protean Generation and Recognition datasets

Appendix I: Building LoCO AUV

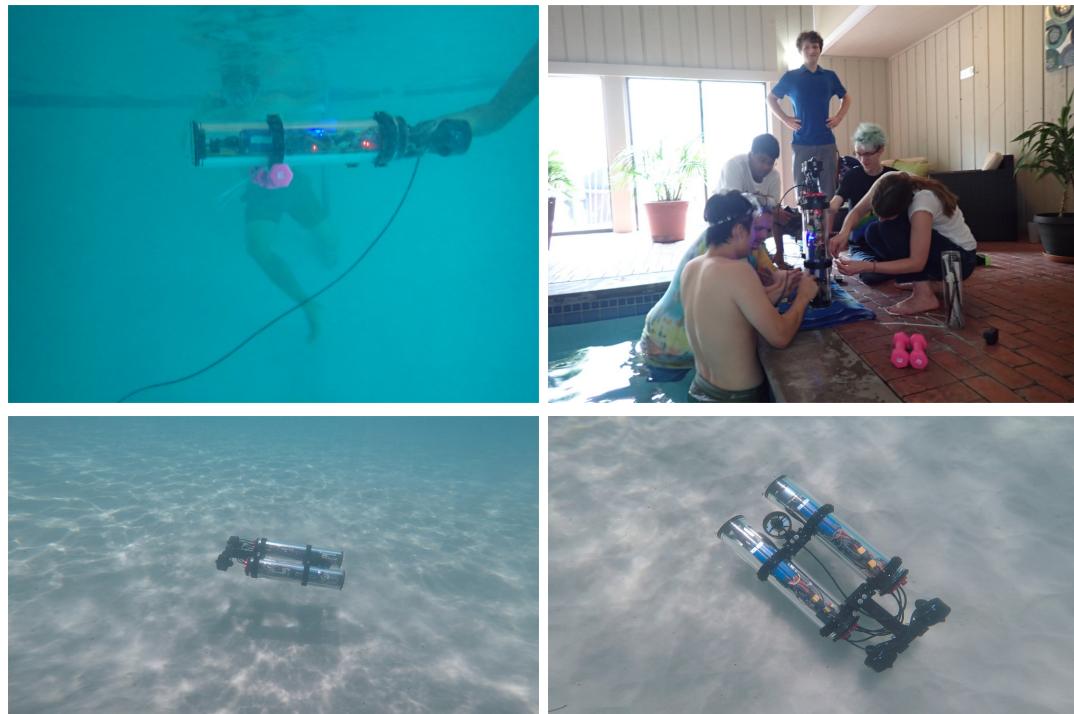


Figure A1.1: A sampling of early LoCO deployments.

While work presented in this thesis was conducted and implemented onboard a variety of robots, the vast majority was implemented on the LoCO AUV. The **Low Cost Open Autonomous Underwater Vehicle** is a small, human portable AUV which is built from off-the-shelf and 3D-printed parts. LoCO was briefly discussed in the thesis proper, primarily in Chapter 1, Section 1.3.2. In this appendix, we provide a description of the initial development of the LoCO AUV, along with more complete descriptions of the

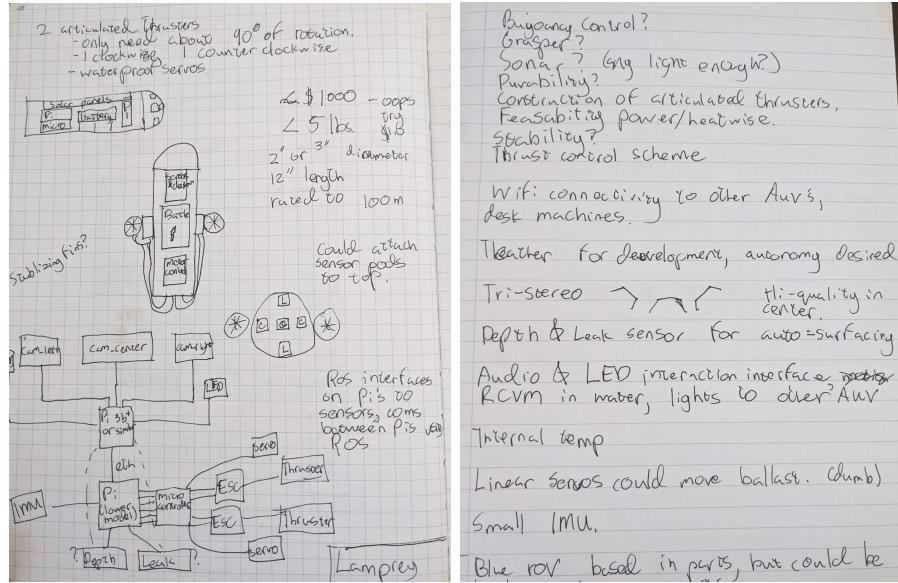


Figure A1.2: Original sketch and concepts for LoCO from Michael Fulton's notebooks, drawn around September 2018. This design evolved into the abandoned submarine concept in Figure A3, but many elements that are core to LoCO were established here.

hardware, software, and assembly of the AUV.

Initial Concept and Design Process

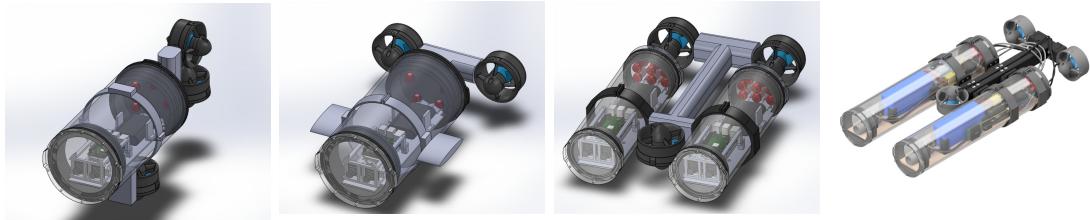
LoCO came into existence in the minds of both Md. Jahidul Islam and myself around the same time, as we both proposed building a small, low-cost AUV sometime around the fall of 2018. The core idea of creating an AUV and keeping costs low is nothing new, but LoCO is unique in its open design, its price point, and its capabilities. Some aspects of these ideas can be seen in the notes and diagram contained in the pages of Figure A1.2, which were drawn in August or September 2018. By this point, some parts that were selected for the final AUV were being considered, concepts such as multi-camera vision had been suggested, and issues that would be long-standing such as heat management had been raised. After both Ph.D. students brought their ideas to Dr. Sattar, further interest was expressed by multiple students, and a team of graduate and undergraduate students (initially titled “Cheapo Blub Blub Dev Team”) was convened with the purpose



Figure A1.3: The MiniEye stereo vision rig, a precursor to the LoCO AUV.

of developing the idea into a platform.

In the initial months of development, the concept was drastically simplified to make enable the team to produce something in time for the upcoming Barbados Sea Trials in January 2019. This concept was eventually reduced to producing a non-powered, self-contained stereo-IMU rig, with which to collect stereo images, IMU data, and depth information. This device was christened the MiniEye and gave the team their first experience in building mounting structures inside the watertight enclosure from BlueRobotics that forms the center of both LoCO and MiniEye. MiniEye performed admirably in Barbados in 2019 (Figure A1.3), convincing the team (now the “MiniEye/MiniBot Dev



(a) “Helicopter” style. (b) “Submarine” style. (c) “Binocular” style. (d) Final design.

Figure A1.4: CAD renderings showing the development of LoCO AUV.

Team”) that the approach of building sensor rigs and AUVs out of off-the-shelf components, placed inside a BlueRobotics enclosure was viable.

Upon returning from Barbados, the team began to shift to developing a fully-featured AUV. Many concepts for the construction of this AUV were suggested, as seen in Figure A1.4. The design which was settled upon, the two-tube or “binocular” design, was selected. After nearly a year of development, the AUV (now called the Minibot/Edgebot), was brought to Barbados for sea trials in January 2020. At this point, the robot was functional, but non-autonomous, having only a teleoperation system and no way to trigger software externally. This was fixed during the sea trials of 2020, where Michael developed a basic autopilot control system, and adapted the HRI control systems which were used on Aqua to use on LoCO. The 2020 trials also gave LoCO its final name: “LoCO-AUV”. After some delay due to the beginning of the COVID-19 pandemic, LoCO’s design and assembly instructions would be released publicly at loco-auv.github.io in November of 2020, two years and three months after the first images of the robot were sketched. While the adoption of LoCO as a research platform has been somewhat slow, teams of students have worked on building LoCO and LoCO-inspired AUVs around the world.

Hardware Description

LoCO is designed to be an all-purpose AUV, adaptable to various missions. The standard configuration is a dual-camera, vision-guided AUV with three thrusters.

LoCO AUV Specifications

Dimensions (L x W x H)	73.1cm x 34.4cm x 14.1 cm
Weight	12.47 kg (27 lb)
Maximum Speed	1.5 m/s
Battery Life (Idle)	18 hrs, 30 mins
Battery Life (Average)	2 hrs, 20 mins
Battery Life (Max Thrust)	30 mins

Table A1.1: LoCO AUV Specifications.

Overall System Layout

LoCO is comprised of two water-tight enclosures, each containing various components, with one thruster mounted between the enclosures, and two mounted behind, as seen in Figure 1.4, in Chapter 1. While several designs were considered (see Fig. A1.4), the two-enclosure design was selected for various reasons. Firstly, it allowed for a reasonable placement of a pitch-control thruster, along with providing space in between the enclosures which would be an excellent place to mount new sensors, thrusters, or manipulators in the future. Additionally, the design called for two enclosures side by side, narrower than the enclosures used for the other designs, which would reduce the robot’s forward profile, allowing it to move through the water with less resistance. Finally, the separation into two enclosures enforces a base level of modularity. Most control-related electronics are in the left-hand enclosure, with the computational hardware for deep learning inference in the right-hand enclosure. While any type of hardware modification requires changes throughout the system, changes or replacements can be made with minimal impact on the layout of components internally.

Watertight Enclosures and 3D-printed substructure

The enclosures used are cast acrylic structures with a 10.16 cm internal diameter, produced by Blue Robotics. The enclosures are rated to 100m depth of operation, sufficient for our needs. Their length is variable to any modifications that are made, but the length for our configuration is 50.8cm. The front of each enclosure is a flat piece of cast

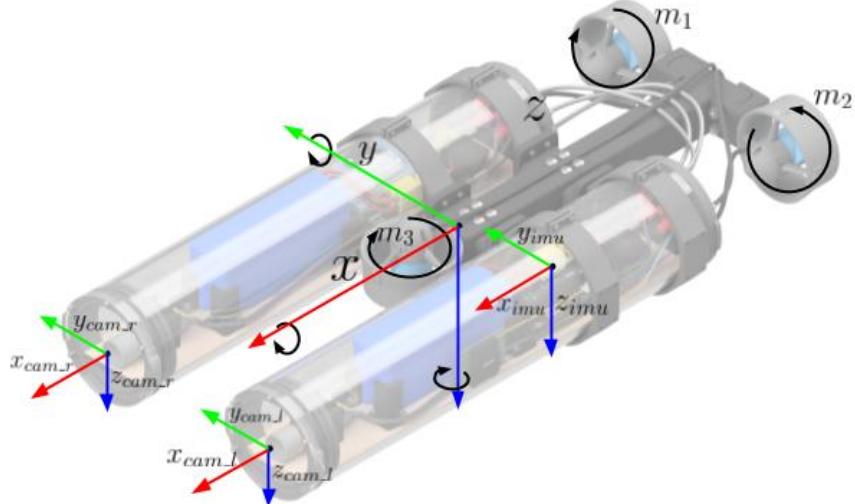


Figure A1.5: A free-body diagram of LoCO with IMU, camera, thruster, and robot frames.

acrylic to reduce visual distortion and increase surface area for mounting cameras, while the back cap is aluminum with openings for cables, plugged by nut-and-bolt assemblies termed penetrators. They are linked with a custom 3D-printed structure connected to aluminum clamps manufactured by Blue Robotics. This structure is also used to mount the thrusters.

Internally, components are mounted on a piece of laser-cut medium-density fiberboard (MDF), with 3D-printed mounting substructures. The MDF is held in place within the enclosures via a 3D-printed part that attaches to the penetrators poking through the back plate. A strip of velcro beneath the MDF is used to attach 28-gram blocks of ballast cut from stainless steel bar stock. There is also enough space under the MDF to fit bags of desiccant, which help to eliminate condensation in the event that the enclosure is sealed in humid air (common in pool and tropical environments). This internal design is beneficial to the modularity of LoCO, as it allows for additional parts to be mounted by allocating space on the MDF and then simply attaching them with screws. Additionally, the external structure design is flexible, allowing for the movement of the clamps along the enclosures to fix the thrusters wherever is appropriate. The

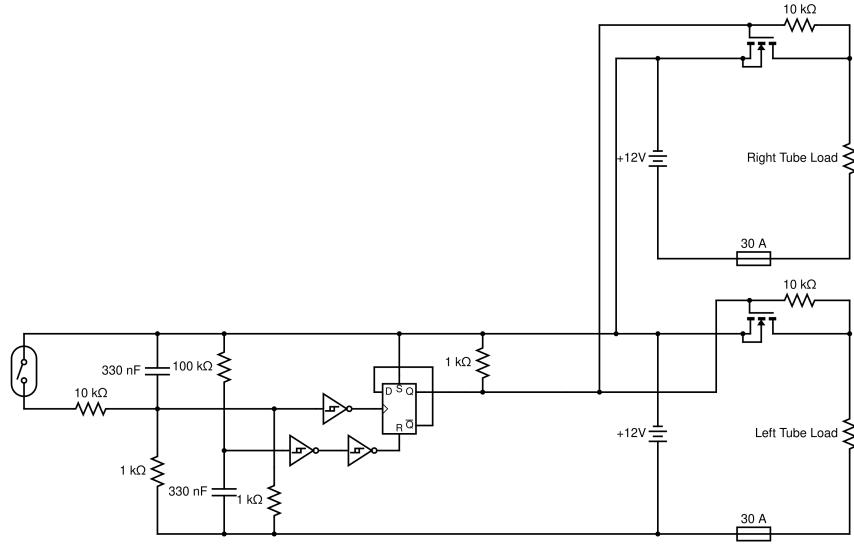


Figure A1.6: Power Systems Diagram.

“backbone” 3D-printed part between the clamps can be omitted with no apparent loss of structural integrity. It would, however, provide useful mounting space for external sensors or actuators, such as a sonar altimeter or a gripper.

Electrical Systems

LoCO’s electronics are powered by four 11.1 volt lithium-polymer 8000 mAh batteries, two in each enclosure. The batteries in each tube are connected in parallel, providing a supply of 9.6-12.6 volts depending on their charge. Each tube has a low-voltage alarm which sounds when the two batteries fall below the minimum threshold of 9.6 volts. The electrical systems in each enclosure are designed in such a way that they only power components in their respective tube. Each pair of batteries is connected to a power distribution board (PDB) which provides six 12 volt outputs, and one 5 volt output. In the case of the left-hand enclosure, an external 5 volt step-down converter was added in order to power the additional computing systems in that enclosure. Despite the electrical systems of each tube being separate, a single power switch circuit controls the On/Off state of the entire robot. The circuit is situated in the left enclosure and utilizes a magnetic reed switch for input. This allows LoCO to be turned on and off using

a magnetic “key”. By using a magnetic key rather than a physical switch, we reduce possible points of failure due to leakage. The left portion of Fig. A1.6 contains the reed switch and components that create the toggle signal, while the right side diagrams how the toggle signal is used to control the power sources.

Thruster Control Systems

The thrusters employed for LoCO are Blue Robotics T100s, which use a brushless DC motor and a plastic propeller. Blue Robotics is discontinuing the T100, but they also produce a more powerful thruster (the T200) in the same dimensions, so future builds of LoCO AUVs will use T200s. These thrusters are controlled via pulse width modulation (PWM), which is managed by electronic speed controllers (ESCs). In turn, the ESCs are controlled by a Pixhawk autopilot board, employing the ArduPilot/ArduSub control software. While we do not currently make use of all the features of the Pixhawk and ArduSub, the open software and hardware of the PX4 [284] and ArduPilot [285] projects lend themselves well to our goals of creating an open platform which others can contribute to and adapt to their needs.

Computing Systems

LoCO has two primary computer systems: a Jetson TX2 for deep learning inference and a Raspberry Pi 4 (4GB) for control. The Jetson TX2 is mounted on a Connect Tech Orbitty carrier board for interfacing and is largely responsible for managing processes that involve deep neural network inference. Due to its location in the right-hand enclosure, however, the TX2 is also responsible for managing the robot’s OLED display via the connected microcontroller and processing images from the camera mounted in the right enclosure. In the left enclosure, the Raspberry Pi is used to process images from the camera in that enclosure and used as the controller interfacing with the Pixhawk autopilot over a serial connection. The Jetson and the Raspberry Pi are connected via Ethernet with a Cat5e cable for a maximum throughput of 100Mbit/s.

Vision Systems And Other Sensors

LoCO was designed with a stereo vision system in mind. However, creating such a system has proved quite difficult. Many self-contained stereo modules are too large to fit in the enclosures selected, making them non-starters. Some modules were the appropriate size, but proved to be unsuited in terms of their software licensing. The original goal was to enable stereoscopic vision by synchronizing single cameras in each enclosure, but the cameras evaluated thus far have been unsuitable for one reason or another. Both enclosures currently contain a significantly less expensive USB camera from Blue Robotics, which is an excellent low-light camera. Sadly, these cameras cannot be hardware-triggered, and therefore cannot be used for stereoscopic vision. In addition to its vision system, LoCO employs a pressure sensor to measure its depth under the surface, which is mounted on the back plate of the left enclosure. There is also an inertial measurement unit (IMU) contained within the Pixhawk autopilot unit. A sonar altimeter is also being integrated.

Software Description

LoCO has a variety of computing devices: a Raspberry Pi 4, an Nvidia Jetson TX2, a Pixhawk autopilot unit, and an Adafruit Trinket Pro microcontroller. The Raspberry Pi and Jetson TX2 both run versions of Ubuntu, a popular open source operating system, while the Pixhawk runs a real-time open source OS with ArduSub, and the Trinket is flashed directly with open-source software. The software which allows LoCO to function as an untethered autonomous vehicle rather than an ROV is distributed across the computing devices, and consists of a mixture of ROS packages, ArduSub, and Arduino code. All of this software is under some form of permissive, open source license, which makes LoCO's software stack free for users to explore, expand, and enhance. In this section we describe a portion of LoCO's software, omitting a full description for brevity.

State Estimation

A number of options are under development for state estimation in LoCO. Firstly, the Pixhawk provides an IMU-based state estimation using the Extended Kalman Filter [286]. While this is useful, LoCO currently uses the `robot_localization` [287] ROS

package to estimate its orientation via IMU data directly from the Pixhawk’s IMU, as this provides greater control over the tuning of the extended or unscented Kalman filters provided by the package. Additionally, the package allows the fusion of multiple sources of information into one estimate, so if additional IMUs or sources of information became available, they could easily be integrated. Two possible sources of this information are currently in development: a downward-facing camera for monocular visual odometry and an odometry estimate based on a combination of thruster inputs and a hydrodynamic motion model of the robot.

LoCO Pilot Controller

In order to facilitate motion control of LoCO from a variety of sources (a teleoperation mode, or autonomous behavior algorithms), a motion control package entitled `loco_pilot` has been developed. The package provides an interface which abstracts the control into a simple message type (`\loco_pilot\Command`), containing thrust, pitch, and yaw values between -1.0 and 1.0 . This allows users to avoid the `mav_ros`, MAVLink, and Ardusub systems which are required for controlling the robot, and simply publish these messages to the correct topic. In addition, the `loco_pilot` package implements a set of motion primitives and advertises them as ROS services, allowing one to simply call a service to turn the robot to an angle, move forward or back, or follow a circle or square trajectory. The package is under active development, adding more features and capabilities as the state estimation of the robot improves.

Menu Control System

LoCO has a menu system, implemented in the ROS package `loco_menu` which allows an operator to control the robot in untethered mode, by inputting commands via hand gestures or ARTags [288]. This menu system allows a user to select an option, which can be assigned to be a variety of subroutines, including ROS service calls, ROS launch files, etc. The robot’s user selects from a set of 5 options using one of the methods described in the following subsection. While these options are displayed on LoCO’s OLED display for ease of use, the menu system operates independently of its display component. These options could be connected to a variety of ROS endpoints, including running a launch file,



Figure A1.7: LoCO’s OLED display showing a menu.

launching a node, calling a service, terminating a node or setting a parameter. Menu options can also be set to submenus, allowing nesting and categorization of options. Once a menu item has been selected, while response depends on what endpoint the option has been linked to, typically an action is taken relatively quickly, then the menu is available again once the action has been completed. In some cases, the option has a timeout associated with it, or must be manually canceled. The system is designed to be as adaptable as possible, with menu definitions being loaded in the form of yaml configuration files, making the process of writing new menu configurations as painless as possible.

ARTags and Hand Gestures

LoCO employs two vision-based methods to select menu items. The first uses small AR tags as “flashcards”. A ROS package based on AR Toolkit detects tags in the view of the left enclosure’s camera. To select a menu item, one simply has to display an AR tag corresponding to the number of the item. Alternatively, LoCO has been outfitted with a gesture recognition system, which can be used similarly. This gesture recognition system was initially presented for the Aqua AUV [2], and is based on a hand pose classification deep neural network. To select a menu item, one must simply initiate a selection with the “Ok” gesture, then show the camera the number gesture for the appropriate item.

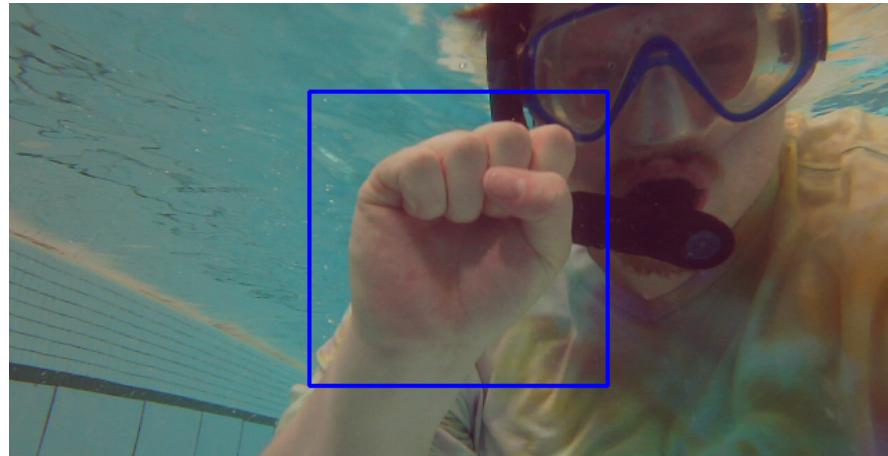


Figure A1.8: Gesture detection for a “0” gesture.

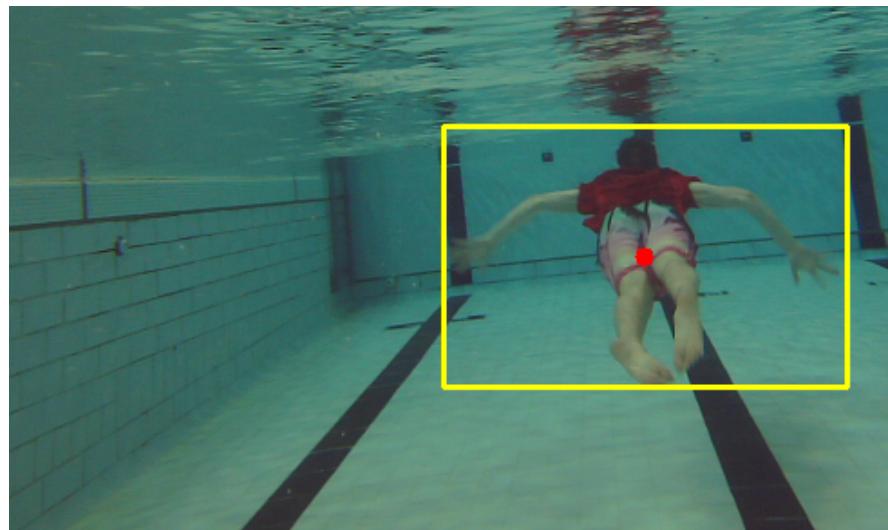


Figure A1.9: LoCO-eye view of following a diver.

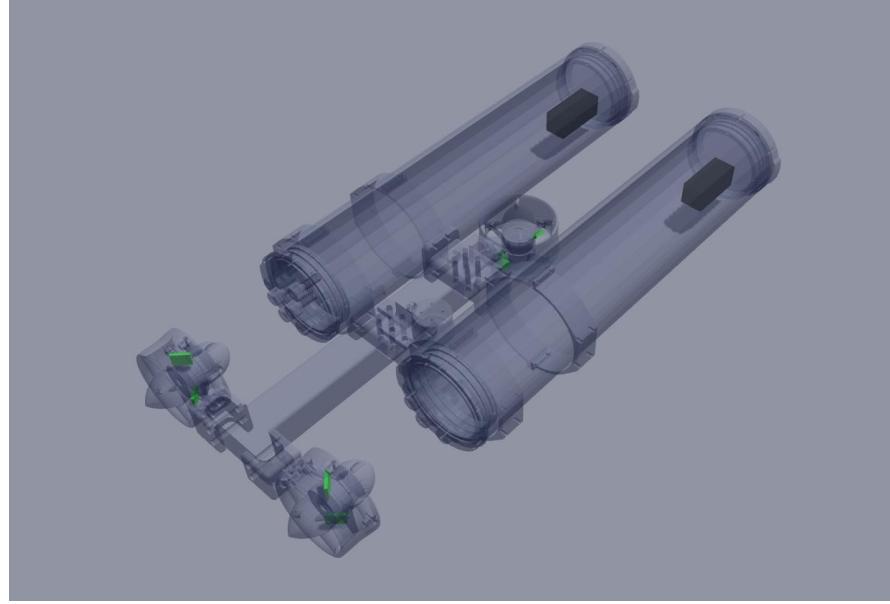


Figure A1.10: The LoCO AUV in Gazebo simulation.

Diver Following

Another package considered a standard part of LoCO's software is a diver-follower. While diver following has been explored by a variety of authors, the ability of an AUV which works with human partners to follow those partners is foundational. Diver following can be used to convoy with a robot to a chosen location, to guide a robot through a specific route during data collection, or simply as the end goal, to name a few uses. The diver detection is completed using a Tiny-YOLOv4 model trained on VDD- \bar{C} (Chapter 5). Once the detection has been made, a PID-based controller is used to generate thruster control inputs which will maneuver the bounding box into the center of the frame. The algorithm uses bounding box size as a rough, stereo-free estimate of distance to the diver. While there is room for improvement in the diver following in practice, the existing algorithm does a serviceable job of following.

Gazebo Simulator

As with most fields of robotics, underwater roboticists often use simulators to validate algorithmic ideas before developing systems for field deployment. For this purpose, a Gazebo-based simulator for LoCO was developed, using the calculation of hydrodynamic

forces applied to LoCO directly. This simulator, which is visually simple but dynamically complex, is pictured in Figure A1.10. LoCO's inertial matrix and center of mass location were approximated based on a uniform density for all components using the SolidWorks Mass Analysis tool, taking into account the pre-existing mass measurements of LoCO. The center of mass was approximated between the LoCO tubes laterally and vertically, just behind the vertical thruster (see Fig. A1.5). To improve simulation efficiency while not affecting simulation physics, the mesh file that provides the visual representation of the robot does not include internal components. Collision properties for each link of the robot were also specified so the simulation can model physical impacts between the robot and its surroundings. The collision boundaries were modeled to be the same as the visual boundaries for all links except the main body of LoCO. Due to the size and complexity of the mesh, the collision boundary for the main body was modeled as a box in order to decrease simulation computational requirements.

Building Instructions

One of the purposes of the LoCO project was to create an AUV that could be built by anyone with the skills, the tools, and a relatively small budget (when compared to other AUVs). The complete building instructions for LoCO can be found at https://github.com/LoCO-AUV/loco_config/wiki#assembly-instructions, but a brief summary follows. After acquiring the required parts, the internal and external mounting structures must be fabricated. The primary base of the robot is two MDF boards, which can be cut on a laser cutter, or (with extreme care) be cut and drilled by hand. The other mounting structures are 3D printed using PETG or PLA materials. Following this, the power system should be assembled and validated. Ensure that the batteries for each tube are connected in parallel, or the majority of the components in the enclosures will fry. The power switch is the most complex component in LoCO, requiring a bit of custom soldering, but should be manageable by anyone with at least an intermediate level of soldering skill. With the power systems assembled, the thrusters, computing, and network systems can all be built relatively quickly. Again, some minor soldering is required, along with cable fabrication. Take care when building cables, the quality of these cables will have ripple effects in the system. After assembling these systems along

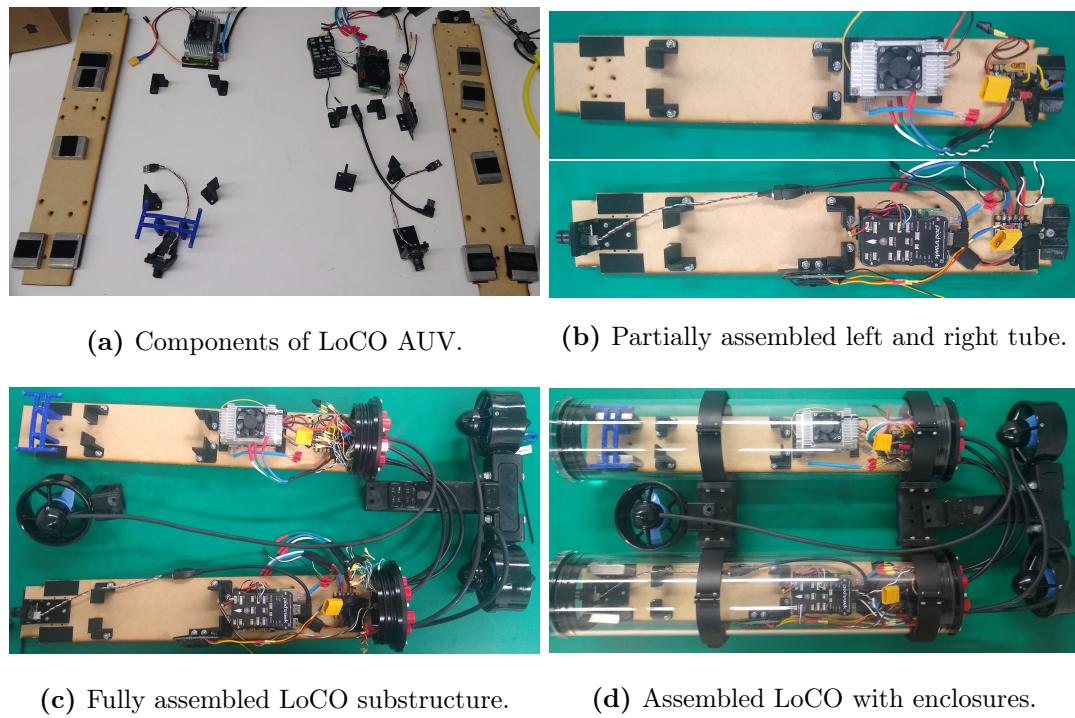


Figure A1.11: The assembly process for LoCO AUV, from components to fully assembled (sans batteries). The process takes approximately 2 hours with one person and components pre-fabricated.

with the OLED and sensors, the robot is ready to assemble. The results of this process can be seen in Figure A1.11. It is recommended to complete the software configuration of the computers at this stage, prior to placing them in the enclosure, as it will be more difficult to get access to various ports after assembling the robot. The assembly process will take at least an hour, possibly more. The processes featured on the LoCO wiki should not be considered error-free. Much has been improved since documentation that has not yet been released and changes have occurred in the tech landscape since the first LoCO build. Use the procedures in the wiki as a guide, learn from your mistakes, try not to make those mistakes on expensive hardware, and most of all, have fun while you're doing it.

Appendix II: Human Study

Research Methods

The research presented in this thesis is highly multi-disciplinary. This directly results from the topic at hand: human-robot interaction in underwater environments. To conduct research in this space, one must have programming and hardware experience, be capable of designing and administering human studies using quantitative and qualitative methods, and be familiar with UX and UI design. As a computer scientist by training, my skills tended towards developing software and hacking on hardware, with human research methods being much less familiar. This appendix covers a small portion of what I have learned about human research methods, not formatted as an academic contribution or teaching materials, but a brief set of recommendations based on the experience gained over my Ph.D. Do not treat this as an instructional manual, simply as advice from a student of this field during the years 2017-2023 (at least). Because of this, the writing in this appendix will also be somewhat more conversational and informal than the rest of the thesis.

HRI Research Practices

Studying human-robot interaction is a difficult task. As discussed in the opening of this appendix, the field is multi-disciplinary by definition, which increases the number of skills that one has to acquire or source externally in order to effectively conduct research in HRI. The recommendations of this section focus on ways to develop your skills in this area, expand your knowledge, and contribute effectively to this field.

Recommended Reading

There are sadly no definitive textbooks on HRI at this stage, though this may be changing in the near future. Texts that have already been published may soon become standard, such as the introductory textbook by Bartneck *et al.*, [289], which I recommend. Additionally, the book on HRI research methods by Jost *et al.*, [290] is an excellent text. Bartneck's book is more well-organized, whereas Jost's is essentially a collection of papers rather than a cohesive book. Both have useful information for researchers, regardless of their experience in the field. Additionally, I highly recommend reading the recent paper by Leichtmann *et al.*, [291]: "Crisis Ahead? Why Human-Robot Interaction User Studies May Have Replicability Problems and Directions for Improvement". This paper discusses the "Confidence Crisis" that is rampant in the soft/social sciences and applies the same methods of criticism to the field of HRI. Beyond being a sobering look at the field as a whole, this paper provides a number of extremely helpful suggestions for changes in research methods and statistical analysis, which I highly recommend implementing. Between these texts, the reader will gain an understanding of a large portion of the HRI community's opinions on research methodologies.

Find a Research/Lab and Analyze Their Methods

Particularly when one is less experienced, it can be helpful to imitate the methods of others in the field. This is a recommendation that must be taken with a grain of salt, as other researchers are not infallible simply because they are successful. With these caveats in mind, it can be useful to find a researcher whose work you admire and analyze their recent papers, not for their goals or contributions, but for the methods of testing and analysis employed. Consider the core testing methodology (quantitative, qualitative, mixed), sample sizes, training methods, and statistical tests. Do not simply form a list and attempt to use the same methods, but consider which practices you wish to imitate, which are domain-specific, and which are questionable and should be avoided. This practice can be incredibly helpful in developing your own sense of good methods.

Follow Needs, Find Experts

This is a good piece of advice in general for developing HRI systems: find a need, then find the people who are experts on that need. Using their input, develop a solution to the need, and test the solution amongst experts. However, this recommendation is a slight variation on that theme. When designing an HRI project, whether you are considering the software/hardware systems, the study, or the analysis, focus on what needs you have, and attempt to find experts in that area. For instance, when exploring the use of sound in underwater HRI (Chapter 4), I sought input from Rafa Absar, whose Ph.D. thesis focused on the use of sound for human-computer interaction. Her advice was invaluable in designing both the sonemes of the SIREN system and the evaluating studies, as well as writing the paper. As much as possible, seek collaboration with experts in the areas that you need help with, but remember to not only lean on their input but learn from it so that you will be more capable in the future.

Don't Be Afraid to Innovate

Lastly, after the previous recommendations, which tend towards greater compliance with a supposed standard of practice in the field, I recommend the following: don't be afraid to innovate! While innovation may be less advisable in areas such as the statistical methods you employ, the process of evaluating an HRI system is often extremely specific to the form the system takes. Make your innovations carefully and avoid introducing confounding variables, but don't be so paranoid that you never try to expand the field methodologically.

Recommendations for Human Study Design

One of the more difficult aspects of studying human-robot interaction, at least for students coming from engineering backgrounds, is designing and running human studies. This task requires a different skillset than most engineering tasks, and may not be something that you have been adequately prepared to do. The following section contains a few recommendations on this topic, but beyond what is listed here, I would recommend taking a course or finding a textbook to guide you, if you truly have no experience with human studies and do not have an advisor or collaborator who has experience.

Seek Input Early and Often

My first advice is to seek input. The best people to ask for advice are experts on study design (faculty who have conducted many studies over their career, senior graduate students, etc.). However, simply by explaining your design to another layperson, you may find issues that had not immediately become apparent to you. In fact, in the process of writing an IRB protocol (which is discussed in the following set of recommendations), you may find issues or unmet needs. For this reason, I cannot recommend enough starting by seeking input. Before building any study instruments (forms, testing scripts, etc), before advertising the study, and certainly, before conducting it, seek input from others.

Avoid Bias In Baselines

A common problem in HRI studies (including some of the studies presented in this thesis) is difficulties with comparing or reproducing studies. This is partially an aspect of the field: it can be difficult to compare two robots' capabilities if they are significantly different. However, this issue can also arise due to poor scientific rigor. When testing an HRI system, there is often a need for a point of comparison, a baseline against which to test your system. In more formal terms, this could be termed a "control". While it can be tempting to simply throw together a rough baseline system, I would caution against it. When selecting or building a baseline system, I recommend following this heuristic:

1. Look for an in-use system. This could be the system presented in another paper, the common method used in industry, etc. If such a system exists, you should use it, unless you have a very convincing reason not to (if you cannot afford the tech, for instance).
2. If there is an in-use system that you cannot actually use, attempt to replicate its capabilities as carefully as you can.
3. Lastly, if there is no system currently in use, you have two choices. Firstly, you could simply profile the capabilities of your system in some way, without a point of comparison. If you are researching a truly new capability, having no point of

comparison should not be a significant downside. Secondly, if you decide you must have a baseline, do your absolute best to create a reasonable baseline approach.

In some instances, building a baseline system may lead you to other interesting work. For instance, the HREye and SIREN communication systems presented in Chapters 3 and 4 respectively evolved out of the baseline systems created for the third RCVM study, Study III (presented in Section 2.8). Regardless of the difficulty, ensure that you built fair baseline systems. Your research will be more robust for the effort, even if the effort of building baselines itself is never rewarded.

Value High-Quality Participants Over High Volume

A common question encountered in human studies is “how many participants is enough?” In many ways, the answer is as many participants as one can collect, but there are other ways of looking at it. Firstly, a common method for selecting sample sizes is conducting a power analysis *a priori* to determine the required number of participants. However, the types of studies required for HRI, particularly underwater HRI, are expensive and hard to gather participants for. In many cases, researchers may be left with a choice between waiting for months longer to collect a more technically correct number of participants, or attempting to publish now with the number they have. While I would caution researchers to attempt to follow appropriate sample size selection protocols whenever possible, I am familiar with the mercenary nature of science and understand that sometimes the paper simply needs to be done. With those things in mind, I recommend the following to the reader: value high-quality participants over high volumes of participants. Even if the number of participants that test your system is insufficient for a proper, statistically significant, power-analyzed effect to be determined, qualitative and mixed methods can still be applied to extract useful information. When deciding between spending more time per participant for higher-fidelity training and testing or collecting more participants at a lower fidelity, lean towards the lower volume, higher quality sample.

Evaluate In-Person and Avoid the Wizard, or Study the Difference

Online studies are tempting, particularly if you mostly use quantitative methods. They seem to offer greater data with lower effort, providing you with huge amounts of information to trawl through. In the field of HRI, however, online studies are often fools' gold. While they have their uses and can be incredibly impactful, it is currently unclear the extent to which online or simulated robot interactions compare to in-person interactions. As such, I would encourage the use of in-person study methodologies and avoiding Wizard-of-Oz studies whenever possible. Wizard-of-Oz aspects in a study are often unavoidable, but everything that the study staff takes over from the robot is another aspect of the interaction that is being lost and cannot be studied. With that in mind, I additionally suggest that the question of whether in-person, online, simulated, or even virtual reality-based robot interactions can be compared to one another, and the effects that these different study environments have on the results of a human study is a topic of great interest and importance, one which could form the basis of an entire thesis on its own. It is however a very difficult problem, with quite an enormous effort required to study.

Record Everything, But Know Your Interest

When conducting in-person studies with robots, ensure that you are recording all data that is reasonable to record and store. Too often I have completed studies, kicking myself internally that I chose not to record a certain ROS topic or camera angle. However, it is also possible to conduct a study with too much data recording, crushing yourself under the weight of data management and storage concerns. This balance is difficult to strike, but as the section heading suggests, you must find a balance between recording every possible piece of information and only recording what you are interested in.

How to Write IRB Protocols

Writing study protocols for the Institutional Review Board (IRB) at your research institution is another difficult task for which many HRI researchers are ill-prepared. Time-consuming, bureaucratic, and frustrating, this procedure is nonetheless absolutely critical to conducting good human research. Without ethical oversight, human studies have

the potential to cause serious harm: physically, mentally, and socially. Beyond this, IRB submissions often require a level of rigor in the submission that will be helpful for you, particularly in your first few human studies. When writing IRB protocols, you must be familiar with which forms you are required to submit and with IRB policies in general. If possible, seek out a course from your institution or work directly with an IRB coordinator to get oriented to which forms you must use. If you cannot find such a course, you will have to learn by doing. Filling out these forms is a long process, so make sure that you set aside plenty of time to complete this. The following recommendations pertain to the writing of IRB protocols and submissions and are focused on general policies that one can apply to their writing rather than specific information about forms.

Writing Defensively, Avoid Pain Points

When writing protocols, it is best to **write defensively**. Avoid terms that will pique IRB concern such as “subject”, and ensure that you include precautions for any risk, no matter how slight, to participant privacy. As you submit protocols and receive feedback, take note of the terms or concepts which caused concern, and incorporate them into your designs for future studies. Be cautious, even to the point of longwindedness, as it may be helpful in getting IRB approval for your proposals. Remember that the majority of IRB-approved studies deal with the testing of drugs and medical equipment and that your lower-risk study is unlikely to be that interesting to IRB professionals, as long as you make their jobs easy. Don’t add unnecessary data collection (another reason to balance the data one records vs the data one needs), particularly if the data to be collected has identifying information in it. Over time, you will learn these pain points and how to avoid them, and your proposals should go from taking months to get approval, to getting approved in a week or two. However, you may also find yourself proposing more and more complex studies, lengthening your approval time once more. Regardless of this interplay, learning to write defensively by avoiding areas of concern and using the expected language and procedures is a key aspect of successfully obtaining IRB approval. While your proposals should be exactly as complex and long as they need to be to achieve your goals, the more frequently you can avoid going to a second round of revisions, the better.

Avoid Unnecessary Specificity, Leave Room

Just as avoiding pain points is a key part of writing proposals defensively, avoiding unnecessary levels of specificity is important in writing IRB proposals. While it is important to know exactly what you plan to do, providing more specificity than required can end up trapping you in a commitment you did not need to make in order to remain compliant with your approved proposal. For instance, if it is not necessary to retain data past the time of the study, do not bother, as the data storage policies may be complex and require further effort and revisions than you need. Again, leaving room for oneself and writing defensively can be conflicting goals, leading to conflict between collaborators or within oneself. It is important to remember that the ultimate goal is always to achieve the research, and if modifications must be made, they must be made. If that delays the process, so be it. When attempting to remain vague to leave yourself the room to change procedures, remember to always ensure two things: firstly, that there is no ethical reason that the IRB needs to know a certain piece of information. For instance, if the detail is whether Python or C++ is used to write a piece of software, this has no ethical bearing on the project and can be omitted. However, if the information relates to participant treatment, say whether or not their name is recorded in the long term, the IRB needs to be given all available details to ensure ethical oversight. Secondly, when considering whether or not to be vague on a detail, ask yourself if you yourself know what that detail is. Do not use vagueness as an excuse to not create the plan for your study, because undefined choices often get made in inconsistent ways, an inconsistency is the enemy of reproducibility. In summary: write defensively and generally, but do have your plans fully fleshed out and ensure that you cover all information that is relevant to ethics, even if the relevancy is not obvious.

Statistical Analysis Tips

Lastly, beyond human study design and writing IRB protocols, the area of human-robot interaction that an engineering degree will not prepare you for is statistical analysis. Most college students have by the level of graduate school encountered statistics at some level at least twice, but never necessarily had to undertake a new analysis of a study that has never been evaluated before. The process of taking observations of data

and using statistics to form concrete statements about the data is one that a statistics class may not actually prepare you for, though you may understand the arithmetic of the field. The recommendations in this section focus on this area, though it is not my specialty.

Learn R or SPSS

Firstly, I highly recommend learning a statistical testing framework. In my opinion, R is the most fully featured, best for graphing, and easiest to learn with any experience in programming, particularly if one is experienced in MATLAB. With such a background, R is quite simple to learn and provides enormous capabilities:

1. The ability to easily visualize any data, in mere seconds. This enables fast inspection of data while trying to find patterns.
2. The ability to create extremely polished and high-quality visualizations for publications. This enables you to only work with one tool for all of the analysis steps, including generating visualizations for publications and presentations.
3. High-efficiency and error-free computation. While it is obviously possible to make mistakes, as long as the inputs to a function are correct, the results can be assumed to be correct. It is therefore important to ensure you understand the use of each function, but you may not need to know how to conduct the actual test yourself.
4. Easily reproducible analysis. When returning to a project after years or even months, it can be difficult to remember what steps were taken to come to the conclusions of a paper. By possessing all of the analysis and visualizations in R, it can be easy to find old results or make corrections and further investigations of the data.

I am less familiar with SPSS, but it, along with other systems, is also available. It is likely that which software you use is less important than how well you know whatever software you are using. This is the case for software for most purposes.

Don't P-Value Hack

While this recommendation comes with significant ethical concerns, it is important to say: don't hack your p-values. This is to say, don't come into your analysis process knowing which tests need high p-values and simply try different tests and configurations until you get a p-value that looks good enough. Even worse, make sure that your alpha level is set prior to analysis. Don't pick what level of p-value you consider statistically significant based on results. These are examples of p-value hacking (modifying data and analysis methods to achieve desirable results) which is not only unethical but bad for scientific rigor. While p-value hacking can be done unintentionally or without realizing it, it is very important to consciously avoid it. If your results are not statistically significant or have sufficient power for their effect to be worth considering, consider qualitative or mixed methods.

Test Assumptions

One way to avoid unintentional p-value hacking is to go by the book when selecting which statistical test to apply to a certain situation. Test all assumptions, including normality of data, equal variances, whether data is ordinal vs categorical, and so on. If a test assumes data normality, even if the p-value of a test indicates significance, the result is invalid. Make sure that you test all of your assumptions, and test them in order.

Use Multiple Raters If Rating Anything

If any data must be transformed by rating it (*e.g.*, correctness of answers) or comes in the form of ratings (*e.g.*, participant ratings of the helpfulness of a robot), use multiple raters and consider their agreement. This is particularly important for data transformed by rating. It can be extremely easy to shift the ratings you give to support a hypothesis, even unconsciously. Have people with less experience and knowledge of the task as raters, have several of them rate the data independently, and check their agreement with an inter-rater reliability score like Krippendorf's alpha or Cohen's kappa. This technique can also be applied to consider answers from participants which are ratings, but this is often less useful.

Beware Repeated Testing

A common source of statistical inaccuracy is family-wise errors introduced by multiple testing. This is referred to as the multiple comparisons problem and is one which is often missed by younger researchers. Consider using the Bonferroni correction or other more modern p-value correction methods, particularly if the software you are using supports it. Even if the software does not support a p-value correct option, Bonferroni correction is easy to apply, as it simply modifies the alpha value for tests that are multiple comparisons.

Beware Contamination and Internal Invalidity

It is extremely easy to contaminate one's own dataset or introduce a threat to internal validity. This is one of those procedures that is essentially an endless responsibility. Often, one will realize a way that a previous study could have been improved several years down the road. Many times, the reason that the test could have been improved can be found in an error that introduces confounding, contaminating variables. Every time you design a study, ask yourself what quantities you can control, and which are important. There are likely to be many variables that simply do not matter (*e.g.*, what food a participant ate that morning), and only a few which do (*e.g.*, how a participant's hearing or vision performs in terms of colorblindness, hearing loss, etc.). Finding them can be a difficult task, but try your best to keep adding ideas to your list of variables to check, control, or ignore. As you develop a set of common methodologies, you can often find ways to improve your studies by growing a continuing set of variables that can be managed, producing better, more clearly understandable, and more significant results.

statcheck.io

A final recommendation: R has a statistical checking package called **statcheck**, which has been extended into a website called **statcheck.io**. This tool and website can help you determine if your paper is reporting statistical tests correctly, and if the results presented are consistent with one another. Statcheck has two primary weaknesses:

1. Statcheck cannot identify tests that are reported in an entirely unknown way. You can check if each test is listed in the statcheck results, but if you are not careful,

you can entirely miss evaluating a reported test.

2. Statcheck also cannot determine if there are issues in your assumptions or conclusions based on your tests. It is possible that at some point in the near future, there will be a large-language model which can actually do this, but I would take its results with a large grain of salt until they have been proved consistent.

Despite these two weaknesses, `statcheck` and `statcheck.io` are two extremely useful tools, particularly if they are correctly applied, and not misused or taken to be more powerful than they are.

Appendix III: Marine Debris Detection

Underwater trash poses a significant threat to the Earth’s entire ecosystem, from the direct effects on underwater flora and fauna to the effects on terrestrial life, including human health and safety. Plastic trash causes particularly serious issues in our ecosystem, as it does not decompose into organic material but rather disintegrates into microplastic beads, which are ingested by organisms. Once the trash has disintegrated, it becomes nearly impossible to collect due to its size, making cleanup at an early stage of degradation extremely important. While private organizations and governments have made collaborative efforts to collect the trash, they have been only marginally successful due to the scale of the problem. A fleet of AUVs equipped with the ability to **detect**, **localize**, and **manipulate** underwater trash would be able to approach the problem of trash removal at scale, without the risk to human life inherent in diving operations. The first of the required capabilities for such an AUV, the ability to detect trash, can be addressed using visual trash detection methods. In this appendix, we present two pieces of work supporting the visual detection of trash in underwater environments: the Trash-ICRA19 dataset and the TrashCan dataset along with the algorithms trained on it. These datasets, which are some of the first of this kind, have enabled the development of visual trash detection for underwater robots. This has begun the important work of building capabilities for AUVs that will allow them to find and one day remove trash from underwater environments.



Figure A3.1: Sample images showing underwater trash made of plastic and metal at different stages of shape and color deformation.

The Trash-ICRA19 Dataset

The Trash-ICRA19 dataset was sourced from the work of the Global Oceanographic Data Center, part of the Japan Agency for Marine Earth Science and Technology (JAMSTEC). JAMSTEC has made a dataset of deep-sea debris available online as part of the larger J-EDI (JAMSTEC E-Library of Deep-sea Images) dataset [292]. This dataset has unannotated images dating back to 1982 and provides debris data in the form of short video clips. The videos that comprise that dataset vary greatly in quality, depth, objects in scenes, and the cameras used. They contain images of many different types of marine debris, captured from real-world environments, giving us a variety of objects in different states of decay, occlusion, and overgrowth. Additionally, the clarity of the water and the quality of the light varies significantly from video to video. This allows us to create a

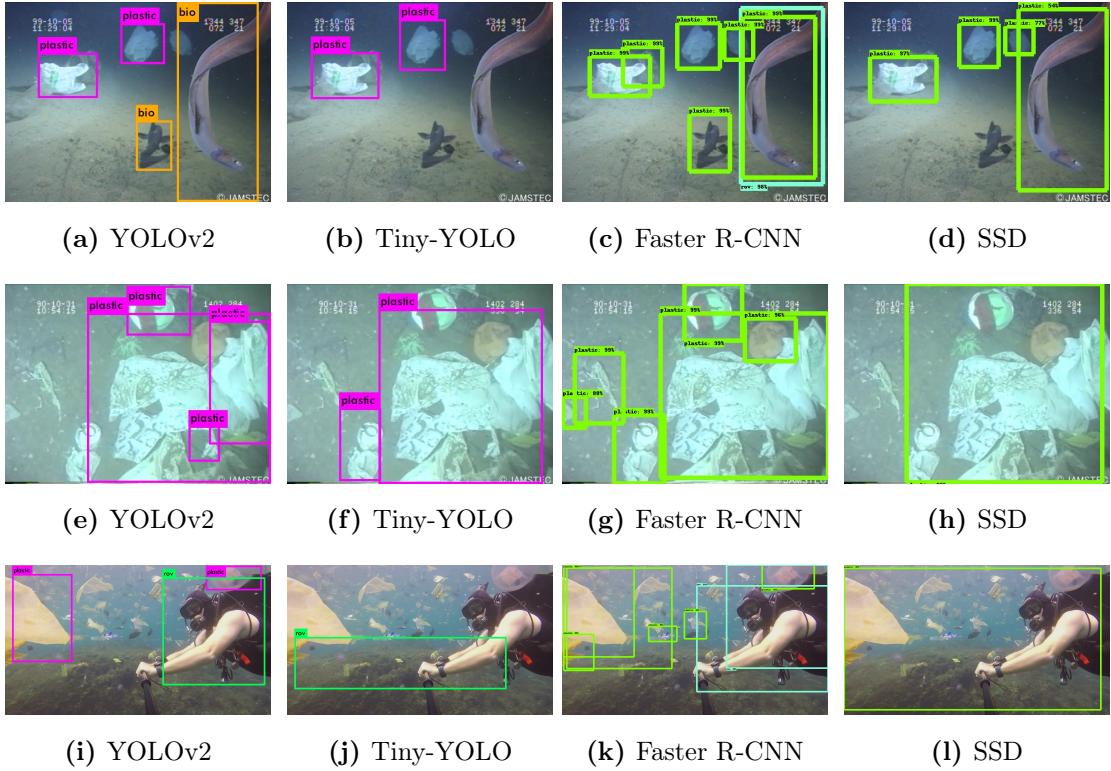


Figure A3.2: Example detection results. In images A3.2a-A3.2d it can be seen that while all networks detect some plastic objects, only YOLOv2 (A3.2a) correctly identifies the fish in the scene as bio and Faster-RCNN (A3.2c) detects more individual plastic items than any other network. In images A3.2e-A3.2h, the same evidence of Faster-RCNN’s (A3.2g) ability to detect more individual objects can be seen.

dataset for training that closely conforms to real-world conditions, unlike previous works, which mostly rely on internally generated datasets that are limited in the represented visual conditions and levels of decay of the trash.

We selected all videos which appeared to contain some kind of plastic debris from JAMSTEC data collected between 2000 and 2017. This was done in part to reduce the problem to a manageable size for our purposes, but also because plastic is an important type of marine debris [293]. At this point, every video was sampled at a rate of three frames per second to produce images that could be annotated to prepare them for use in learning models. These images were manually annotated with the following labels:

Table A3.1: Detection metrics in mAP, IoU, and AP

Network	mAP	Avg. IoU	plastic AP	bio AP	rov AP
YOLOv2	47.9	54.7	82.3	9.5	52.1
Tiny-YOLO	31.6	49.8	70.3	4.2	20.5
Faster R-CNN	81.0	60.6	83.3	73.2	71.3
SSD	67.4	53.0	69.8	6.2	55.9

Table A3.2: Performance metrics in frames per second

Network	1080	TX2	CPU
YOLOv2	74	6.2	0.11
Tiny-YOLO	205	20.5	0.52
Faster R-CNN	18.75	5.66	0.97
SSD	25.2	11.25	3.19

plastic (any instance of plastic debris), *bio* (flora or fauna), and *rov* (remotely operated vehicles or other human-made equipment). The annotating process was completed by a number of volunteers who used the freely available LabelImg tool. The final training dataset used in this work was composed of 5,720 images, with dimensions of 480x320. This dataset is publicly available online at <https://conservancy.umn.edu/handle/11299/214366> and has been downloaded over 8,500 times as of March 2023.

Detecting Trash on Trash-ICRA19

Following the creation of this dataset, we trained four state of the art deep neural networks (YOLOv2, Tiny-YOLOv2, Faster-RCNN, and SSD) on that dataset. These networks were then tested by running inference on the test portion of our dataset and calculating standard performance metrics. Examples of detection from each network can be seen in Fig. A3.2, with detection accuracy metrics in Table A3.1. In addition, all four networks were evaluated in terms of inference runtime on three devices: a desktop GPU (Nvidia 1080), a mobile GPU (Nvidia TX2), and a CPU. The results of this can

be seen in Table A3.2.

Each network performs within the established abilities of the network on other similar tasks. Overall, YOLOv2 and Tiny-YOLO have lower mAP when compared to Faster R-CNN and SSD. Conversely, Faster R-CNN and SSD have higher processing times, as seen in Table A3.2. These traits are well known, and their performance remains consistent in this application. This trade-off between mAP and FPS does not affect IoU, however. All four network architectures have similar IoU values, meaning that none are the clear victor in terms of how accurate their bounding boxes are. The most substantial result from this evaluation is that deep-learning based visual object detection methods can plausibly be used to detect marine debris in real-time. In terms of which method would be ideal for underwater trash detection, Faster R-CNN is the obvious choice purely from the standpoint of accuracy but falls behind when considering inference time. YOLOv2 strikes a good balance of accuracy and speed, while SSD provides the best inference times on CPU. If performance is the primary consideration, however, Tiny-YOLO outpaces all other algorithms significantly on the TX2, the most realistic hardware for a modern AUV.

The TrashCan Dataset

To provide an improvement on the level of detail provided by the Trash-ICRA19 dataset, we present the TrashCan dataset. Comprised of over 7,000 annotated images containing observations of trash, ROVs, and a wide variety of undersea flora and fauna, this dataset provides much more detailed information about each object in the image. The annotations in this dataset take the format of instance segmentation annotations: bitmaps containing a mask marking which pixels in the image contain each object. While datasets have previously been created containing bounding box level annotations of trash in marine environments, including one of our own creation [294], TrashCan is, to the best of our knowledge, the first instance-segmentation annotated dataset of underwater trash. Additionally, TrashCan’s class mappings provide more semantic information about the material and type of each trash object.

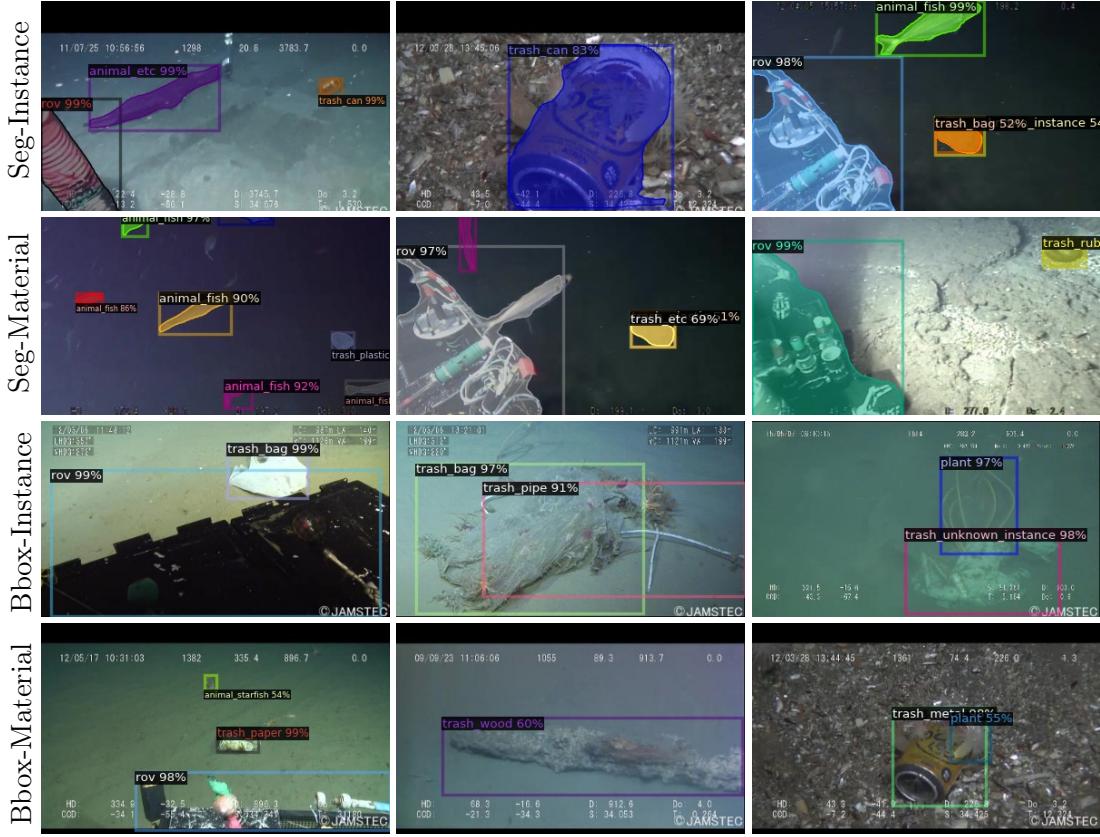


Figure A3.3: Sampled images from segmentation and object detection evaluations.

Dataset Source and Composition

The imagery in TrashCan is sourced from the same J-EDI dataset [292], used as the primary source for our first dataset. A small portion of these videos contain observations of marine debris, and it is from these that all of our trash data is sourced, nearly one thousand videos of varying lengths. Some, but not all of these videos, were contained in the original trash dataset. In addition to the videos of marine debris, additional videos were selected to diversify the biological objects present in the dataset.

Annotation Process and Tools

Once the videos had been selected, frames were extracted from each video at a rate of one frame per second. Once this was done, the videos were combined into similarly-sized

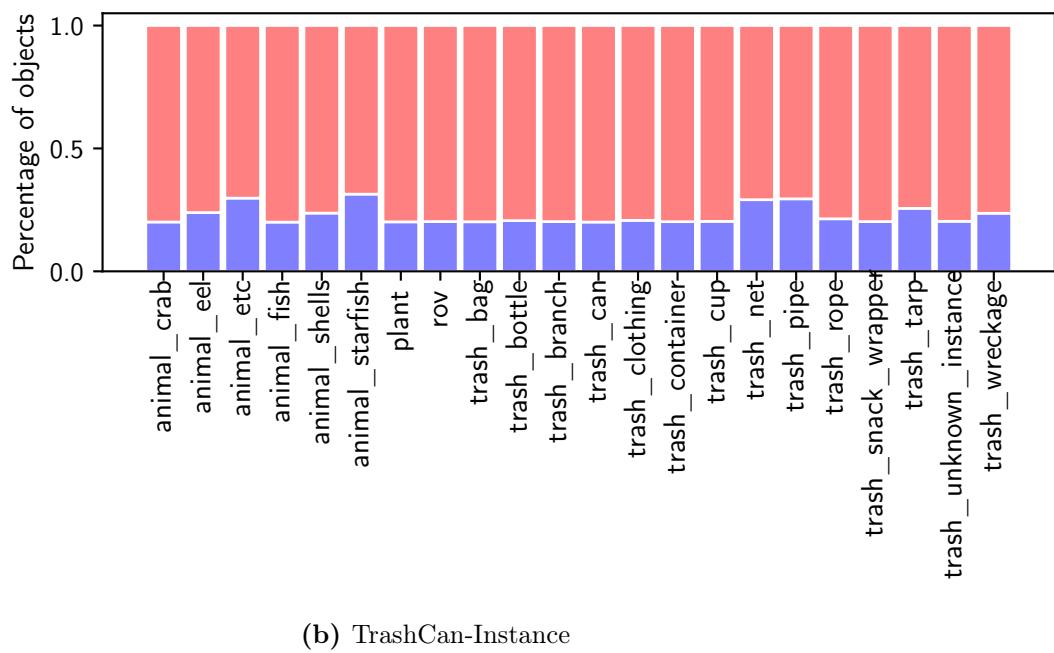
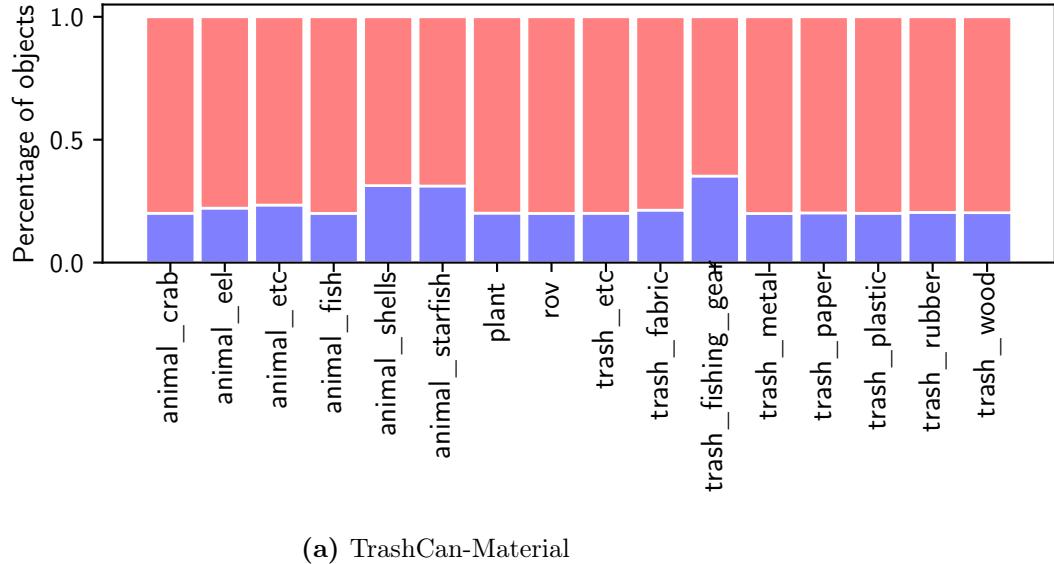


Figure A3.4: Data split between training (pink) and validation (blue) sets per object for the two versions of the TrashCan dataset.

portions and uploaded to Supervisely [295], an online image annotation tool. Once uploaded, the images were annotated by a team of 21 people, one image at a time. This took approximately 1,500 work hours over the course of several months. Next, an annotator drew a segmentation mask over it, marking it as one of four classes: *trash* (any marine debris), *rov* (any man-made item intentionally placed in the scene), *bio* (plants and animals), and *unknown* (used to mark unknown objects). Trash objects were additionally tagged by material (*e.g.*, , metal, plastic), instance (*e.g.*, , cup, bag, container), along with binary tags indicating overgrowth, significant decay, or crushed/broken items. Bio objects were tagged as either plant or animal, and in the case of animals, given a tag with the type (*e.g.*, , crab, fish, eel). ROV and unknown class objects required no additional tags.

Object Class Versions

To prepare the dataset for use in training deep networks, the annotations were converted from a custom JSON format to the COCO format [296]. We converted all objects into one of two dataset versions: TrashCan-Material and TrashCan-Instance, so named for the tag data used to differentiate between different types of trash. In the material version, every trash object was given a class name following the pattern *trash_[material_name]* (*e.g.*, , *trash_paper*, *trash_plastic*), as long as the given material had more than 50 objects in the dataset. Those with fewer examples were given the class *trash_etc*, the same class used by annotators when the material of the object was unknown. Similarly, for the TrashCan-Instance version, trash classes were generated using instance tags which approximated the type of object that was being annotated (*e.g.*, , *trash_cup*, *trash_bag*). The same cutoff of 50 objects was used, with the catch-all class being *trash_unknown_instance*. In both versions, any object labeled with the *unknown* class was typically added to the catch-all *trash* class. The ROV class remained singular for both versions, while bio objects were either transformed into *plant* or *animal_[animal_type]* (*e.g.*, , *animal_starfish*, *animal_crab*) classes, based on tags applied to the object. Animal tags that had been applied to a small number of instance labels were not given their own class but were rather combined into the *animal_etc* class. The classes of both versions, with their distribution into the training or test sets are given in Fig. A3.4.

Table A3.3: Performance metrics in frames per second

Network	Titan V	TX2
Mask R-CNN	6.25	0.22
Faster R-CNN	6.67	0.23

Table A3.4: Overall metrics for each combination of the TrashCan dataset and model

Method	Dataset	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
Mask R-CNN	Instance	30.0	55.3	29.4	23.2	31.7	48.6
Mask R-CNN	Material	28.2	54.0	25.6	24.1	28.7	41.8
Faster R-CNN	Instance	34.5	55.4	38.1	27.6	36.2	51.4
Faster R-CNN	Material	29.1	51.2	27.8	28.2	30.2	40.0

Detecting and Segmenting Trash

We present experiments with state-of-the-art instance segmentation and object detection models using two example datasets to provide a baseline for future model development. For the following experiments, we use the Pytorch Detectron2 [297] library and metrics introduced by the COCO dataset to establish a baseline evaluation.

Detection Experiments

For object detection, we employed Faster R-CNN with a ResNeXt-101-FPN (X-101-FPN) [298] backbone. The model was trained on a pre-trained model with the COCO dataset with an Nvidia Titan XP.

Segmentation Experiments

Mask R-CNN with X-101-FPN was chosen for the instance segmentation task. The model was initialized with weights from the COCO dataset and trained on an Nvidia Titan V.

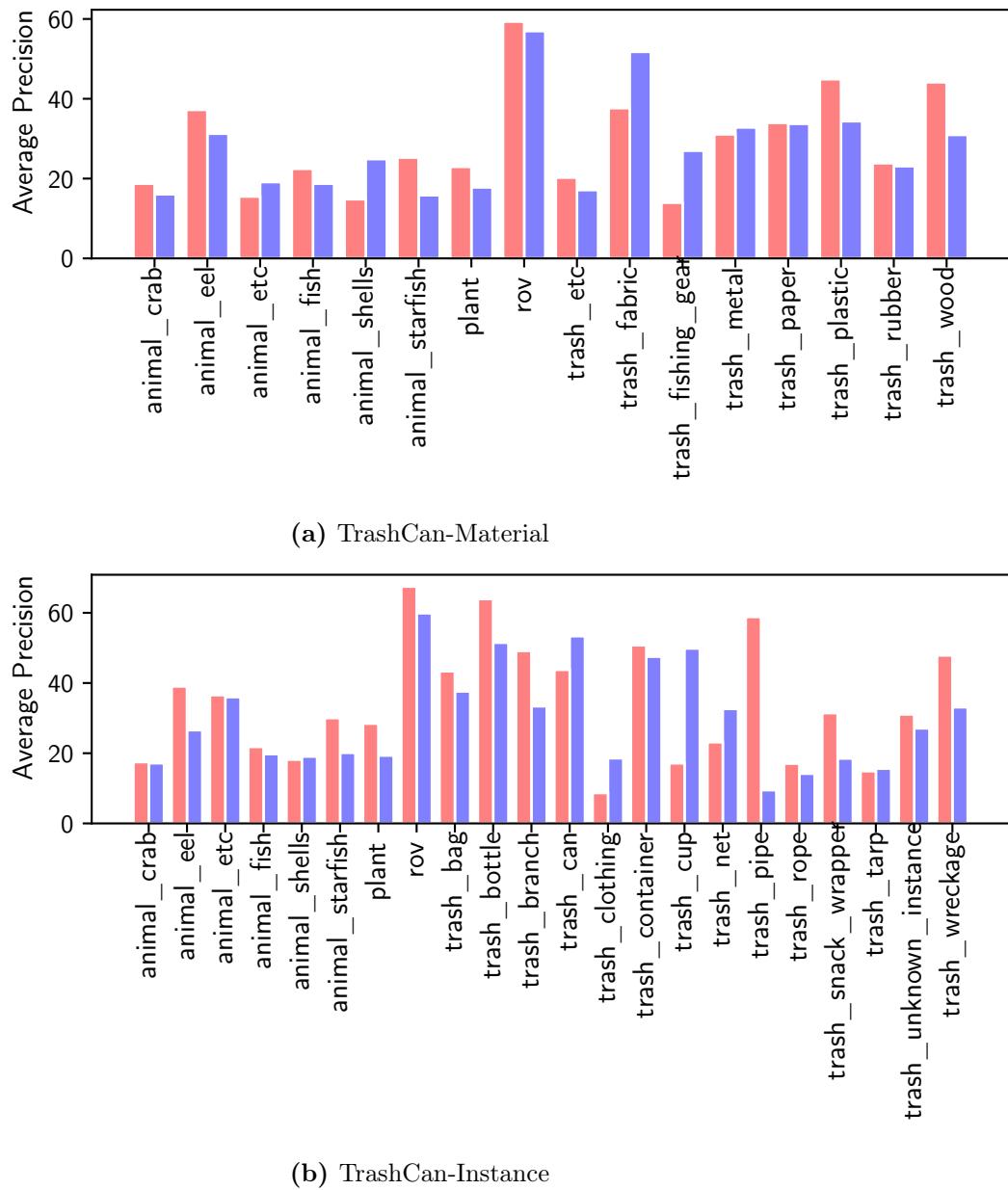


Figure A3.5: Results from Faster R-CNN (pink) and Mask R-CNN (blue) in terms of per-class average precision.

Results

Table A3.4 shows evaluation metrics of object detection and instance segmentation tasks for each dataset, with per-class results in Fig. A3.5. We also provide inference times for both tasks in Table A3.3. Fig. A3.3 displays sampled results from object detection and instance segmentation models trained with both versions of the datasets. From our experiments, we find the following:

- For both object detection and instance segmentation tasks, the models trained with the instance version dataset achieve higher AP in general. We believe this is because more visually similar objects are grouped into the classes: while most cans look similar, not all metal objects look similar.
- The accuracy of object detection and instance segmentation models is lower than that of our original baseline models, trained on Trash-ICRA19. This is because the TrashCan dataset expands the problem of trash detection from a focus simply on plastic items to a much wider variety of object types, which greatly increases the challenge of the task.

Although the baseline metrics are acceptable considering the challenging nature of this dataset, there is room for improvement in future work, either by increasing the size of the dataset or by employing more advanced models. It is our hope that the release of this dataset will facilitate further research on this challenging problem, bringing the marine robotics community closer to a solution for the urgent problem of trash detection and removal. TrashCan 1.0 is available for public use at <https://conservancy.umn.edu/handle/11299/214865> and has been downloaded over 29,000 times as of March 2023.

Appendix IV: Depth-Based Bayesian Localization

Localization is a capability key to the effective operation of AUVs in underwater environments. Without the ability to accurately determine its position in the environment it is operating in, an AUV will only be able to operate within the visual range of a diver, as it will not have the ability to navigate independently. In a broad sense, there are three major techniques used to address the problem of localization for underwater robots: combining inertial data with dead reckoning [299], acoustic transponder-based approaches [300], and landmark-based (also known as *geophysical feature*) approaches (*e.g.*, [301, 302]).

The first of these techniques uses an inertial measurement unit (IMU) and a velocity measurement sensor (*e.g.*, a Doppler velocity log (DVL)) to estimate the position of the robot by correcting the IMU’s drift with velocity information. While this approach is widely adopted, it often struggles with accuracy drifting over time and requires expensive, high-accuracy IMUs and DVLs (*e.g.*, [303–305]). Techniques which use acoustic transponders include long baseline (LBL) [306], ultra-short baseline (USBL) [307], and short baseline (SBL) systems, all of which are highly accurate and fast. However, most of these techniques require either a surface ship carrying a *transponder* or pre-installed beacons on the floor of the water body in question. Additionally, the devices required are quite expensive (at least 10,000 USD). In many applications, it is impractical or impossible to install these devices in the water body due to the added cost or environmental constraints. Lastly, landmark-based methods use visual and acoustic sensors to detect known features in the marine environment, usually on the floor, which can be

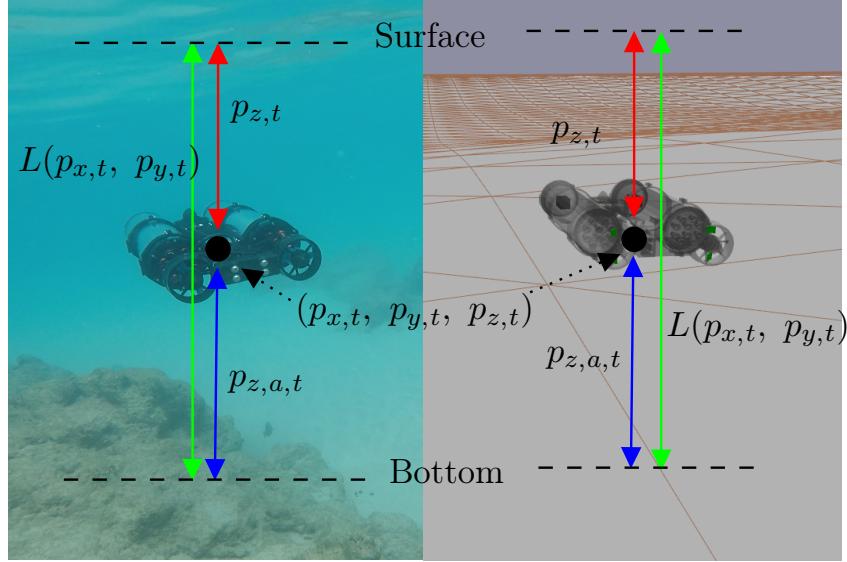


Figure A4.1: Visual representation of AUV location in a body of water. The surface and bottom of the body are indicated by the dashed black lines. $L(p_x,t, p_y,t)$: altitude, p_z,t : depth, $p_{z,a,t}$: range measurement, (Left): LoCO deployed in an open water environment, (Right): LoCO simulated in Gazebo.

used by AUVs to localize themselves relative to the landmarks. However, optical distortions such as scattering, absorption, and attenuation (*e.g.*, arising from turbidity in the water) can be extreme, resulting in features only being visible at close range. A lack of clear visuals can make it challenging to use vision-based methods for broad-area localization. Landmark-based methods with an acoustic sensor provide a practical means to tackle underwater localization problems; however, landmarks must be precisely located via acoustic means, which is often not feasible due to environmental factors and sensor limitations. In contrast to these three common techniques, the methods presented in this appendix require only a depth (pressure) sensor and sonar altimeter, which are both inexpensive (depth sensors generally cost less than 100 USD and altimeters 300 - 500 USD). Additionally, no installation of transponders or beacons is needed, and issues of visual distortion do not affect our approaches in the slightest. This significantly reduces the cost of the required hardware for AUV localization, as well as eliminating costly infrastructure requirements and the need for clear visual conditions.

The approach introduced in this appendix is a subset of Terrain-Based Navigation (TBN) [308], which is widely used across domains and refers to a general set of localization approaches using *a priori* maps. We develop four Bayes filter-based localization methods which utilize bathymetry data as the *a priori* map and serve as a low-cost alternative to the previously discussed methods (inertial, acoustic transponder, and landmark-based approaches). Bathymetry data is a measurement of the altitude of the water column at every (x, y) location on the surface of the water body, with some defined resolution in the (x, y) plane. Often, the data is quantized in the form of rectangular regions, or *grids*, with every (x, y) location in a single grid having the same altitude. For the rest of the appendix, the term *altitude* will refer to the total depth of the water body from the floor to the surface (shown as $L(p_{x,t}, p_{y,t})$ in Fig. A4.1). Although the depth of the robot can vary, the altitude of the water column remains relatively constant (not considering wave effects) at a specific grid location. An AUV needs both a depth sensor and a sonar altimeter to determine the total altitude of the water column at its location and its position in the column. We attempt to quantify the efficacy of bathymetry-based AUV localization algorithms that use depth data from a pressure sensor and range measurement data from a single-beam sonar as inputs; specifically, we evaluate the localization performance of four Bayes filter-based methods: the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), Particle Filter (PF), and Marginalized Particle Filter (MPF). The EKF and UKF are parametric implementations of the Bayes filter algorithm with Gaussian assumptions, and the PF is a nonparametric implementation. The MPF, otherwise known as the Rao-Blackwellized Particle Filter (RBPF), is a “hybrid” approach that combines the Kalman Filter (KF) and the PF [309].

Background: Underwater Localization

Underwater localization using landmark-based methods with acoustic sensors has been widely studied. For these methods, ranging-type sonars including the single-beam profiling and multi-beam varieties have been used [302]. Multi-beam and profiling sonars collect multiple measurements, and they can give more accurate results than single-beam sonars. Table A4.1 summarizes selected existing localization algorithms, along with the

	Sensors	Parameters of state vector s	Algorithms
Teixeira et al. [310]	DVL,Single-beam sonar	$b = (b_x, b_y)s = (x, y, b)$	PF
Fairfield and Wettergreen [311]	Multi-beam sonar	$q = (\phi, \theta, \psi, x, y, z) s = (q, \dot{q}, \ddot{q})$	PF
Ura et al. [312]	Profiling sonar	$s = (x, y)$	PF
Williams and Mahon [313]	Single-beam sonar	$q = (x, y, z)s = (q, \dot{q})$	PF
Meduna et al. [314]	Single-beam sonar	$s = (x, y)$	PMF
Kim and Kim [315]	Single-beam sonar	$p = (\phi, \theta, \psi)q = (u, v, w)s = (x, y, z, p, q)$	MPF
Ours	Single-beam sonar	$s = (x, y, z)$	EKF, UKF, PF, MPF

Table A4.1: Selected existing localization algorithms

parameters of their state vector. The vector (ϕ, θ, ψ) represents the Euler angles, (u, v, w) is the AUV velocity in the body-fixed frame, and (b_x, b_y) is the velocity bias.

Although DVL, multi-beam sonar, and profiling sonar-based methods yield better results than those with single-beam sonar (*e.g.*, [310–312, 316]), such sensors can be prohibitively expensive. Single-beam sonars use a narrow acoustic projection to measure depth to the floor and are thus vulnerable to noise. However, they have been widely adopted to solve localization problems since they are among the most affordable acoustic sensors [305]. Williams and Mahon [313] proposed a localization algorithm based on the PF, but the computational burden of the algorithm is heavy. Meduna et al. [314] presented a point mass filter (PMF)-based algorithm, but it is limited to the (x, y) positions of an AUV. Kim and Kim [315] used a single-beam sonar with the MPF and estimated the 6-DOF position and orientation of an AUV along with the velocity. However, the algorithm requires a highly-accurate IMU, which can be expensive.

Several Bayes filter-based methods have been used to solve the localization problem [308] with sonar data. Among Bayes filters, the EKF and UKF have seen the most use in this domain (*e.g.*, [317, 318]). Karimi et al. [304] showed that the EKF can outperform the UKF when they estimate the position of an AUV by the sensor readings from DVL and inertial navigation system (INS). However, the UKF captures nonlinearity up to the second-order term in the state transition process [302], which in theory could outperform the EKF in similar applications. We thus compare the performance of EKF and UKF-based algorithms in AUV localization. Although the EKF and UKF exhibit high accuracy under Gaussian assumptions, they often fail to converge when the underlying distribution is multi-modal. The inherent nonlinearity of the underwater terrain and of AUV motions underwater makes it challenging for these methods to work reliably. To

address these issues, the PF has been widely used (*e.g.*, [310, 312, 313, 316, 319–325]). However, the PF is computationally expensive and thus can be difficult to run on board AUVs for real-time localization. The MPF, on the other hand, has a lower computational cost and provides similar benefits to the PF, handling nonlinearity to some extent [326], thus making it potentially useful for underwater localization (*e.g.*, [315, 327]). Despite the strong potential shown by the above studies, localization with bathymetry data considering 3-DOF state vectors and using four Bayes filter-based algorithms (EKF, UKF, PF, and MPF) is yet to be extensively studied. The existing studies were conducted on diverse types of robots, and it is infeasible to do a fair comparison of the algorithms.

Problem Formulation

Localizing an AUV is a challenging problem. Given bathymetry data of a target environment and measurements from AUV onboard depth and single-beam sonar sensors, we aim to localize an AUV with Bayesian filters. Bayesian filters require motion and measurement models to estimate system state using the well-known propagate-predict-update process. We propose linear and mixed AUV motion models for updating the AUV position and a measurement model for collecting the depth and range measurements at each location. The motion models and measurement model are subsequently used to obtain AUV state estimates. We specifically propose the two motion models to evaluate each Bayesian filter for estimating both linear and nonlinear states. We use these models to analyze the performance of each filter in different water environments.

Motion Model

A general discrete time state-space model can be represented as Eq. A4.1 to formulate the AUV localization problem where x_t is a state vector, u_t is a control input, and y_t is a measurement. Only the 3D position of an AUV is included in the state vector. f and h can be either linear or nonlinear functions. q_t and r_t represent the noise from motion and measurements. We assume the noise is normally distributed [310, 314, 315]. The model in Eq. A4.1 is used for the EKF, UKF, and PF-based localization algorithms.

The model for the MPF-based localization algorithm is introduced in Section 10.6.

$$\begin{cases} x_t = f(x_{t-1}, u_t) + q_t \\ y_t = h(x_t) + r_t \end{cases} \quad (\text{A4.1})$$

The state vector and control inputs are defined as follows:

$$x_t = [p_{x,t} \ p_{y,t} \ p_{z,t}]^T \quad (\text{A4.2})$$

$$u_t = [v_{x,t} \ v_{y,t} \ v_{z,t}]^T \quad (\text{A4.3})$$

where $\mathbf{p} = [p_{x,t} \ p_{y,t} \ p_{z,t}]^T$ are the x , y , and z components of the robot's *position* in three-dimensional space at time t . We also use the term ‘pose’ to refer to the state vector x_t . The AUV motion models are defined in Eqs. A4.4 and A4.5.

Linear motion model

All state variables are updated linearly.

$$f(x_t, u_t) = x_t + u_t * dt \quad (\text{A4.4})$$

Linear/Nonlinear mixed motion model

Underwater current often causes an AUV to deviate from its original trajectory, which generates irregular movements of the AUV. To simulate such movements without using the Euler angles [328] explicitly, we propose a linear/nonlinear mixed motion model using the bathymetry data. Among the three state variables in the state vector (Eq. A4.2), $p_{x,t}$ and $p_{y,t}$ are updated based on the altitude $L(p_{x,t}, p_{y,t})$ at position $(p_{x,t}, p_{y,t})$. Therefore, this model will cause an AUV to have faster motions in x and y directions from positions with higher altitudes. Unlike the other two variables, the state variable $p_{z,t}$ is updated linearly as in the linear motion model. In Eq. A4.5, a , a_d , a_{off} , b , b_d , and b_{off} are constants specific to each water body such that the AUV is able to navigate in each without collision with their “shorelines”.

$$f(x_t, u_t) = \begin{bmatrix} p_{x,t} + a \left[\frac{L(p_{x,t}, p_{y,t})}{a_d} + a_{off} \right] dt \\ p_{y,t} + b \left[\frac{L(p_{x,t}, p_{y,t})}{b_d} + b_{off} \right] dt \\ p_{z,t} + v_{z,t} * dt \end{bmatrix} \quad (\text{A4.5})$$

Measurement Model

The measurement function h is the same for both the linear and mixed models.

$$h(x_t) = \begin{bmatrix} p_{z,t} & p_{z,a,t} = L(p_{x,t}, p_{y,t}) - p_{z,t} \end{bmatrix}^T \quad (\text{A4.6})$$

L is bathymetry data (depths for each x and y location on a grid), $p_{z,t}$ represents the depth of the AUV from the surface measured by the pressure sensor, and $p_{z,a,t}$ represents the range measurement of the single-beam sonar on the AUV. Therefore, the sum of $p_{z,t}$ and $p_{z,a,t}$ is the altitude $L(p_{x,t}, p_{y,t})$ at the position $(p_{x,t}, p_{y,t})$ as shown in Fig. A4.1.

Localization Methods

To solve the underwater robot localization problem under highly nonlinear perturbation of AUV motion (Eq. A4.5), nonlinear Bayes filter algorithms are essential. The EKF and UKF are widely used to handle nonlinear state estimation with the assumption that the state variables follow the Gaussian distribution, but they could fail when the distribution is not Gaussian [309]. The PF [319] is resilient to various types of noise, but it is computationally expensive. The MPF [325] uses the PF for nonlinear state variables and the KF for linear state variables because the KF is an optimal filter for estimating linear state variables.

Extended Kalman Filter

To approximate a nonlinear system, the EKF takes the first-order of the Taylor series expansion [329]. Linear matrices in the KF are replaced with the Jacobians to make predictions. The Jacobians for the mixed motion model in Eq. A4.5 and the measurement model in Eq. A4.6 are shown in Eq. A4.7 and Eq. A4.8, respectively. The EKF requires the following inputs: state vector, state covariance, control input, and measurements. With these inputs, the EKF uses the following well-known two-step approach:

- *Prediction Step:* In this step, the state vector is updated based on the motion model and control input. Once it is updated, the Jacobian, state covariance, and noise covariance are used to find the state vector and state covariance matrix for the next time step.

- *Correction Step:* The Kalman gain is calculated using the Jacobian, state covariance, and measurement noise covariance matrices. The calculated Kalman gain is then used to refine the state vector from the prediction step.

The EKF assumes that state variables follow a Gaussian distribution with a single mode. Due to this assumption, the EKF is computationally efficient, and each update takes $O(d^3)$ where d is the dimension of the state x_t [330]. The EKF is directly applied to Eq. A4.1 for linear and mixed motion cases to localize AUVs using the depth and range measurements.

$$F_t = \begin{bmatrix} 1 + \frac{a}{a_d} \left[\frac{\partial L(x,y)}{\partial x} \right] dt & \frac{a}{a_d} \left[\frac{\partial L(x,y)}{\partial y} \right] dt & 0 \\ \frac{b}{b_d} \left[\frac{\partial L(x,y)}{\partial x} \right] dt & 1 + \frac{b}{b_d} \left[\frac{\partial L(x,y)}{\partial y} \right] dt & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (\text{A4.7})$$

$$H_t = \begin{bmatrix} 0 & 0 & -1 \\ \frac{\partial L(x,y)}{\partial x} & \frac{\partial L(x,y)}{\partial y} & 1 \end{bmatrix} \quad (\text{A4.8})$$

Unscented Kalman Filter

The UKF [331] is another approach to estimate a nonlinear system using the Unscented Transform instead of the Taylor series. It uses a sampling approach to capture the mean and covariance of Gaussian random distributions. Then, the UKF propagates the points through the true nonlinear system. In this way, the UKF can handle higher degrees of non-linearity than the EKF. The UKF requires the same inputs as the EKF, along with some additional parameters, namely n , α , β , and κ , described below. Like the EKF, the UKF has a two-step estimation process:

- *Prediction Step:* The UKF chooses $2n+1$ sigma points from the Gaussian distribution where n is the number of dimensions. The UKF then passes these sigma points through the motion model f . The parameters α , β , and κ are used to determine weights for each sample and the distribution of the sigma points.
- *Correction Step:* The state vector from the prediction step is used to generate sigma points. Their measurements, noise covariance, and state covariance matrices are used to calculate the Kalman gain. As in the EKF, the state vector is updated

using the Kalman gain. With this, the UKF approximates a Gaussian distribution with the sigma points.

In this way, the UKF can create a more accurate approximation of the Gaussian distribution than the EKF. Since the estimation is based on the Gaussian distribution assumption, it may not work for multi-modal distributions. In most cases, the UKF shows notable improvements compared to the EKF. The computational complexity of the UKF is close to the EKF [330]. The UKF is also directly applied to Eq. A4.1 for linear and mixed motion cases.

Particle Filter

The PF [322] implements the Bayes filter algorithm using sequential Monte Carlo methods. Unlike the EKF and UKF, the PF does not require any assumptions regarding the distribution. Instead, it uses N *particles*, representing potential locations of the state x_t , to approximate the distribution of the state x_t . The more particles there are, the more accurate the approximation of the distribution. The PF can handle distributions with high nonlinearity and multiple modes. However, the computational complexity of the PF for each update is $O(Nd^2)$ where d is the dimension of the state x_t , meaning that the runtime of the PF scales linearly with the number of particles used. When N is much larger than d , the PF can be much slower than the EKF and UKF [324], which is the situation in our specific case. This computational burden is often the main drawback of the PF for real-time implementation.

We propose a depth-based PF localization (d-PFL) in Algorithm 1; *i.e.*, the PF uses the AUV’s *depth* at each iteration as its *measurement*. **PFL_update()** takes a bathymetry map, depth measurements, and control inputs and returns propagated particles representing the AUVs position in three dimensions and their weights. To evaluate weights w_t , we use the multivariate Gaussian distribution as shown in Eq. A4.9.

$$w_t = w_{t-1} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \quad (\text{A4.9})$$

During the update process, only the particles not over the “shoreline” will have non-zero weights (*i.e.*, inside a bathymetry map). Once the weights for the particles

Algorithm 1 Depth-based PF Localization

```

1: D-PFL main
2:  $L$  = Target waterbody
3:  $N$  = The number of particles
4:  $x_{init}$  = Initial pose of a robot
5:  $x_1 = \text{Initialize\_around\_pose}(L, N)$ 
6:  $z_t$  = Sensor measurements.
7:  $u_t$  = Control input
8: for  $t = 1, \dots, T$  do
9:    $x_p = x_t$ 
10:  for  $m = 1, \dots, N$  do
11:     $x(m, :) = \text{motion\_update}(u_t, x_p(m, :), L)$ 
12:     $w(m) = \text{sensor\_update}(z_t, x(m, :), L)$ 
13:  end for
14:   $w_{total} = \text{sum}(w)$ 
15:  for  $m = 1, \dots, N$  do
16:     $w(m) = w(m)/w_{total}$ 
17:  end for
18:   $x_t = \text{resample\_particles}(x(m, :), w, L, rand)$ 
19:   $w_t = w$ 
20:   $est\_pose = \text{PFL\_get\_pose}(x_t, w_t)$ 
21: end for
22: return  $est\_pose$ 
  
```

are calculated, they are normalized to ensure that they sum to 1. Then, particles are resampled based on their weights. To avoid a situation where all the particles are trapped in some local maxima in a bathymetric environment with similar or repeating profiles, some of the N particles are sampled randomly at each time step. Although this can degrade the accuracy of the algorithm, it decreases the chance of incorrect estimations occurring. The pose of the AUV is estimated (Eq. A4.10) once the propagated particles and the corresponding weights are updated.

$$\hat{x} = \sum_{m=1}^M w^{[m]} x^{[m]} \quad (\text{A4.10})$$

Marginalized Particle Filter

The MPF was proposed to reduce the computational complexity of a particle filter while retaining a similar performance when the model has a linear substructure [309]. The core idea of the MPF is to *marginalize* linear state variables from the state vector and use the KF to estimate those variables. The PF is then used to estimate the remaining nonlinear state variables. The computational complexity of the MPF is defined in [332] and can be simplified to $O(d_n^3 N)$ for our case where d_n is the dimension of the nonlinear state variables.

To apply the MPF [326], the model in Eq. A4.1 is separated into linear and nonlinear state variables as shown in Eq. A4.20 and A4.21. The motion model noise q_t^n , q_t^l , and the measurement model noise r_t are assumed to be Gaussian with zero mean. The matrices A , C , and G are determined by the motion model (Algorithm 2).

$$x_t = \begin{bmatrix} x_t^n \\ x_t^l \end{bmatrix} \quad (\text{A4.20})$$

$$\begin{cases} x_{t+1}^n = f_t^n(x_t^n) + A_t^n(x_t^n)x_t^l + G_t^n(x_t^n)q_t^n \\ x_{t+1}^l = f_t^l(x_t^n) + A_t^l(x_t^n)x_t^l + G_t^l(x_t^n)q_t^l \\ y_t = h_t(x_t^n) + C_t(x_t^n)x_t^l + r_t \end{cases} \quad (\text{A4.21})$$

In our case, the ratio $\frac{N(k)}{N_{PF}}$ is 1.1 where $N(k)$ is the number of particles that can be used for the MPF and N_{PF} is the number of particles used for the standard PF. This ratio means that the MPF can use 10% more particles than the PF while having the same computational complexity as the standard PF. However, the EKF and UKF are still more computationally efficient than the MPF, albeit less accurate.

The MPF localization algorithm is shown in Algorithm 2 along with selected simplified equations where Q and P are covariance matrices. The equations can be found in detail in [326].

Algorithm 2 MPF

1: **Initialize particles**

2: Initialize nonlinear state variables

$$x_{0|-1}^{n,(i)} \sim p(x_0^n) \quad (\text{A4.11})$$

3: Initialize linear state variables

$$\{x_{0|-1}^{l,(i)}, P_{0|-1}^{(i)}\} = \{\bar{x}_0^l, \bar{P}_0\} \quad (\text{A4.12})$$

4: **for** $t = 1, \dots, T$ **do**5: **PF measurement update**

6: Evaluate the weights

$$w_t^{(i)} = p(y_t | X_t^{n,(i)}, Y_{t-1}) \quad (\text{A4.13})$$

7: Estimate nonlinear state variables

$$\hat{x}_{t|t}^n = \sum_{i=1}^N w_t^{(i)} x_{t|t-1}^{n,(i)} \quad (\text{A4.14})$$

8: Resample particles

$$Pr(x_{t|t}^{n,(i)} = x_{t|t-1}^{n,(j)}) = \tilde{w}_t^{(j)} \quad (\text{A4.15})$$

9: **KF measurement update**

10: Estimate linear state variables

$$x_{t|t}^{l,(i)} = x_{t|t-1}^{l,(i)} + K_t (y_t - h_t - C_t \hat{x}_{t|t-1}^l) \quad (\text{A4.16})$$

$$\hat{x}_{t|t}^l = \sum_{i=1}^N w_t^{(i)} x_{t|t}^{l,(i)} \quad (\text{A4.17})$$

11: **PF prediction**

12: Propagate nonlinear state variables

$$x_{t+1|t}^{n,(i)} \sim p(x_{t+1|t}^n | X_t^{n,(i)}, Y_t) \quad (\text{A4.18})$$

13: **KF prediction**

14: Propagate linear state variables

$$\begin{aligned} \hat{x}_{t+1|t}^l &= \bar{A}_t^l \hat{x}_{t|t}^l + G_t^l (Q_t^{ln})^T (G_t^n Q_t^n)^{-1} z_t \\ &\quad + f_t^l + L_t (z_t - A_t^n \hat{x}_{t|t}^l) \end{aligned} \quad (\text{A4.19})$$

15: **end for**



Figure A4.2: Visualization of raw bathymetry data in tagged interchange file format (TIFF) of Bde Maka Ska. Source: Minnesota Department of Natural Resources. Darker pixels represent locations with deeper depths.

Simulation Methodology

We developed simulations with real-world bathymetry data to evaluate the proposed localization algorithms. To this effect, we exploit the capabilities of the Robot Operating System (ROS) [333] in creating realistic simulations using the Gazebo tool, with a goal to reduce overhead when transitioning to AUV field deployments. Our approach enables the algorithms to be used in our simulated evaluation, and then deployed in the field with minimal changes.

Bathymetry Data Collection

The following lakes located in Minnesota, USA were chosen as test locations: Bde Maka Ska¹, Lake Nokomis, Lake Hiawatha, Lake Harriet, Lake Turtle, Lake Howard, Lake Waverly, and Lake Pulaski (Table A4.2). We selected lakes that have a wide range

¹ No preceding "Lake" is used for Bde Maka Ska, which was formerly referred to as Lake Calhoun and had its name restored in 2018.

Lake	Altitude (m)		Size ($\times 10^6 m^2$)
	Max.	Avg.	
Bde Maka Ska	-27.43	-10.00	1.71
Nokomis	-9.99	-4.11	0.84
Hiawatha	-9.45	-4.00	0.22
Harriet	-26.42	-9.75	1.39
Turtle	-8.53	-3.42	1.83
Howard	-10.97	-4.89	3.00
Waverly	-21.49	-7.59	1.99
Pulaski	-26.50	-11.04	2.98

Table A4.2: Lake Bathymetry Information

of maximum altitudes, average altitudes, and sizes. Additionally, the lakes are well-studied, have an undulating floor, and are easy to access for future field studies. The raw bathymetry data was acquired from the Minnesota Department of Natural Resources (MN DNR) [334]. Figure A4.2 shows an example of bathymetry data provided in a TIFF format. The original data contains the lake altitudes (measured in feet) every 5m in a grid pattern across the lakes.

Robotic Simulation Development

As mentioned, we use the Gazebo [335] simulation tool and ROS integration for this work. Specific information on the various components of the simulation is provided below.

Bathymetry Data Processing for Simulation

Though Gazebo provides built-in ways to import raw bathymetry data (TIFF format), our preliminary testing shows that using Standard Triangle Language (STL) format provides a more accurate lake bottom profile. With Gazebo's built-in bathymetry methods, the water column depth at each point is only rounded to the nearest meter. We convert the raw data from the TIFF format to the STL format as follows:

1. Extract an array from the raw data, representing the altitude of the relevant lake

Parameter	Value
No. of particles for the PF, N_{PF}	5000
No. of particles for the MPF, N_{NPF}	500
Motion noise cov., Q (m)	$0.005^2 \begin{bmatrix} v_x^2 & 0 & 0 \\ 0 & v_y^2 & 0 \\ 0 & 0 & v_z^2 \end{bmatrix}$
Measurement noise cov., R (m)	$0.005^2 \begin{bmatrix} depth^2 & 0 \\ 0 & alt.^2 \end{bmatrix}$
Initial uncertainty cov., P (m)	$\begin{bmatrix} 1^2 & 0 & 0 \\ 0 & 1^2 & 0 \\ 0 & 0 & 1^2 \end{bmatrix}$

Table A4.3: Model parameters

at each grid location.

2. Convert the altitude from feet to meters.
3. Create a model STL file with this array data.

The STL file is created with a process that linearly interpolates solid surface material between neighboring bathymetric data points in a triangle mesh format. With this, MATLAB can also be used to interpolate and reduce the bathymetry data grid size to 1m from 5m. For any triangle already in the 5m mesh, the corresponding mathematical plane equation can be determined based on its vertices. Any of the finer 1m grid points that land within the triangle can have their depths calculated from this plane equation. The STL files are then imported into the Gazebo simulation world as solid surface models, with a solid ground plane at the top of the model to represent the surface of the lakes.

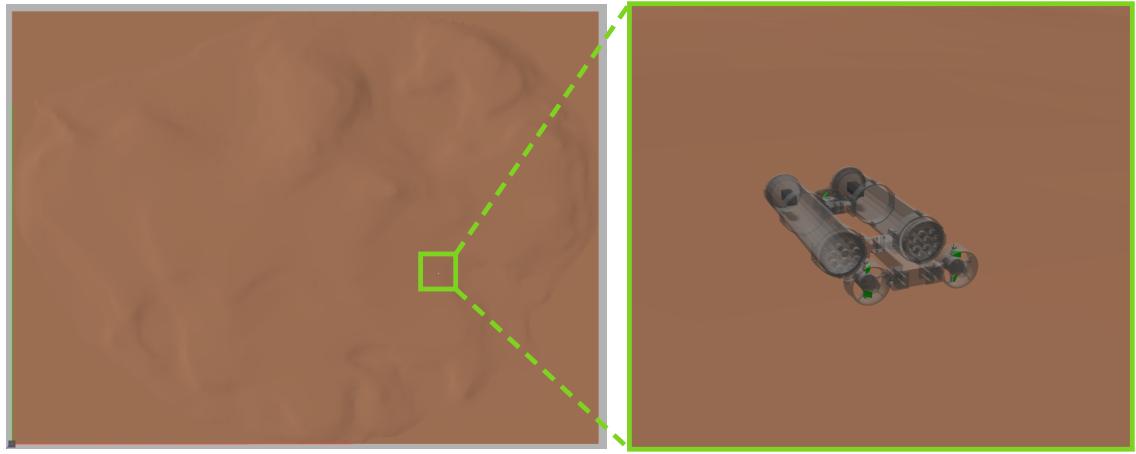


Figure A4.3: One example of simulated lakes (Bde Maka Ska) and LoCO AUV. From the top view, LoCO appears as an white dot in the left image due to the difference in actual scale between the lake and LoCO AUV. The right image shows a simulated LoCO AUV in Gazebo.

AUV and Sensors in Our Simulation

We use the LoCO AUV [29] Gazebo model from our previous work to develop the simulated experiments. We add two single-beam laser sensors to the LoCO simulation. Each laser sensor is a proxy for a depth sensor and an acoustic sensor, respectively. Both provide sensor measurements at a rate of 1 Hz. The depth sensor could be simulated directly by simulating water pressure instead of using the laser sensor. However, this would significantly complicate the simulation design, and developing hydrodynamics for the simulation is out of the scope of this work. Using the laser as a proxy is a convenient alternative since it provides sensor readings used to validate our approach.

Implementation of the Proposed Algorithms

Gazebo interfaces with ROS using individual processes called ‘nodes’ to simulate and test algorithms for various robotic functions. For our study, we develop three programs to facilitate the testing of the four localization filters. The first is the velocity control node. This program reads the simulation time from Gazebo and sends out predefined velocity commands to the robot according to the desired velocity profiles. These commands

are also received by the second program, which is the particular filter estimating the position of the robot. The filter programs are triggered to run when they receive the depth and range measurements from the robot's two sensors. The filter uses this state data, the velocity command, and the data from the previous time step to estimate the current position of the robot according to the algorithm in use. Position estimation data from the filter is received by the third program, which stores all data relevant to the current time step of the estimation filter. This information is stored in a final data file for analyzing each filter's performance during simulations and providing an overall assessment of the simulated world.

Simulation Settings

The goal of this study is to evaluate each proposed localization algorithm with real bathymetry data. It is extremely difficult to evaluate AUV localization in physical, open-water testing, as ground truth to evaluate against is nearly impossible to obtain [336]. No highly accurate localization technique is readily available to provide ground truth for our algorithm, as GPS does not work underwater and options which require equipment such as USBL and DVL devices are prohibitively expensive. As our work also requires variations in bathymetric data, evaluations in a closed-water (*e.g.*, pool) environment is infeasible. Thus, to quantify the accuracy and efficiency of the algorithms, we use our proposed simulation to evaluate the performances of each filter's position estimates against the simulated AUV's motion, since accurate ground truth can easily be collected in simulation. The linear and mixed motion models are designed to test the performance of each localization algorithm on the real-world bathymetry data from different lake environments. As real-world AUV motion is seldom linear, it is important to evaluate the mixed motion models (which contain linear and nonlinear components) as well, as nonlinear motion may cause some localization algorithms to perform poorly. The nonlinear motion model will also help account for perturbations to AUV motion arising from waves and general hydrodynamic factors.

Table A4.3 shows the model parameters for the simulation. The control inputs for each algorithm and lake are separately designed due to the lakes' different sizes and altitudes. The parameters are defined for the linear motion in Eqs. A4.3 and A4.4, and the mixed motion in Eq. A4.5. For the mixed model, x and y are nonlinear state

Lake	Method	Linear motion model									Mixed motion model								
		Iteration Time (ms)			RMSE(m)			RMSE(%) w.r.t vel			Iteration Time(ms)			RMSE(m)			RMSE(%) w.r.t vel		
		x	y	z	x	y	z	x	y	z	x	y	z	x	y	z	x	y	z
Bde.	EKF	20.5	0.151	0.309	0.005	0.863	5.831	2.683	20.1	5.040	5.990	0.016	53.509	182.882	8.683				
Bde.	UKF	0.3	0.185	0.035	0.003	1.087	0.666	1.534	0.3	1.786	3.488	0.019	33.166	110.868	12.082				
Bde.	PF	115.0	0.395	0.540	0.006	2.492	10.286	3.861	113.1	0.906	3.000	0.009	15.796	88.855	5.816				
Bde.	MPF	61.3	0.200	0.041	0.003	1.277	0.933	2.332	62.3	1.801	3.510	0.003	33.475	112.669	1.615				
Nok.	EKF	24.0	0.349	0.295	0.004	1.523	2.687	1.831	22.3	2.007	1.282	0.004	10.530	12.605	2.254				
Nok.	UKF	0.3	3.952	1.940	0.031	18.857	18.523	19.373	0.3	5.871	2.908	0.031	32.075	31.783	19.110				
Nok.	PF	113.9	0.553	0.182	0.007	2.623	1.667	4.465	115.0	3.437	2.016	0.008	19.068	22.464	5.311				
Nok.	MPF	61.1	0.216	0.055	0.003	0.931	0.503	1.582	62.3	2.806	1.378	0.002	15.462	15.188	1.470				
Hia.	EKF	3.0	0.289	0.240	0.004	2.337	4.233	1.111	2.9	0.641	0.330	0.004	5.320	5.727	1.121				
Hia.	UKF	0.3	0.121	0.037	0.004	1.020	0.692	1.001	0.3	0.708	0.345	0.004	4.965	4.912	1.125				
Hia.	PF	112.5	0.472	0.272	0.004	4.224	4.709	1.334	112.4	1.073	0.509	0.006	8.539	8.873	1.728				
Hia.	MPF	62.5	0.089	0.022	0.003	0.714	0.390	0.926	62.3	0.711	0.346	0.005	4.988	4.930	1.417				
Har.	EKF	17.4	0.236	0.152	0.003	1.405	1.5170	0.925	17.2	1.385	0.822	0.003	9.367	9.844	0.856				
Har.	UKF	0.3	0.812	0.406	0.003	6.820	6.931	1.222	0.3	1.610	0.810	0.003	11.460	11.088	1.077				
Har.	PF	112.9	0.290	0.452	0.005	1.640	5.128	1.668	112.7	1.665	1.372	0.006	10.496	15.660	1.708				
Har.	MPF	61.3	0.143	0.035	0.003	0.749	0.401	0.891	62.2	2.103	1.058	0.003	11.550	11.192	1.034				
Tur.	EKF	24.7	0.355	0.211	0.003	3.090	1.993	0.831	22.4	0.479	0.347	0.004	3.366	12.087	0.930				
Tur.	UKF	0.3	0.128	0.075	0.004	1.126	0.711	1.221	0.3	0.419	0.090	0.004	2.983	3.006	0.966				
Tur.	PF	112.5	0.516	0.251	0.006	4.815	2.399	1.893	115.6	0.948	0.723	0.006	7.521	28.647	1.816				
Tur.	MPF	62.0	0.091	0.041	0.003	0.775	0.366	0.763	62.2	0.451	0.098	0.003	3.210	3.295	0.864				
How.	EKF	94.4	0.423	0.315	0.007	4.010	2.447	1.715	84.0	1.622	0.971	0.006	12.304	8.386	1.747				
How.	UKF	0.3	0.115	0.074	0.005	0.991	0.640	1.351	0.3	1.771	1.807	0.006	13.722	14.078	1.789				
How.	PF	113.2	0.745	0.747	0.008	6.478	7.003	2.330	115.5	1.730	1.820	0.007	13.379	14.387	1.971				
How.	MPF	61.4	0.127	0.076	0.006	1.135	0.719	1.809	61.3	1.831	1.867	0.004	14.265	14.617	1.180				
Wav.	EKF	37.1	0.154	0.100	0.004	2.877	1.609	0.985	40.5	0.649	0.572	0.004	6.526	9.100	1.034				
Wav.	UKF	0.3	0.050	0.025	0.004	0.853	0.456	1.246	0.3	1.040	0.710	0.004	10.464	10.885	1.245				
Wav.	PF	115.5	0.309	1.113	0.006	5.666	19.694	1.770	116.3	0.995	0.564	0.006	11.545	9.238	1.805				
Wav.	MPF	61.7	0.059	0.033	0.006	1.029	0.612	1.727	61.5	1.057	0.721	0.005	10.670	11.084	1.515				
Pul.	EKF	69.8	0.118	0.103	0.004	1.990	1.775	0.906	62.1	2.120	2.346	0.005	17.248	16.117	1.546				
Pul.	UKF	0.3	0.055	0.028	0.003	0.936	0.514	0.792	0.3	3.781	3.708	0.009	25.876	25.425	2.775				
Pul.	PF	113.3	0.563	0.865	0.004	10.618	16.022	1.087	113.9	2.951	2.978	0.006	20.136	20.368	1.679				
Pul.	MPF	61.8	0.076	0.052	0.006	1.411	1.003	1.930	61.3	3.764	3.691	0.002	25.756	25.302	0.665				

The intensity of each color is proportional to the error. Red, green and blue colors are used for the x, y, and z-axis, respectively. Grey color represents the percentage errors across all the axis.

Table A4.4: Localization performance evaluation for an AUV.

variables, and z is a linear state variable. We measure the performance of our algorithms on an AMD Ryzen 7 5800X 8-Core processor with 3.8GHz clock speed running Ubuntu 20.04.2 LTS with 32GB of DDR4 memory with ROS Noetic and Gazebo 11.3. We select five paths per motion with the four Bayesian filters for total of eight lakes, accumulating to a total 80 unique simulated trials.

Results of Simulation Experiments

Results and Discussion

The results of our trials are summarized in Table A4.4. We use the root-mean-square error (RMSE) in Eq. A4.22 as a metric to evaluate the performance for each axis of

motion where T is the number of steps, p_g is the ground truth, and p_e is the estimated position.

$$RMSE = \sqrt{\frac{\sum_{t=1}^T (p_g - p_e)^2}{T}} \quad (\text{A4.22})$$

The widths and lengths of the lakes are significantly larger than the depths, meaning that the velocity of x and y axes is generally two orders of magnitude larger than the velocity of the z axis in our simulation. This yields notably smaller errors in the z axis than in the other axes (Table A4.4). We also evaluate positional errors ($|p_g - p_e|$) relative to velocity for each axis.

In Fig. A4.4, we show one particular trial in Bde Maka Ska as an example. For both the linear and mixed cases, the MPF generally performed well in this trial. We observe degraded performance from the PF and EKF evaluations in the linear case; and from the EKF and UKF evaluations in the mixed case compared to the other filters. The EKF had the worst performance overall in Bde Maka Ska. The PF outperforms the other filters in the mixed motion case, but it was the least accurate of the four filters in the linear case. A likely cause of the degraded performance of the PF is that the randomly selected particles can cause the estimation to diverge from the ground truth when a water body's bathymetry data does not have sufficient variation.

Linear Motion Case

Looking at the overall results from the linear motion case, the EKF does not always perform best, but it provides reliable and relatively accurate results with lower computational complexity. While the UKF estimates the position of LoCO with the smallest RMSE for most lakes, it is not reliable, since its performance varies widely between lakes. For instance, the average UKF RMSE of the x, y , and z axes is ten times bigger than the average MPF RMSE in Lake Nokomis. The average UKF error is three times greater than the average EKF error in Lake Harriet. The PF's performance is the worst overall, particularly in Lake Howard and Waverly Lake since the paths chosen for the two lakes have fewer variations in altitude compared to the paths in the other lakes. Although the MPF does not present the smallest RMSE in every lake, it does not show significantly degraded accuracy in any of the lakes. The MPF generally gives accurate and reliable results, and it is computationally cheaper than the PF. Overall, the MPF

is the most reliable and accurate filter based on the results of these experiments. It is worth mentioning that the EKF is a good option if an AUV does not have sufficient computational power for the MPF and if high accuracy is not required. The PF can be used for localization if the bathymetry data has enough variation in altitude and has an asymmetrical structure, provided that the AUV has enough computational power.

Mixed Motion Case

Unlike in the linear motion cases, the EKF is much more unreliable for mixed motion cases, achieving the least accurate estimation out of the four filters in some lakes while performing best in other lakes. For example, the EKF RMSE is larger than twofold of the other filters' RMSE in Bde Maka Ska. The UKF performs poorly in some lakes due to the high nonlinearity of the motion. However, it needs the least amount of computational resources among all the filters. The PF displays the same issue in mixed motion cases that it has in the linear motion cases: the estimations diverge when there are not enough variations in bathymetry data (*i.e.*, Lake Hiawatha, Lake Nokomis, and Turtle Lake). Similarly to the linear cases, in the mixed motion cases the MPF gives reliable and accurate results overall. Therefore, our recommendation for using the MPF remains the same for mixed motion cases, but we do not recommend using the EKF in these cases.

Discussion

The MPF consistently produces reliable and accurate results for both the linear and mixed motion cases while maintaining relatively low computational costs. If an AUV needs a well-rounded algorithm for the localization problem, the MPF is the best filter among those we have evaluated. However, if the bathymetry data of a lake has sufficient variation and the task requires high accuracy, then the PF is a better option, albeit at a higher computational cost. Determining what constitutes enough variation is a complicated question affected by several factors: the size of littoral zone, the paths of the AUV, and the average and maximum altitude of the lake. We perform our analysis on lakes that have a wide range of such factors. Defining the magnitude of variation quantitatively is a problem that should be explored in future work. While

PF and MPF have high accuracy in some cases, they require memory to store the state of their estimated particles, making them computationally expensive. The MPF is significantly less computationally demanding than the PF, but still requires more memory and processing power than the EKF and UKF. Therefore, for an AUV with low computational power, the EKF could be the best filter for localization, but lower accuracy should be expected, particularly if the AUV is not constrained to linear motion.

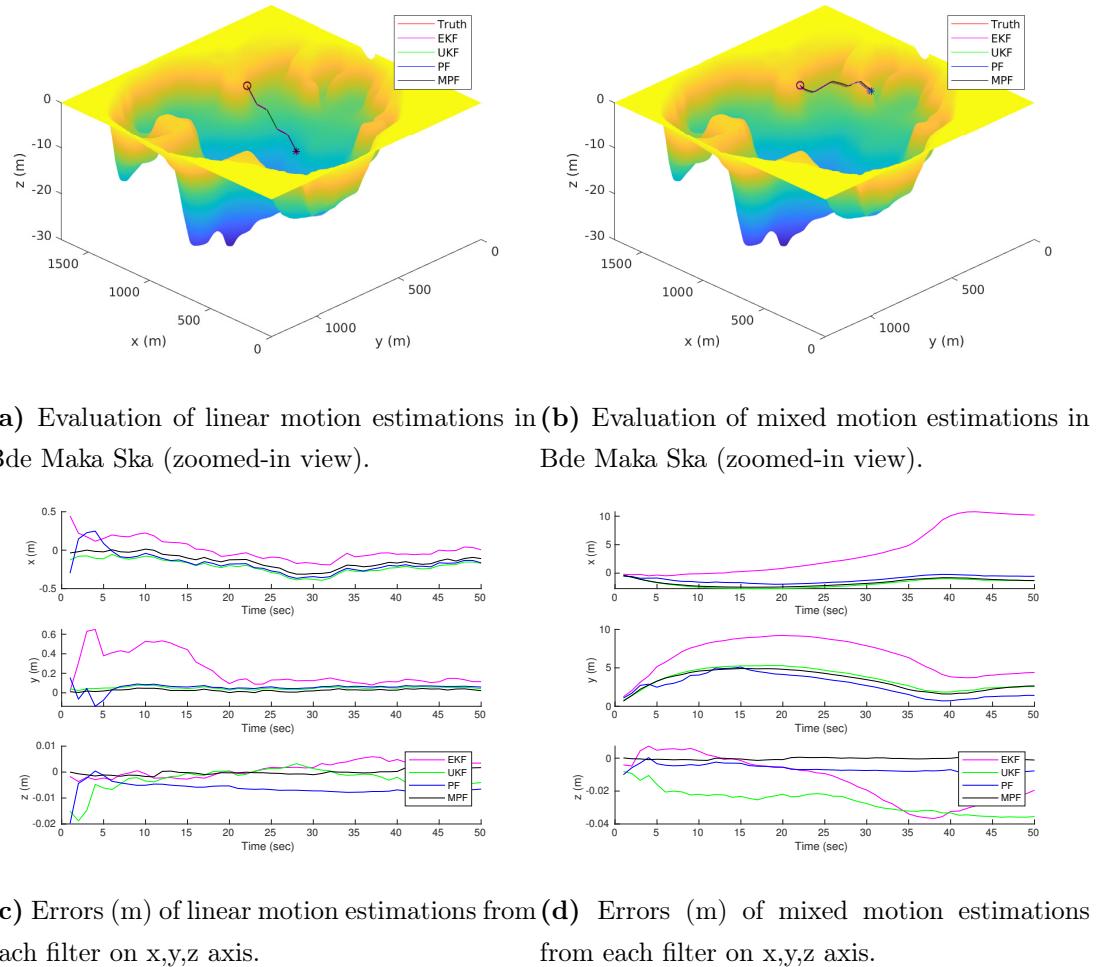


Figure A4.4: Localization performance of the four algorithms for an AUV with an altimeter and depth sensor within Bde Maka Ska using bathymetry data (Start: \circ End: $*$).

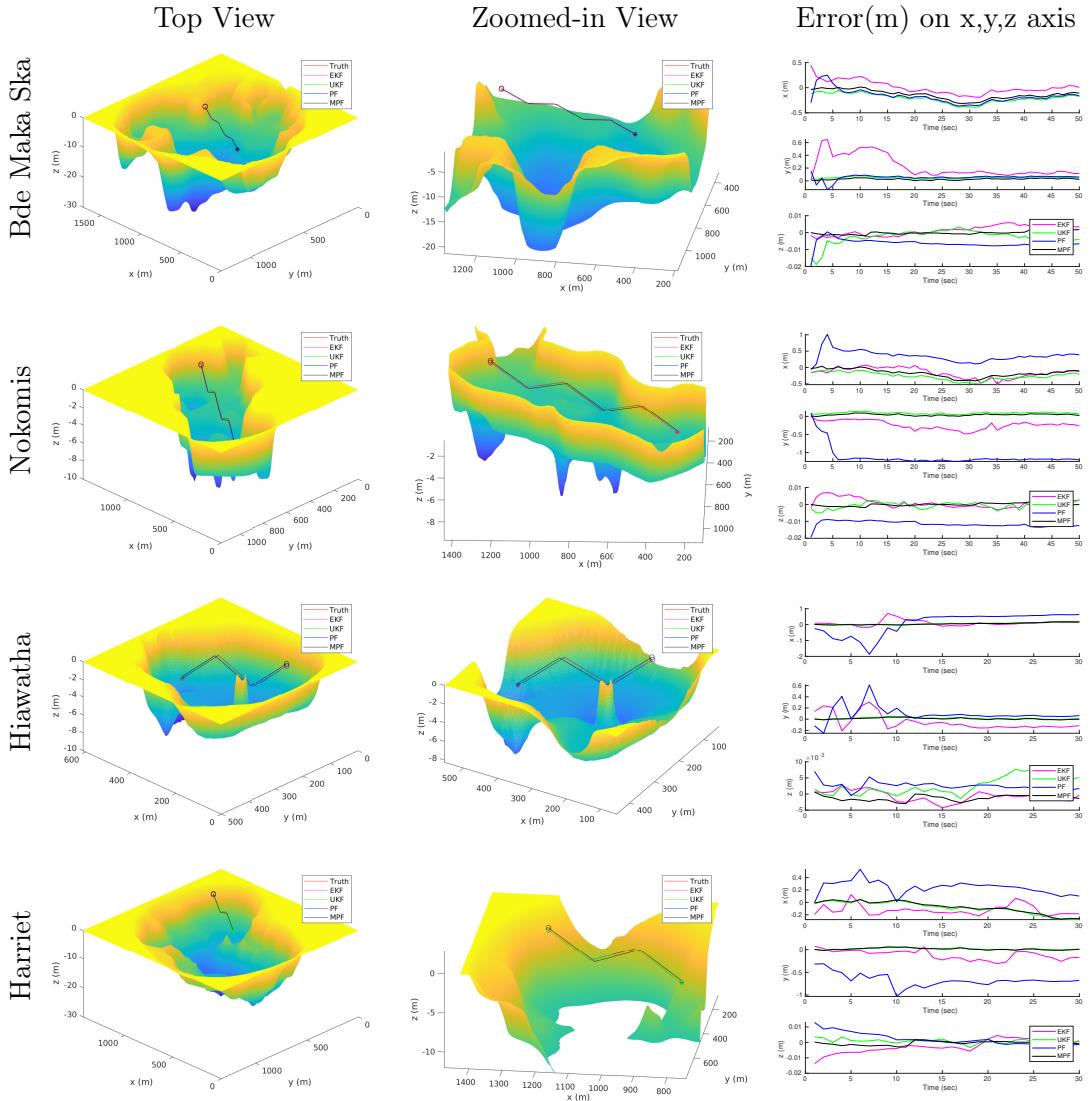


Figure A4.5: Localization performance of the four algorithms for an AUV with linear motions (Best readability at 2x zoom).

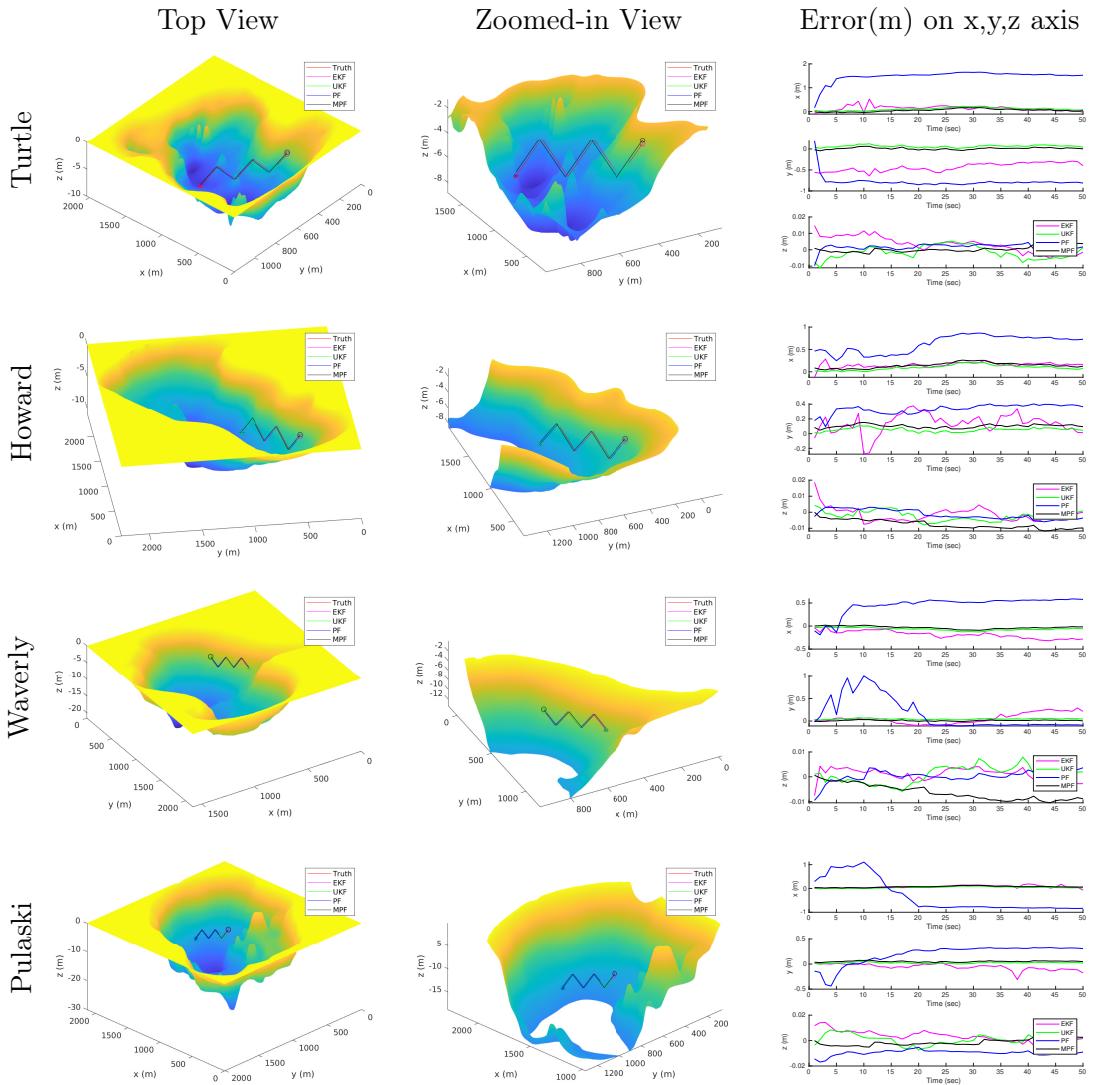


Figure A4.6: Localization performance of the four algorithms for an AUV with linear motions (Best readability at 2x zoom).

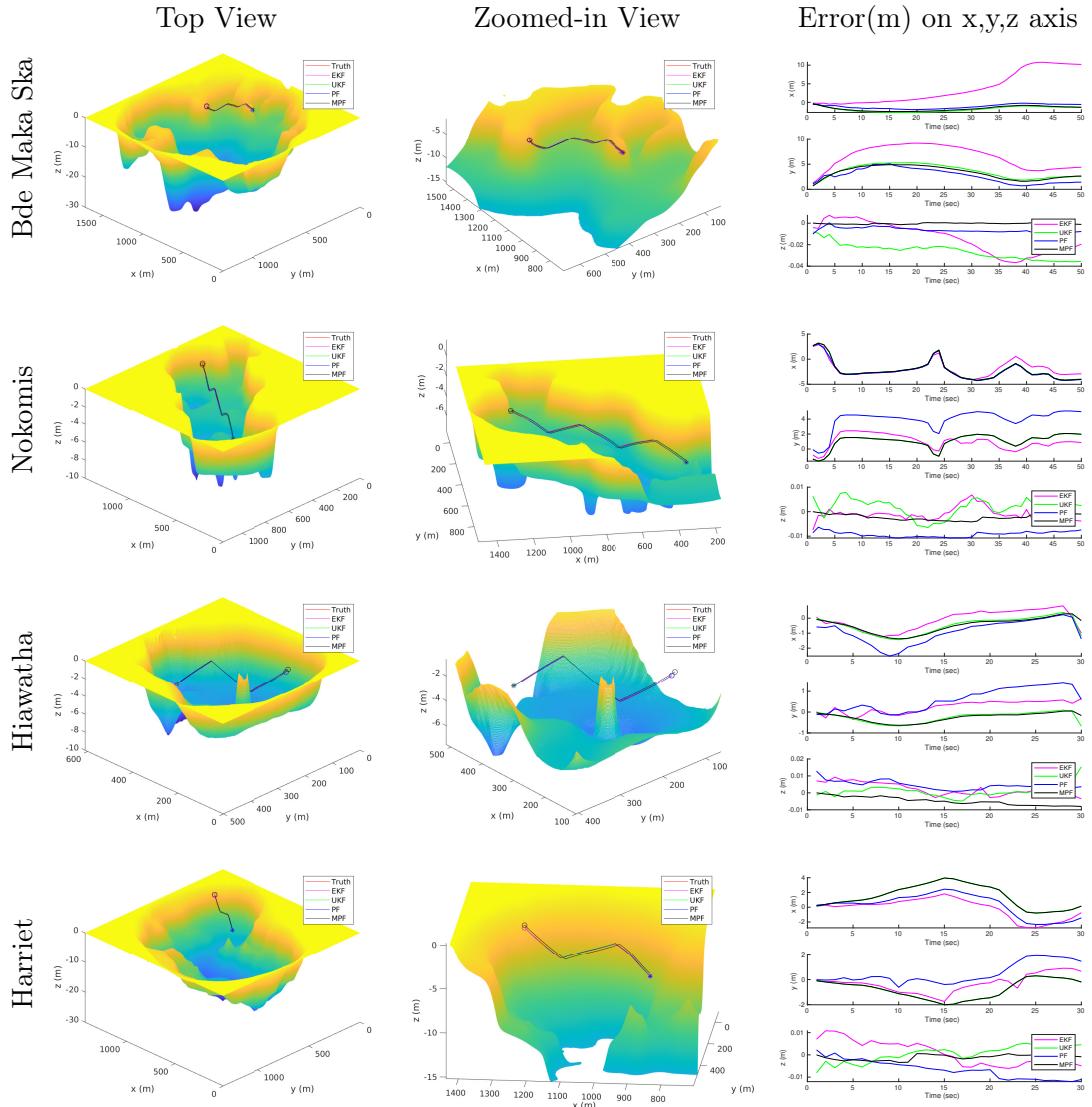


Figure A4.7: Localization performance of the four algorithms for an AUV with mixed motions (Best readability at 2x zoom).

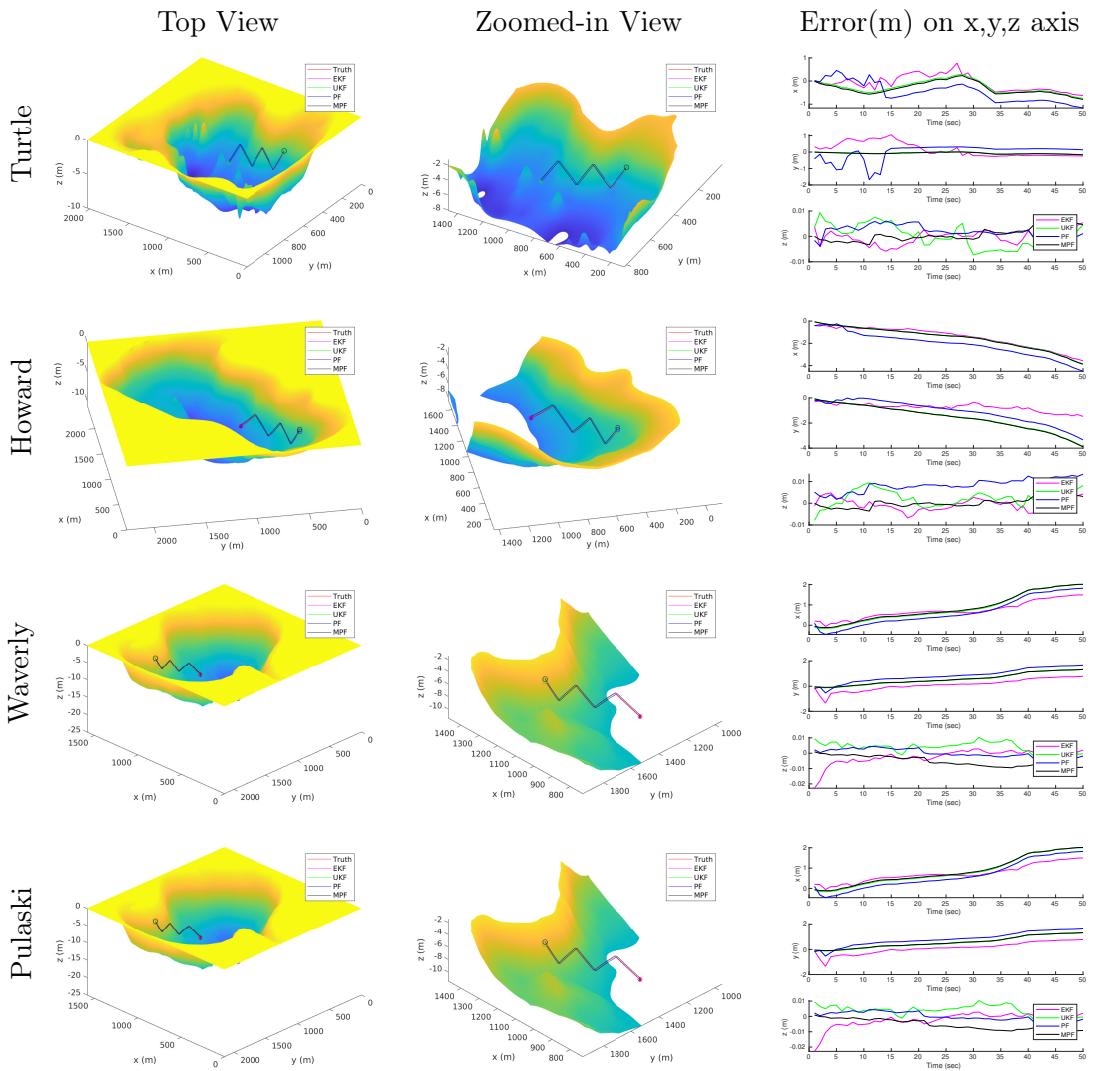


Figure A4.8: Localization performance of the four algorithms for an AUV with mixed motions (Best readability at 2x zoom).

Appendix V: Available Datasets

A great deal of the work presented in this thesis required the development of new datasets or produced interesting data. If you are interested in accessing any of the data presented in this paper, the following subsections should cover the majority of the involved datasets. If you do not see the data you are interested in, please contact Michael directly, or the Interactive Robotics and Vision Lab.

Human Studies Data

The majority of human studies data presented in this thesis is not available for public access, due to privacy concerns and data security procedures established in the IRB protocols for each study. If you are interested in accessing the data from any of the studies presented in this thesis, please contact Michael Fulton (updated contact information can likely be found at <https://michaelscottfulton.com>, but be aware that you will likely not be allowed direct access to the data).

VDD- \bar{C}

VDD- \bar{C} is available for public use at <https://conservancy.umn.edu/handle/11299/219383>. Please cite the dataset as well as the relevant IROS paper if you use it.

IROS Paper Citation: K. d. Langis, M. Fulton and J. Sattar, "Towards Robust Visual Diver Detection Onboard Autonomous Underwater Robots: Assessing the Effects of Models and Data1," 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 5372-5378, doi: 10.1109/IROS51168.2021.9636099.

Dataset Citation: de Langis, Karin; Fulton, Michael; Sattar, Junaed. (2021). Video Diver Dataset (VDD-C) 100,000 annotated images of divers underwater. Retrieved from the Data Repository for the University of Minnesota, <https://doi.org/10.13020/6qrp-wy09>.

OceanPose Dataset

The OceanPose dataset will be made available for public use at some point in the future. Currently, it remains closed due to ongoing work and the need to appropriately manage privacy of the people shown. Please see the “Resources” menu on <https://irvlab.cs.umn.edu/> to see if it has been released.

PGR Dataset

The Protean Generation and Recognition dataset will be made available for public use at some point in the future. Currently, it remains closed due to ongoing work and the need to appropriately manage privacy of the people shown. Please see the “Resources” menu on <https://irvlab.cs.umn.edu/> to see if it has been released.

Trash-ICRA19 Dataset

The Trash-ICRA19 dataset is available at <https://conservancy.umn.edu/handle/11299/214366>. Please cite the dataset as well as the relevant ICRA paper if you use it.

ICRA Paper Citation: M. Fulton, J. Hong, M. J. Islam and J. Sattar, "Robotic Detection of Marine Litter Using Deep Visual Detection Models," 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 5752-5758, doi: 10.1109/ICRA.2019.8793975.

Dataset Citation: Fulton, Michael S; Hong, Jungseok; Sattar, Junaed. (2020). Trash-ICRA19: A Bounding Box Labeled Dataset of Underwater Trash. Retrieved from the Data Repository for the University of Minnesota, <https://doi.org/10.13020/x0qn-y082>.

TrashCan Dataset

The TrashCan dataset is available at <https://conservancy.umn.edu/handle/11299/214865>. Please cite the dataset if you use it.

Dataset Citation: Hong, Jungseok; Fulton, Michael S; Sattar, Junaed. (2020). TrashCan 1.0 An Instance-Segmentation Labeled Dataset of Trash Observations. Retrieved from the Data Repository for the University of Minnesota, <https://doi.org/10.13020/g1gx-y834>.

A Closing Note:

If you have read this far, thank you very much. If you simply skipped to the end, go back and check out some of the stuff you missed, it's cool research. This thesis is one of the biggest things I have ever done in my life. The effort that it took was immense, and while I would like to believe that it is perfect, I am too honest about my own abilities not to note: there may be errors within this work. If you discover one, do contact me and let me know, I'd be happy to update the thesis itself or a list of errata. Thank you for taking an interest in my work, and I wish you all the best in your endeavors. Make sure you take time for yourself.