3D Visual Tracking and Inter-Robot Communication Through Full-Body Gestures

Karim Koreitem

Master of Computer Science (Thesis)

School of Computer Science

McGill University

Montreal, Quebec

2019-02-26

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

DEDICATION

This thesis is dedicated to my parents, Lina and Hassan, to my sisters Zeina and Ala, and to Emilie, my partner and closest friend.

ACKNOWLEDGEMENTS

My journey at McGill and in particular, the Mobile Robotics Laboratory (MRL), has been an incredibly enlightening and memorable experience and I would like to thank everyone who has been a part of it. Foremost, I would like to express my sincere gratitude to my supervisor Professor Gregory Dudek for the opportunity to be a part of MRL as an undergraduate student at first and a graduate student afterwards, for his support, his guidance and for inspiring me through his enthusiasm for fresh ideas and unrivaled creativity. I would like to thank Professor Dave Meger for always ensuring MRL is a welcoming and vibrant environment, his leadership, and his everlasting excitement for all things robotics. I would like to express special thanks to Jimmy Li, for never-ending advice, always taking the time to explain things, critique my ideas and support me throughout my projects, to Ian Karp for help across a seemingly uncountable number of field trials and experiments and journeying together into the Unreal and to Travis, for excellent feedback, motivating me and for his wizardry with integrating hardware into Aqua, namely the Nvidia Jetson. Thank you to all members of MRL for creating a wonderful learning community, helping me through pool trials and in particular, to Florian for being an inspiring embodiment of rigor and hard work, to Juan for his infectious excitement, to Nikhil and Andrew for obsessing over elegance and beauty in code, to Wei-Di and Edward for great ideas, criticism and many fun nights of gaming, to Johanna for leading our volleyball team to a championship, to Arnold for many coffee chats and to Scott, Auguste and Lucas for many great discussions. I also wish to express my appreciation to my UofT

family Aldrich, Matthew, Muntaha, Niki, Heindrik, Karim, Ekansh, Phil, Aaron, and Ashis, and to my childhood friends, Sam, Bassel, Jana, Aya, Amine, Armando and Amer for being there for me. Finally, I would like to thank my family: my partner, Emilie, my parents, Lina and Hassan, my sisters Zeina and Ala, my brother-in-law John, and my cousin Moe for their unconditional support and encouragement. If you should have been included here but are not, I apologize, and will shamefully hear your outcry.

ABSTRACT

In this thesis, we consider inter-robot communication in robot convoying settings. In particular, we investigate passive communication for radio-denied environments by using whole-body gestures performed by an underwater robot to provide cues regarding future actions. We first focus on visually detecting and tracking the 3D pose of autonomous underwater vehicles to enable robust multi-robot convoying. We follow the approach of tracking-by-detection, which combines the robust, drift-free nature of object detection with the temporal consistency of tracking algorithms. Our approach relies on a multi-output convolutional network that jointly predicts the target robot's presence in the image, its 2D bounding box, and its 3D orientation. This, combined with camera intrinsic parameters and prior knowledge of the robot's scale, allows us to recover the full 6-degree-of-freedom pose of the target robot. We then leverage the tracked pose to develop a visual communication protocol whereby information is transmitted through codewords: a series of actions executed by the swimming robot. These sequences are chosen to optimize robustness and transmission efficiency given the observability, natural activity of the robot and the frequency of different messages. The observer robot then uses an adaptation of classical decoding methods to infer the transmitted message. To train our network, we rely exclusively on synthetic images and we test our system on underwater datasets in both pool and ocean settings. Our evaluation demonstrates successful generalization of both the learned tracking model and the visual communication protocol to real underwater footage of the target robot.

ABRÉGÉ

Dans cette thèse, nous considérons la communication inter-robot dans les environnements de convoi de robots. En particulier, nous étudions la communication passive pour les environnements sans radio en utilisant des gestes du corps entier exécutés par un robot sous-marin afin de fournir des indices sur les actions à venir. Nous nous concentrons d'abord sur la détection visuelle et le suivi de la pose 3D de véhicules sous-marins autonomes afin de permettre un convoyage multi-robot robuste. Nous suivons l'approche du suivi par détection, qui conjugue la nature robuste et sans dérive de la détection d'objet et la cohérence temporelle des algorithmes de suivi. Notre approche repose sur un réseau convolutionnel à plusieurs sorties qui prédit conjointement la présence du robot dans l'image, son cadre de délimitation 2D et son orientation 3D. Ceci, combiné aux paramètres intrinsèques de la caméra et à la connaissance préalable de la taille du robot, nous permet de récupérer la pose complète à 6 degrés de liberté du robot. Nous exploitons ensuite la pose suivie pour développer un protocole de communication visuelle dans lequel de l'information est transmise via des mots de code (ou *codewords*): une série d'actions exécutées par le robot nageur. Ces séquences sont choisies pour optimiser la robustesse et l'efficacité de la transmission en fonction de l'observabilité, de l'activité naturelle du robot et de la fréquence des différents messages. Le robot observateur utilise ensuite une adaptation des méthodes de décodage classiques pour déduire le message transmis. Pour entraîner notre réseau, nous nous basons exclusivement sur des images synthétiques et nous testons notre système sur des jeux de données sous-marins en piscine et en

océan. Notre évaluation démontre la généralisation réussie du modèle de suivi appris et du protocole de communication visuelle sur de vraies images sous-marines du robot.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

1.1 Introduction

We present a vision-based robot-to-robot communication system that leverages the rich geometric information recovered from 3D target tracking to unambiguously transmit messages through gesturing in multi-robot convoying systems. The communication protocol relies on a robot executing a set of gestures that are in turn decoded by another robot from their visual appearance.

Humans commonly use gestures in various situations to communicate intent or issue commands when other forms of communication are difficult. Some examples include aircraft marshalling in aircraft ground handling as shown in Fig. 1–1, hand gesturing by scuba divers or cyclists signalling their trajectory to drivers and fellow cyclists. This behavior is not exclusively human and has been extensively studied in the animal kingdom. One notable example is the "waggle dance" used by bees to communicate locations of food sources (von Frisch, 1993).

With the increased deployment of robots in constrained environments and rise of robots with a variety of non-compatible custom hardware, gesturing can be seen as a suitable universal communication method adaptable to many robotic settings.

Figure 1–1: Airport marshaller requesting the aircraft to face him.

The gestures we use, represented by a sequence of pose configurations, are treated as *codewords* in a code which accounts for different transmission costs associated with each pose configuration. The code must be prefix-free (Huffman, 1952; Dumitrescu, 2006) to ensure no codeword can be a prefix of another codeword, avoiding ambiguity when decoding any message linearly. We formulate a generic communication protocol and design it to be applicable to any robotic setting where agents are capable of executing different discernible pose configurations. These pose configurations can be executed by a subset of the robot such as with an arm or the robot's entire body. In this thesis, we limit our examples to the latter and we focus our application and experiments in an underwater setting, where robot communication is naturally more constrained and other communication methods require additional

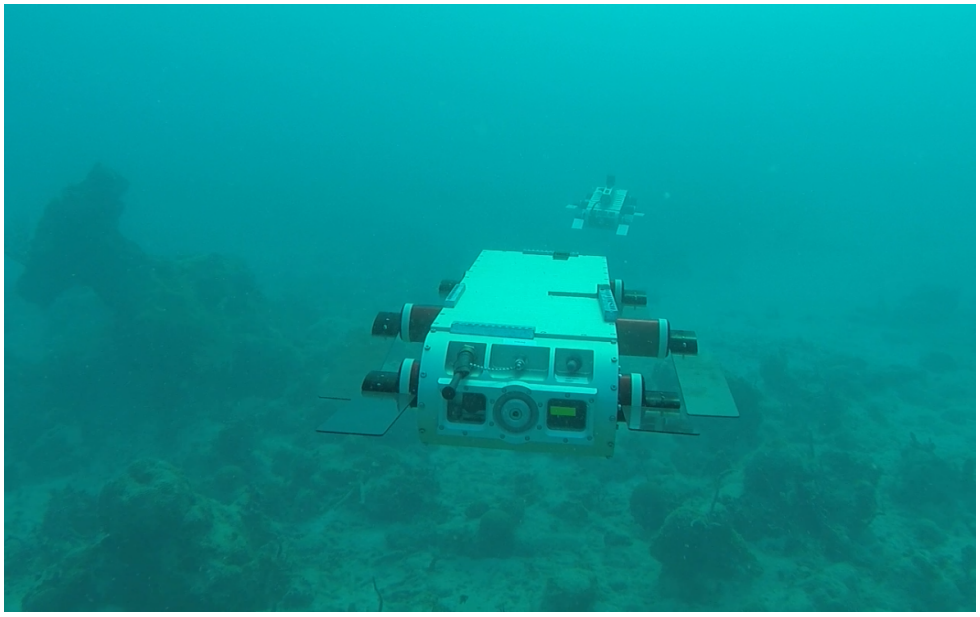


Figure 1–2: A typical convoy of two Aqua robots in position for visual communication.

hardware. In Fig. 1–2, we show two Aqua robots in a standard convoy setup where visual communication can be used by the leading robot.

Our solution employs *tracking-by-detection*: the relative position and orientation of a robot is detected at every frame, and integrated temporally via filtering. The detector we employ is a convolutional neural network (CNN) trained on synthetic images rendered using a CAD model of the robot. The trained network jointly uses regression to compute an estimate of the robot's orientation as a quaternion, as well as a tightly fitting bounding box around the robot in the image plane. Given the camera's intrinsic parameters and the absolute scale of the robot, we can use the detected orientation and bounding box to directly estimate the 3D translation of the

target robot. In this way we acquire the full 6-degree-of-freedom (6-DoF) pose of the target.

Training a convolutional network typically requires a large training dataset, but manually labelling 3D ground truth on real images is labor-intensive, and requires the labeller to learn to use sophisticated software tools. Thus, we opt to generate our entire training data synthetically using a custom-designed underwater environment designed in Unreal game development engine (EpicGames, 2018). We follow the domain randomization approach of Tobin et al. (2017), and generate a large variation of non-photo-realistic training images featuring the robot rendered with a variety of textures and on different background patterns. We then test our trained model on a test set of both synthetic and real images containing hand-labeled 3D poses. Only having to annotate test set images substantially reduces our data management cost. Overall, our detection method and training regime are widely applicable to many multi-robot convoying systems, since a CAD model of the robot is usually readily available and the only on-board sensor needed is a calibrated RGB camera.

We validate our system on the Aqua family of amphibious hexapod robots (Sattar et al., 2008) shown in Fig. 1–4. Aquas are highly manoeuvrable and rely on six actuated flippers to traverse underwater environments. Navigation and gesturing are handled by the robot's 3D autopilot (Meger et al., 2014) which relies on an on-board inertial measurement unit (IMU) and pressure sensor to perform closed-loop control over requested depth, attitude and thrust. Aqua's controllers and sensor suite are operated within the Robot Operating System (ROS) (Quigley et al., 2009) framework. Our visual tracker and decoder rely on images captured by the front-facing

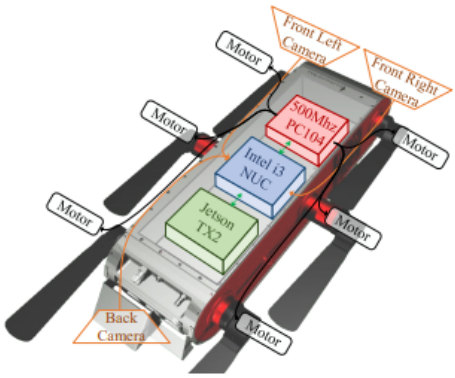
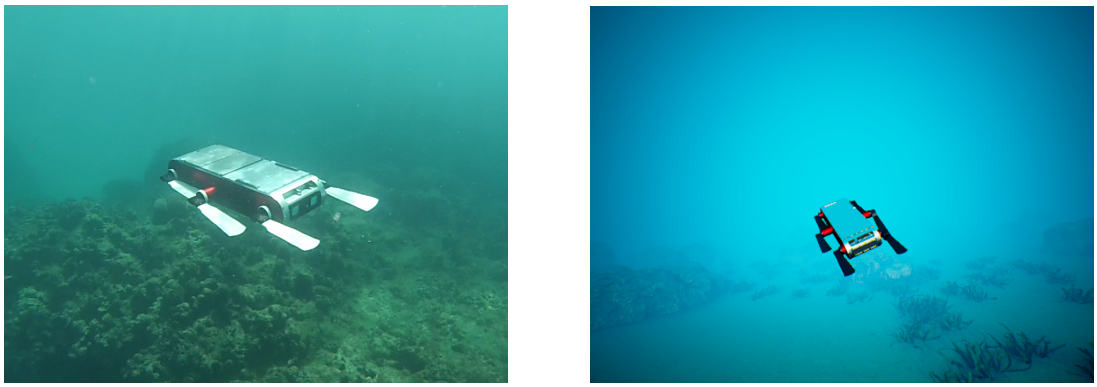


Figure 1–3: Overview of the Aqua robot and functional block diagram showing its main computers, cameras, and motors. Diagram was first published in Manderson and Dudek (2018).

RGB camera and runs at frame rate on the robot's laptop-grade dual-core Intel i3 CPU and Nvidia GPU Jetson TX2 (Manderson and Dudek, 2018). An overview of the Aqua robot and a functional block diagram is shown in Fig. 1–3.

We conduct experiments both in simulation using the synthetic marine environment designed in our custom Aqua simulator (Manderson et al., 2018) built on Unreal and on real footage of the robot collected underwater in both marine and pool settings. In our simulated runs, we analyze the robustness of our tracker by varying the complexity of the trajectory executed by the target robot. For evaluating our communication system, we analyze the visual decoder's performance on both static synthetic and real footage of the robot executing messages and on swimming trajectories which include messaging. Experiments on real images show that our learned model successfully generalizes and can be utilized to reliably implement multi-robot convoying underwater with inter-robot communication support.



(a) An Aqua robot pictured underwater in Barbados

(b) An Aqua robot simulated in Unreal Engine

Figure 1–4: The Aqua robot: a highly maneuverable amphibious hexapod.

1.2 Contributions

The contributions of this thesis are summarized in the following list:

1. The design, associated analysis and evaluation of a visual tracking system that relies on a pose estimation network and enables robot convoying.
2. The design, associated analysis and evaluation of a gesturing-based visual system for inter-robot communication which relies on an optimal prefix-free encoding of poses and a visual decoder that is robust to ambiguity through a flexible framework for assigning variable transmission costs.
3. The design, associated analysis and evaluation of a joint 3D object detection and pose estimation multi-output CNN network trained on synthetic data which generalizes to real images with no manual modifications.

1.3 Thesis Outline

This thesis is organized as follows. In Chapter 2, we present previous work in visual tracking, pose estimation, synthetic to real learning as well as robot and visual communication. In Chapter 3, we introduce our 3D visual tracking pipeline. This pipeline includes our custom joint 3D object detection and pose estimation network, our synthetic training datasets generated in Unreal and a summary of its performance. In Chapter 4, we present our visual communication algorithm with an overview of our prefix-free encoding of poses, visual decoder algorithm, the collected synthetic and real datasets collected to evaluate the system and our experimental results. Finally, Chapter 5 summarizes the system's performance and limitations and presents avenues for future work. This thesis and its tables and figures are taken from work published by the author (Koreitem et al., 2018) and unpublished work currently in review (Koreitem et al., 2019). The author is a full contributing and lead author to both of these contributions.

CHAPTER 2
Background and Related Work

This thesis builds on related work from visual tracking, 3D object detection, synthetic training, robot and visual communication and work in information theory on optimal prefix-free encoding. A brief background summary of each of those areas are discussed in the following sections.

2.1 Visual Tracking and Convoying

There is extensive literature on visual tracking. Motion-based methods have been shown to be robust to slow moving objects (Jepson et al., 2003) and many model-free tracking algorithms are designed to cope with unforeseen object instances. For example, in Yu et al. (2008), the algorithm is initialized with a bounding box around an arbitrary object to be tracked, and the algorithm is expected to adapt to the target's appearance changes throughout tracking. Our approach is closer to model-based tracking, in which a model of the target is built ahead of time. Like Manz et al. (2011), we use a CAD model of the target to train a discriminator but instead of using hand-designed edge and vertex features, we train a convolutional neural network (CNN) to recognize the target. Early non-CNN based methods for

learning features for recognition used principal components analysis (Jugessur and Dudek, 2000).

Our method falls in the category of tracking-by-detection. The target is first detected independently at every frame, and the detections are then integrated via tracking. This approach has been shown to effectively combine the temporal consistency of tracking with the robust, drift-free nature of object detection (Andriluka et al., 2008).

In the domain of visual convoying, fiducial markers have been added to the target to make detection and tracking easier (Schneiderman et al., 1995a). However, this approach is often susceptible to the marker going out of view, so we opt to directly model the natural appearance of the target as in Giesbrecht et al. (2009). Explicit signalling behaviors executed by the target has also been exploited to improve the efficiency and robustness of multi-robot convoying (Dudek et al., 1995).

In Shkurti et al. (2017), Shkurti et al. demonstrate an underwater convoying system based on tracking-by-detection that operates on the 2D position of the robot in the image plane. In this work we track the 3D pose of the target, which allows us to potentially leverage a more geometrically detailed motion model to better predict the motion of the target, ultimately resulting in more robust tracking. This is becoming especially relevant as Manderson et al. have successfully demonstrated complex exploration behaviors on the Aqua robot in which the robot stays close to corals while avoiding collision and barren uninteresting regions (Manderson et al., 2018). Convoying under this setting requires visually tracking the robot while it

carries out complex maneuvers and understanding the heading of the robot through its orientation can be very beneficial.

Other approaches to tracking include the use of CNNs in 2D (Hong et al., 2015), sparse vectors (Bao et al., 2012) and other techniques (Kristan et al., 2015), typically based on 2D image models as well as unified 2D CNN frameworks for joint object detection and tracking (Feichtenhofer et al., 2017). As in prior work (Kendall et al., 2015; Koreitem et al., 2018), we regress the orientation as a quaternion using CNNs. Using 3D pose information offers the distinct advantage of capturing the implicit intent of the robot as it positions itself to take a certain heading. This results in a better predicted motion model and thus more robust tracking. The notion of tracking for motion prediction and estimation has been considered in several contexts and has many applications (Ren and Beard, 2004; Leonard and Durrant-Whyte, 1991; Tribou et al., 2015; Dudek and Jenkin, 2010; Schneiderman et al., 1995b; Ren and Beard, 2004).

2.2 3D Object Detection

Recently there has been substantial interest to move beyond 2D bounding box detection and to infer 3D information about objects in the image. While earlier work relies primarily on detecting hand-crafted features such as local feature descriptors e.g. SIFT (Lowe, 2004), HOG (Dalal and Triggs, 2005) in the image and matching them to features on known 3D object models (Collet et al., 2009) (Lim et al., 2013) (Fidler et al., 2012), newer methods often leverage convolutional networks, either to directly learn a mapping from pixels to pose information (Song and Xiao, 2016) (Li et al., 2017), or to use features produced by the network to facilitate subsequent pose

optimization (Izadinia et al., 2017). A variety of object representations have been investigated, including 3D object centroids (Tobin et al., 2017), 3D bounding boxes (Song and Xiao, 2016), 3D skeletons (Li et al., 2017), and CAD model instances retrieved from a database (Izadinia et al., 2017).

In the target tracking domain, we are able to assume that the physical dimensions of the target is known. We leverage this in our detection method, wherein we train a network to output only the target's orientation and 2D bounding box; we then combine this information with the known scale to directly optimize the target's translation. As in Kendall et al. (2015), we regress the orientation as a quaternion.

2.3 Training on Synthetic Data

Modern object detectors based on convolutional networks are heavily dependent on abundant training data. Compared to traditional detection tasks that typically output only 2D bounding boxes in the image plane, 3D pose detection introduces an expanded output space, which leads to even greater data requirements and much more labor-intensive ground truth annotation procedures. Highly sophisticated software tools are typically required to characterize the 3D pose of objects that appear in images, which adds to the challenge of creating sufficient training data (Xiang et al., 2016).

To sidestep the need for data labelling, there has been considerable interest in using synthetic images rendered from CAD models as training data (Su et al., 2015) (Movshovitz-Attias et al., 2016) (Sun and Saenko, 2014). Peng et al. find that a pre-trained network that is fine-tuned on synthetic data can better adapt to new tasks than directly training a network for the new task using few labeled real images (Peng

et al., 2015). We take a similar approach by bootstrapping our network with weights from the VGG network trained on ImageNet (Simonyan and Zisserman, 2014), and retraining on synthetic data.

We are also inspired by the work on domain randomization by Tobin et al. (2017), which demonstrates that when a model is trained on a large set of unrealistic images that exhibit sufficient variability, the real world can simply be considered as another variation. This approach is especially relevant for the underwater domain since factors such as light absorption and reduced visibility caused by suspended sediment are difficult to simulate. Our synthetic training data renders the target robot using a variety of textures and an assortment of background patterns in order to help our learned model generalize to real images.

2.4 Robot and Visual Communication

In marine environments, radio communication is often impractical or impossible and several authors have examined interesting alternatives (Sutantyo and Levi, 2015; Wang et al., 2017; Doniec et al., 2010). In Sattar et al. (2007), the authors employ Fourier tags in order to explicit allow communication over the visual channel. In contrast, whole-body motions and activities are widely employed in the animal kingdom and have also inspired several robotics efforts (Dudek et al., 2007, 1995; Nishimura and Schwager, 2018; Jones and Andersson, 2013; Landgraf et al., 2011; Srinivasan, 2010; Baillieul and Özcimder, 2012; Raghunathan and Baillieul, 2008; Corke et al., 2007)

Several authors have considered the utility and nature of gesture-based communications in robotics (Feil-Seifer and Mataric, 2005; Kortenkamp et al., 1996). There

has also been some prior work on allowing robots to communicate via body movement (Nakata et al., 1998) or mutual observation (Rekleitis et al., 2003; Roumeliotis and Rekleitis, 2004). In (Dudek et al., 1995), gestures performed by a target robot are used to communicate heading in a robot convoying setting. While the robot behavior is important, a key aspect of this scheme is specifically engineering a set of body markings (helical drawings) for the robot that wishes to communicate. In addition, the vocabulary that is encoded is very simple and does not include any provision for error correction.

While most of these methods rely on additional hardware, our system benefits from RGB cameras, typically available on most platforms. In an underwater setting, our method also benefits from being diver-friendly as gesturing is much more interpretable by divers already occupied with their current dive plan.

2.5 Optimal Prefix-free Codes

Finding a minimal cost prefix-free code in which the encoding alphabet features r symbols of unequal letter costs is a well-studied problem (Karp, 1961; Golin and Rote, 1998; Bradford et al., 2002). Such an encoding represents a generalization of the classical Huffman coding problem (Huffman, 1952) of constructing a binary ($r = 2$) prefix code which minimizes the expected transmission cost. The generalization relaxes the binary requirement for the encoding alphabet and introduces variable costs for each encoding character (codebit). This is desirable when it is preferable to minimize the average number of codebits and when codebits of the encoding alphabet have a varied transmission cost such as in the Morse code $\{\cdot, -\}$, as seen in Fig. 2–1. In our setting, the variable cost is also an excellent way of penalizing

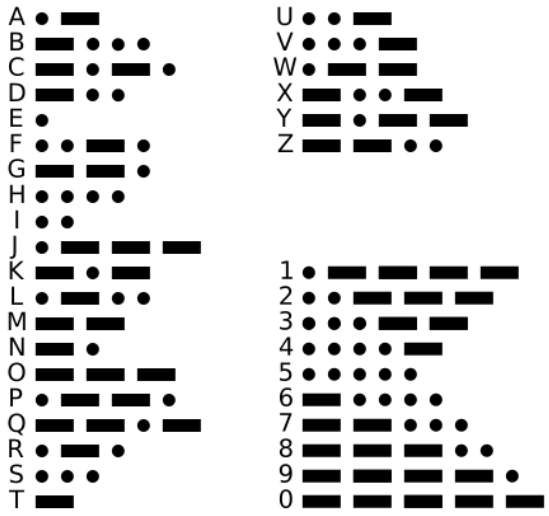


Figure 2–1: Morse code is an encoding scheme used in telecommunication which uses sequences of two signals of different transmission duration.

pose configurations that are energy intensive, harder to reliably detect and more ambiguous during day-to-day operation of the robot.

Karp (1961) was the first to study the problem and proposed an exponential time integer linear programming solution. Several methods to reduce algorithm run-time of designing optimal prefix-free codes with unequal letter costs have been proposed but all impose constraints on the problem. In Bradford et al. (2002), the authors restrict the letter costs to a binary set. In Golin and Rote (1998), the authors propose a dynamic programming algorithm to build the tree in a top-down fashion, where the costs are integers. We implement the latter algorithm in our method due to its flexibility with the cost requirements. The algorithm runs in polynomial time but it

is still unclear whether the general problem with non-integer costs is polynomial-time solvable or $\mathcal{NP}$-hard.

CHAPTER 3
Visual Tracking

In this chapter, we present our vision-based approach for tracking the 3D pose (translation and orientation) of an autonomous robot in underwater environments. Our tracker enables robust multi-robot convoying, which has been studied extensively in the robotics research community under a variety of settings ranging from self-driving cars (Schneiderman et al., 1995a) to aerial drones (Lugo et al., 2013), but relatively little in the marine context. In the context of automated surveillance of marine ecosystems, convoying enables the deployment of highly configurable heterogeneous robot teams in which each robot collects data using different sensing devices. In contrast to deploying a single highly-capable robot, this distributed approach is more scalable and less prone to a single point of failure (Dudek et al., 1995).

While prior work has used fiducial markers on the target to simplify the visual detection task (Schneiderman et al., 1995a), our method relies solely on the natural appearance of the target. In comparison, our approach overcomes difficulties due to the marker not always being visible as the target robot is seen from a variety of poses. Moreover, in contrast to previous methods that require expensive hardware such as mobile beacons (Chandrasekhar et al., 2006) or acoustic sampling (Corke

et al., 2007) to achieve localization underwater, we capitalize on commodity RGB cameras.

3.1 Joint Object Detection and Pose Estimation

We define a multi-output convolutional network that consists of a robot classifier, an orientation regressor (in quaternion form) and a bounding box detector. We train the network on monocular images generated synthetically using the robot's CAD model and various backgrounds constructed in the Unreal (EpicGames, 2018) game engine. The classifier simply outputs a probability p of the image containing the robot. The orientation regressor head of the network outputs the robot's orientation as a normalized quaternion vector $q = (w, x, y, z)$. The bounding box detector outputs a vector b outlining the diagonal coordinates of the bounding box: $(x_{min}, x_{max}, y_{min}, y_{max})$. These coordinates are normalized with respect to the width and height of the image to lie in $[0, 1]$. Note that directly regressing 3D relative translation will prevent the trained network from operating on cameras with varying focal lengths, so we compute it instead from the estimated orientation and bounding box, as we will discuss later.

We design our pose estimation network using the VGG16 architecture as a baseline as it has been demonstrated to perform well as a feature extractor, namely on visual classification tasks (Simonyan and Zisserman, 2014). VGG is a convolutional neural network architecture which achieves top-5 visual classification accuracy on the ImageNet dataset (Deng et al., 2009), an image dataset of over 14 million images hand-annotated with over 20 000 classes. While our task is not classification exclusively, the VGG network acts as a good starting feature extractor. We use VGG

weights from pre-training the VGG network on ImageNet. Our images are resized to $(224, 224, 3)$ to match ImageNet's scaling. We discard the VGG fully connected layers (FC) and augment the network for orientation regression with four FC-ReLU layers, the bounding box regression with two FC-ReLU layers and classification with two FC-ReLU layers.

The loss function we minimize combines the binary cross-entropy for the classification, the L2-norm of the four coordinates of the bounding box and the L2-norm of the orientation quaternion with lambda scale factors to balance the losses. Formally, it is defined as:

$$L = \lambda_b * L_b + \lambda_q * L_q + \lambda_c * L_c \tag{3.1}$$

where the L2-norms are defined as

$$L_q = \frac{1}{2n} \sum_x ||q_{gt} - q||^2 \tag{3.2}$$

and

$$L_b = \frac{1}{2n} \sum_x ||b_{gt} - b||^2 \tag{3.3}$$

and the binary cross-entropy is defined as

$$L_c = -\sum_x (p_{gt} \log(p) + (1 - p_{gt}) \log(1 - p)) \tag{3.4}$$

To obtain the relative translation t^* of the robot, we solve the following minimization problem:

$$t^* = \underset{t}{\operatorname{argmin}} \|b - \pi(q,t)\|_2 \tag{3.5}$$

where $b = (x_{min}, x_{max}, y_{min}, y_{max})$ is the detected bounding box, and $\pi(q,t) = (x^{\pi}_{min}, x^{\pi}_{max}, y^{\pi}_{min}, y^{\pi}_{max})$ is the projected bounding box of the robot at translation t with orientation q. Note that computing $\pi(q,t)$ requires that the camera's intrinsic parameters and the robot's absolute scale are known.

During convoying, we run a Kalman-Filter (Welch and Bishop, 1995) based tracker on the follower robot to integrate the detected translation and orientation of the leading robot and the bounding box over time. We use a PID controller to interface with the autopilot on the follower so that it maintains a fixed pose relative to the leader while both robots swim.

3.2 Evaluation

We first evaluate the pose estimation network in isolation on a real underwater test set by measuring mean angle errors across the dataset as well as their distributions over angles. We then evaluate the tracking performance by analyzing the mean orientation errors across entire trajectories of varying complexity using the Kalman-Filtered pose and bounding box estimates. We do this on both synthetic and real trajectories.

3.2.1 Pose Estimation Dataset

Our pose estimation dataset consists of a mixture of synthetic images generated with Unreal Engine (EpicGames, 2018) and real manually annotated underwater images collected during field trials at McGill University's Bellairs Research Institute in

Barbados. Unreal Engine is a free-to-use C++ game development engine developed by Epic Games, first released in 1998. More recently, it has become a popular platform to develop simulations because of its high flexibility and powerful rendering and physics engines. We restrict the training set to only use synthetic images generated with Unreal. This setup highlights our goal of avoiding dependence on the tedious manual annotation process by relying on easily generated synthetic data for training. Our experiments display the network's ability to generalize to real images.

Figure 3–1: Image samples from synthetic training data generated using Unreal.

Unreal synthetic dataset

The training data consists of 50 000 synthetic images generated in Unreal (EpicGames, 2018) with 45 lighting variations (including varied angle and intensity), 2 robot chassis materials (matte and shiny), 2 custom parts variations on the robot on 4 sets of backgrounds, including a simulated custom-designed underwater world, a simulated pool, and random textures. We opt to use synthetic images exclusively because of our ability to generate a much larger dataset as opposed to using real footage and also in order to have an unbiased separate real dataset for evaluation.

In order to increase the realism of the simulated environments, professionally made photo-scanned assets and textures are used from Quixel Megascans (Quixel, 2018), including rocks, coral, sand, fish, and plant life. The simulated underwater environment used for both dataset generation and evaluation are inspired by previous footage and data from field trials in Barbados and is approximately $0.5km$ by $0.5km$. An image of the environment is presented in Fig. 3–2. The simulated pool environment is created to match the dimensions of the McGill University pool. Visual effects within Unreal Engine such as Exponential Height Fogs and Post Process Materials are added and tuned to mimic the visibility and hues of the real environments.

The Aqua CAD model is randomly placed in the field of view using a randomly generated pose with $[-85°, 85°]$ bounds on the three rotation axes (roll, pitch, yaw) and $[0.5m, 2.5m]$ bounds on the robot's distance from the camera. These bounds represent the realistic range of operation of the robot during convoying tasks. Image samples of the training data are shown in Fig. 3–1.

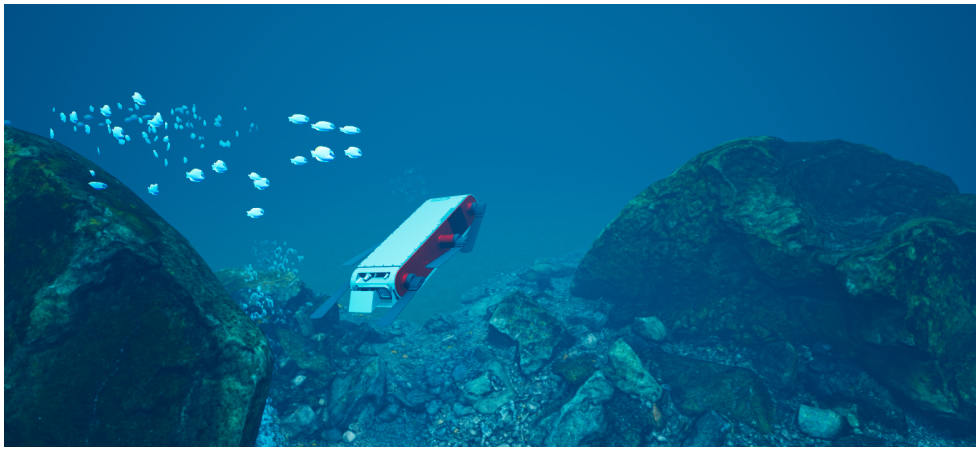


Figure 3–2: Aqua in the simulated underwater environment generated in Unreal.

Real underwater dataset

Our test set consists of 1 000 real images collected during underwater field trials off the west coast of Barbados. The images are captured from diver-held GoPro cameras and an Aqua robot's on-board camera. We annotate the 6-DoF pose of the robot in each of these images using a custom-built annotator, which allows the user to mark keypoints on the robot assigned from the CAD model. The annotator then iteratively fits a wireframe to the robot using its known dimensions in order to generate the ground truth pose.

3.2.2 Trajectory Generation

Unreal synthetic trajectories

In order to evaluate our tracking performance in simulation, we generate a dataset of synthetic videos showcasing the robot performing a variety of trajectories in the Unreal underwater environment. This allows us to quickly compare our

tracker's projected trajectory to the Aqua's actual swimming path without the tedious process of annotating real underwater footage frame by frame.

In Unreal Engine, we use the custom-designed underwater world described in Sec. 3.2.1 as the backdrop with a gamepad-controlled simulated Aqua model. About 2 meters behind the Aqua model, we place a simulated camera that serves as the tracking camera. Default settings in Unreal Engine cause the camera to translate and rotate perfectly with the object it is following, so we introduce artificial lag that allows the target Aqua to partially escape the camera frame, but to never fully leave the camera view. We then record four distinct videos of the Aqua maneuvering the simulated underwater environment from the point of view of the tracking camera, restricting different rotation axes for each. This provides us with trajectories showcasing the robot rotating only along a) yaw, b) yaw and pitch, c) yaw and roll, and finally d) roll, pitch, and yaw. Along with each saved frame from the tracking camera, ground truth information including the 6-DoF pose and bounding box of the target Aqua and the pose of the camera are stored as a ROS bag file and can be replayed on the real robot if necessary.

Real underwater trajectories

We generate a dataset of underwater trajectories from footage captured by an Aqua of a secondary target Aqua in a multitude of underwater environments in Barbados. Unlike the synthetic trajectories, these real trajectories do not display a high amount of variability across the roll and pitch axes and do not include as much clutter. We opt to have higher variability in our simulation to increase the robustness of the system and future-proof our system against future applications.

In each video, we annotate the ground truth 6-DoF pose every 10 frames using our custom annotator and evaluate the trajectory errors over the annotated frames. These trajectories are used in the next section to evaluate tracking performance.

3.3 Experimental Results

We summarize results from evaluating both the 3D detection and pose estimation in isolation on our real images test set and the tracking performance on synthetic and real trajectories.

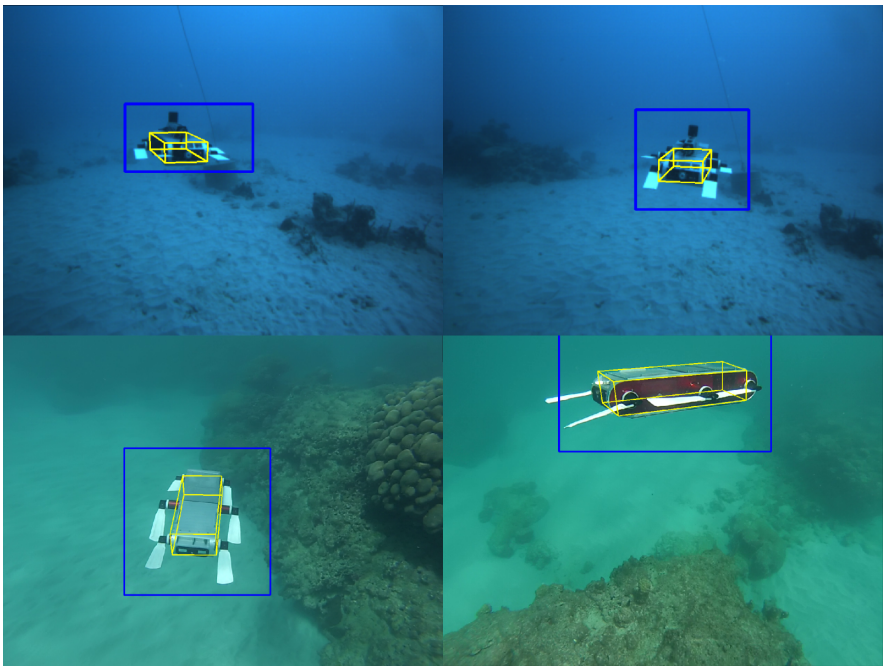


Figure 3–3: Sample detections and pose estimates on test images.

3.3.1 Pose Estimation

Qualitative detection results showcasing the wireframe of the robot overlaid on test images are presented in Fig. 3–3. The images used for this evaluation are real

underwater images collected in Barbados from the test set described in Sec. 3.2.1. We summarize the performance of our model on the test set in Tab. 3–1. The table shows that pitch dominates the rotational error. The mean rotation error represents the mean angle difference between the predicted and ground truth quaternions over the dataset.

Table 3–1: Base metrics evaluated over the real underwater test set collected in Barbados, referred to in Sec. 3.2.1.

Mean Rotation Error	**Mean Roll Error**	**Mean Pitch Error**	**Mean Yaw Error**
23.51°	7.29°	12.05°	5.87°

We also measure the recall of the orientation estimate from the test set over a number of θ thresholds, 30° usually being the default. Here, recall is simply the number of angle predictions that have a rotation error of less than threshold θ over the entire dataset. Our results are presented in Tab. 3–2.

Table 3–2: Orientation estimate recall for different θ thresholds.

	60°	45°	30°	22.5°	15°	7.5°
Recall	0.89	0.79	0.57	0.40	0.21	0.03

A plot of a randomly sampled subset of angle errors relative to their respective angle value shows a concentration of errors below the 20° mean orientation error. While pitch dominates the rotational error as per Tab. 3–1, Fig. 3–4 shows that pitch errors tend to jump after the 50° pitch angle. This might be explained by the loss of view of some important key features of the robot's back. As the robot's top plate is plain aluminum with very little variation, our network might find it more

difficult to extract enough features to distinguish pitch angles beyond this mark. The behavior is slightly less present with the yaw axis and almost non-existent in the roll axis which in particular maintains a good view of the back plates of the robot.

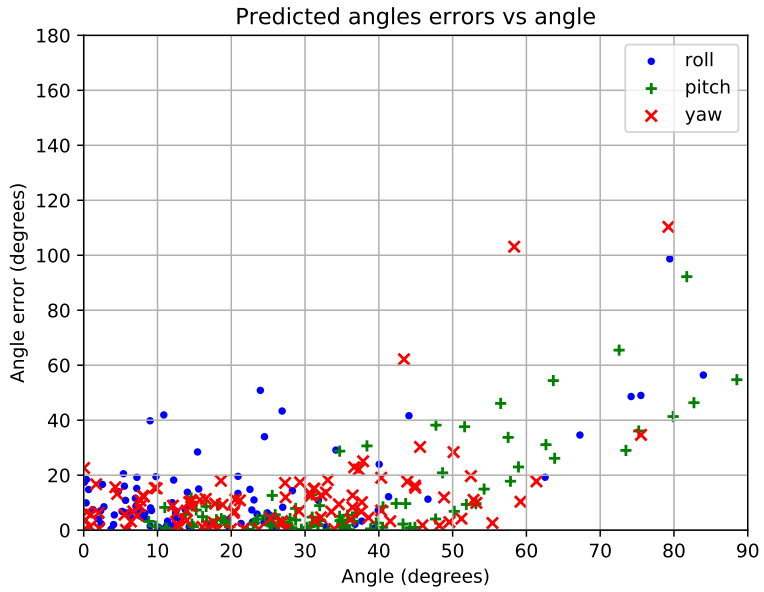


Figure 3–4: Angle errors vs angle values for yaw/pitch/roll from a subset of the test set.

3.3.2 Trajectories

We measure the root mean squared error (RMSE) of the translation and the orientation of the target's Kalman-filtered pose against the ground truth trajectories that were generated in Unreal or annotated manually. For synthetic trajectories, we restrict various axes of rotations in order to understand the impact of certain axes on the orientation RMSE. The results of our evaluation are summarized in Tab. 3–3.

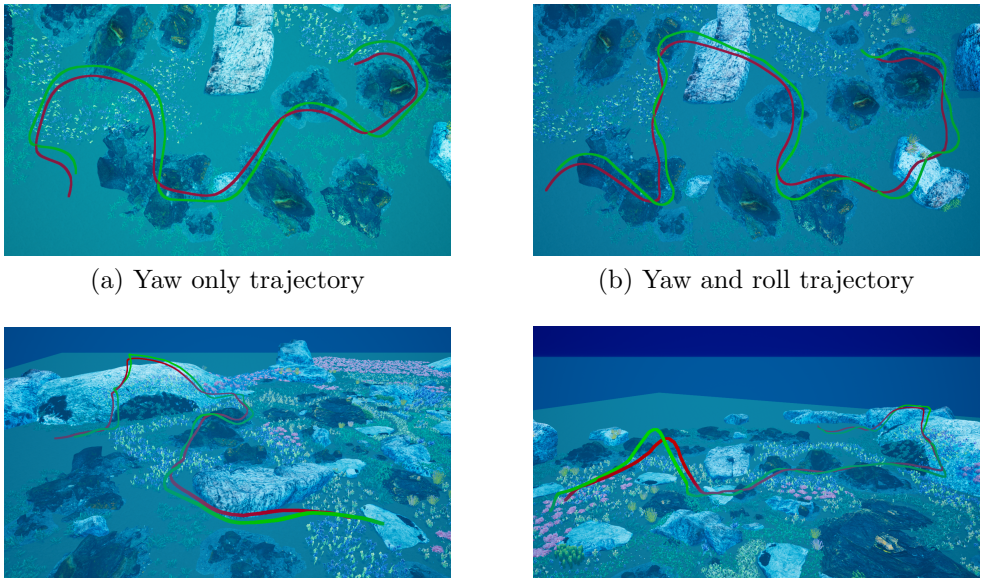
We note that our calculated translation results in an average error of $< 1m$ while orientation RMSE is on average $< 30°$, consistent with our test set evaluation.

As mentioned in Sec. 3.2.2, the real trajectories display a much lighter amount of clutter and much more predictable paths with less variations across the roll/pitch axes. This explains the overall inferior performance on the synthetic trajectories, which include fish animations, heavy rocks and corals and a variety of textures. However, the table demonstrates our system's ability to robustly track the robot in real underwater trajectories despite never training on real images.

Table 3–3: Kalman-Filtered pose errors over synthetic and real trajectories. The tracking camera is located within a range of $[0.5m, 2.5m]$ from the target.

Sequence	**Sequence Length (s)**	**Translation RMSE (m)**	**Orientation RMSE**	**Roll RMSE**	**Pitch RMSE**	**Yaw RMSE**
Synth Y Only	68	0.60	21.76°	10.92°	8.44°	19.20°
Synth Y/P	75	0.61	32.86°	29.11°	25.5°	17.37°
Synth Y/R	70	0.79	21.65°	10.63°	14.52°	14.60°
Synth Y/P/R	83	1.29	31.97°	26.57°	16.69°	16.93°
Real BBD 1	70	0.72	17.59°	11.87°	4.59°	12.11°
Real BBD 2	40	1.17	14.88°	11.74°	5.87°	7.12°
Real BBD 3	30	0.40	20.96°	14.23°	7.19°	14.40°

To better visualize the trajectories, we include plots of the ground truth trajectories and tracked trajectories overlaid on the underwater environment for the synthetic trajectories in Fig. 3–5. It is important to note that the tracking robot was not using the pose estimates entirely to follow the target in this case. It is pre-set to follow the robot within a certain window as explained in Sec. 3.2.2.



(a) Yaw only trajectory

(b) Yaw and roll trajectory

(c) Yaw and pitch trajectory

(d) Yaw, pitch and roll trajectory

Figure 3–5: Plotted ground truth (green) and tracked (red) trajectories.

CHAPTER 4
Visual Communication

In this chapter, we leverage the rich geometric information recovered from 3D target tracking for gesture-based communication. In robot convoying, (Dudek et al., 1995) leverages gestures performed by the target to communicate its intended heading to the follower, which makes the overall convoy more robust. Inspired by this line of work, we design a visual communication system capable of transmitting generic messages based on 3D gesturing.

Our visual communication system consists of two main components: 1) an optimal prefix-free encoding of poses where each codebit corresponds to an orientation bin with a defined transmission cost and 2) a visual decoder which relies on our CNN-based orientation regressor to detect the 3D orientation of the robot to in turn decode the codeword.

4.1 Optimal Prefix-Free Encoding of Poses

We consider the problem of efficiently encoding a set of messages based on robot pose configurations. These messages can include urgent announcements, commands, and parameters to be passed between robots deployed in the field on a collaborative task. An example list of messages are:

- HELP
- DANGER
- LOW_BATTERY
- U_TURN
- START_MAPPING
- GO_TO_DIVER_X
- DESCEND_X_METERS
- STOP

Let n be the number of messages we wish to encode and communicate using an encoding alphabet $\Sigma = \{\beta_1, ..., \beta_r\}$ which consists of r *codebits*.

Each codebit β_i is associated with a *transmission cost* $c_i = T(\beta_i)$ and a *codeword* $cw = \{\beta_{i_1}\beta_{i_2}...\beta_{i_k}\}$ - a list of codebits from Σ. A codeword has a transmission cost equivalent to the sum of the costs of its individual codebits:

$$T(cw) = \sum_{j=1}^{k} c_{i_j} \tag{4.1}$$

A *code* W is defined as the set of codewords $cw_1, ..., cw_n$ and is considered *prefix-free* if no codeword $cw \in W$ is a prefix of another. For example, a code containing codewords $\{\beta_1, \beta_1\beta_4, \beta_3\beta_3\}$ is not prefix-free as the first two codewords share the same prefix. We can then define the cost of a code as the expected transmission cost of a codeword:

$$C(W) = \sum_{i<=n} T(cw_i) \cdot p_i \tag{4.2}$$

where p_i is defined as the probability of transmitting message i.

In order to choose codebits of the encoding alphabet Σ, the orientation space of the robot is binned. The roll, pitch and yaw axes are each discretized into bins of size θ_r°, θ_p°, θ_y° respectively. We then take the combinations of the bins from each axis

to represent the codebits. Individual axes can be ignored as needed depending on the robot capabilities. In this paper, we choose to forego the roll axis to maintain a smaller number of codebits which is sufficient for our needs. For an example list of codebits, please see Tab. 4–1.

Table 4–1: Example list of 9 codebits generated from binning of the yaw and pitch axes with $\theta_p = \theta_y = 60°$ and associated angles.

Codebit	**Roll (°)**	**Pitch (°)**	**Yaw (°)**
β_1	0	-60	-60
β_2	0	0	-60
β_3	0	60	-60
β_4	0	-60	0
β_5	0	0	0
β_6	0	60	0
β_7	0	-60	60
β_8	0	0	60
β_9	0	60	60

To assign codewords to messages, we sort the messages by their probability, and assign higher probability messages to codewords with lower transmission cost.

We define a transmission cost function that is based on three constraints:

$$T(\beta_i) = p(\beta_i) \cdot \bar{e}(\beta_i) \cdot d(\beta_i) \tag{4.3}$$

where we define p, $\bar{e}$ and d as:

- $p(\beta_i)$: the probability of a codebit in regular operation. This value allows us to ensure high probability codebits are penalized and not used in our code so that gestures are not confused with regular operation. In order to obtain this

probability distribution, we run our pose estimator on footage of the robot in operation and extract the histogram of orientation bins.

- $\bar{e}(\beta_i)$: the normalized mean error of the orientation regressor when executed on the corresponding bin of the codebit. This helps avoid using difficult to detect codebits in our encoding.
- $d(\beta_i)$: an application-specific value which can represent the time it takes to execute a codebit, or other engineering restrictions in maintaining a certain codebit, also normalized to $[0, 1]$. This penalizes gestures that are difficult to execute.

where the input to each of these measures is the bin that corresponds to β_i as defined previously. Based on the calculated transmission cost, a cut-off could be used to eliminate certain codebits from the encoding alphabet. A histogram showing the probability of codebits in regular operation is presented in Fig. 4–1.

Given our list of codebits and their associated costs, we implement the optimal prefix-free dynamic programming algorithm presented by (Golin and Rote, 1998) to obtain the code-tree that minimizes the total cost of the prefix-free code. An example code tree is presented in Fig. 4–2.

4.2 Visual Decoding of Poses

Using the codewords from the tree generated in Sec. 4.1, we can now execute each encoded message on the robot. In order to simplify the decoding algorithm, we insert a neutral codebit β_z between every codebit in a codeword. This serves as a marker to register when every codebit is executed. For example, codeword $\{\beta_1\beta_2\}$ becomes $\{\beta_1\beta_z\beta_2\beta_z\}$. Fig. 4–3 shows Aqua executing the codeword $\{\beta_4\beta_z\beta_2\beta_z\}$

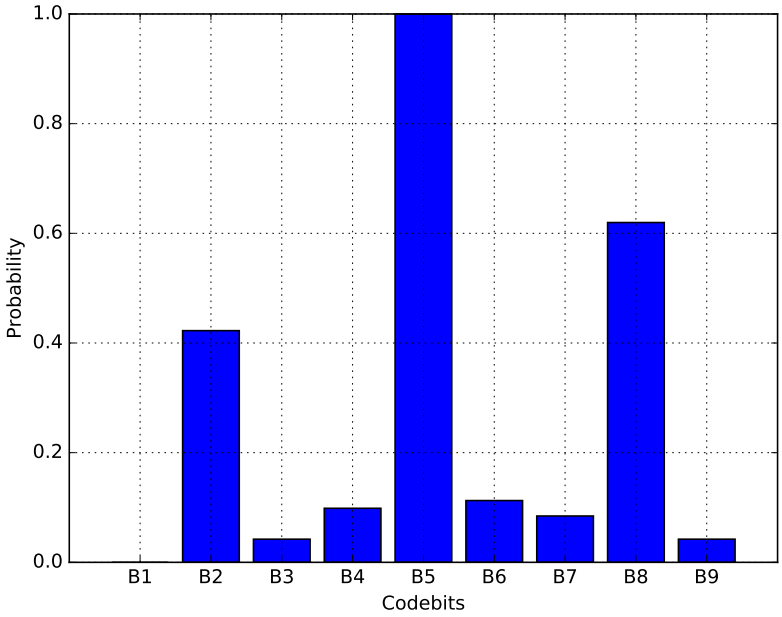


Figure 4–1: A histogram showing the codebits probabilities captured from 5 minutes of footage of regular operation of the robot. We discourage the use of high probability codebits in message encoding to prevent the false detection of gestures during regular operation.

In order to get an orientation estimate, we rely on our pose estimator from Chapter 3 trained on our training datasets as presented in Sec. 3.2.1. Once we have an orientation estimate, we proceed to bin said estimate according to the bins chosen in Sec. 4.1. To account for pose estimation errors and viewpoint variations, the codebits are detected if they are within a bin of the target codebit angles with the bin limits offset from the center by $[-20, 20]$.

In order to decode an executed codeword, we obtain the filtered pose estimate on every frame and bin the orientation estimate. We then check if the bin corresponds to any codebits of the encoding alphabet. If we have detected a valid codebit that is

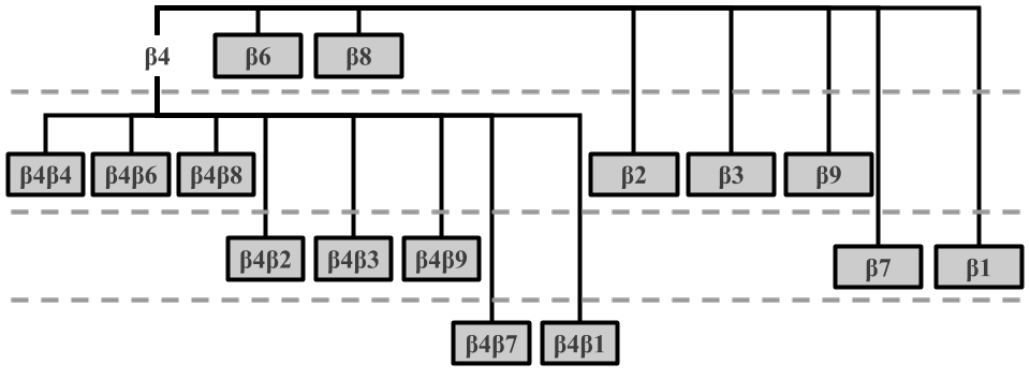


Figure 4–2: An example optimal prefix-free code tree using the codebits from Tab. 4–1 with $r = 8$ codebits and $n = 15$ messages. The leaves of the tree (highlighted in gray) represent the final codewords that make up the code. The cost associated with each codebit is defined in the following list: $c = [1, 1, 1, 2, 2, 2, 3, 3]$ for codebits $[\beta_4, \beta_6, \beta_8, \beta_2, \beta_3, \beta_3, \beta_9, \beta_1, \beta_7]$ respectively. The costs are the equivalent of the depth level of the tree.

not identical to the previous detected ones, we check if this codebit β_t is a prefix of any of our codewords. If we are already tracking a candidate sub-codeword $cand_{t-1}$, we instead check if $cand_t = cand_{t-1} \cup \beta_j$ is a prefix of a codeword in code W. If it is a codeword in W, we have detected a message.

The prefix-free nature of the code means that codewords are non-ambiguous and the transmission costs used to generate the code help to ensure that codewords are not confused with regular pose configurations that occur on a normal execution of the robot.

Note that this algorithm assumes the observer is mostly following the target and looking at it from a limited viewpoint window, as shown in the training datasets of the pose estimator. The observer can have translation offsets but generally assumes the target is executing messages with a local frame of reference that is relative to its

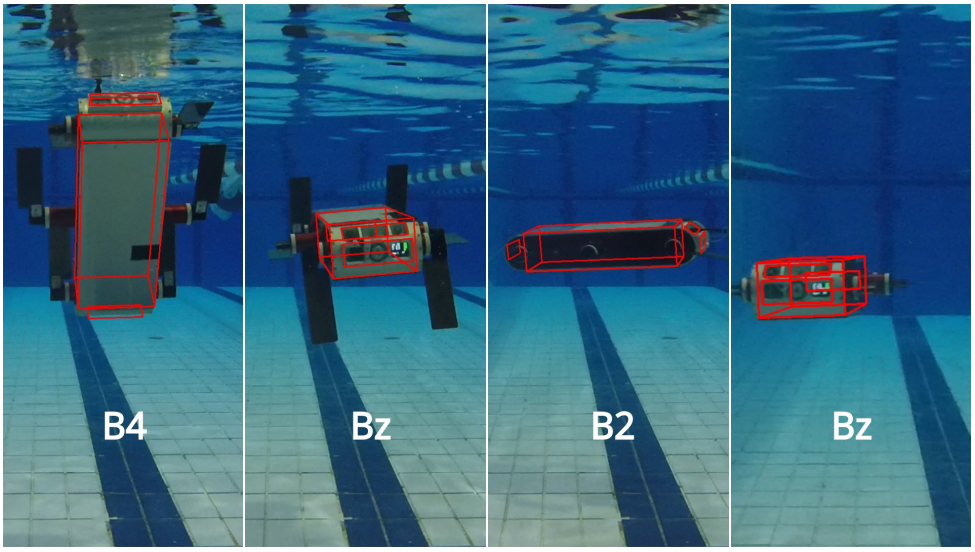


Figure 4–3: An Aqua robot performing codeword $\beta_4\beta_2$ in the pool with neutral codebits β_z in between.

camera. To better handle smaller viewpoint variations expected from any moving observer, we update the codebit bin centers according to the latest neutral codebit detected and its offset from its original neutral codebit center up to $[-10°, 10°]$.

4.3 Experimental Results

We evaluate our system on both synthetic and real footage of the Aqua executing codewords both statically and mid-swimming. We present our dataset collection setup and the performance on these datasets in the following sections.

4.3.1 Visual Encoding Dataset

To evaluate our visual encoder, we prepare a dataset containing the Aqua executing codewords in both real pool trials and in the Aqua simulator (Manderson et al., 2018). To create this testing dataset, we record both generated synthetic videos

of a simulated robot and real videos of the physical robot executing the motions that correspond to a subset of the codewords generated in Fig. 4–2.

Simulated gestures using Unreal engine

The synthetic testing data is comprised of recordings of the simulated robot executing motions for each of the codewords listed in Tab. 4–2 within the realistic simulated underwater environment, totalling 50 recordings per codeword.

To execute the motions, a controller within the Aqua simulator is fed a target orientation offset, θ_{β_i}, corresponding to a codebit along with the time allowed, Δt, for the Aqua to reach the target orientation.

Once the target is reached, there is a pause for approximately 1.5 seconds before returning to the neutral orientation and continuing onto the next codebit or codeword. This simple, idealized dynamics model gives a solid baseline for comparing real world examples.

Variations to each execution of a codeword include a) a random starting orientation within the simulated underwater environment with bounds of $[-10°, 10°]$ for roll, $[-20°, 20°]$ for pitch, and $[-180°, 180°]$ for yaw, b) random additions to the target orientation for each axis with bounds of $[-5°, 5°]$, and c) changes in speed of the robot through random scaling of the amount of time allotted for each motion, normally 2 seconds, with bounds of [.8, 1.2]. Each random variable is chosen with uniform distribution.

Images of the synthetic Aqua performing a particular codebit in Unreal are presented in Fig. 4–4.

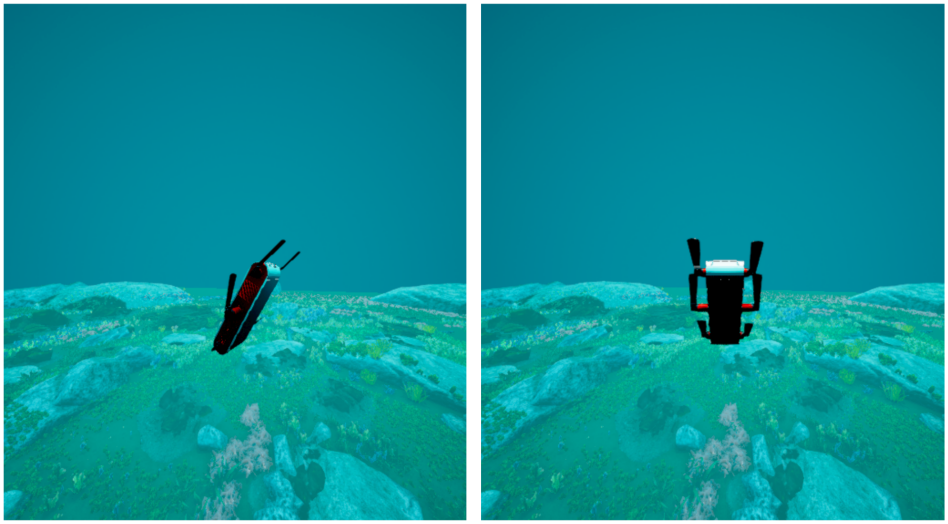


Figure 4–4: Synthetic Aqua robot positioned according to codebit β_1 (left) and β_4 (right).

Gestures in the pool

In this section, we describe the generation and recognition of gestures using the actual Aqua robot in a swimming pool. By working in a pool, we can evaluate the possibility of our methods in the field and in a more realistic context while still maintaining a domain simple enough in its logistics, and where data can be systematically collected.

The resulting testing dataset used to evaluate visual encoding in this real world setting consists of, on average, 10 recorded examples of the Aqua executing gestures in the McGill University pool for each of the codewords listed in Tab. 4–3.

In order to execute the gestures corresponding to codebits on the physical Aqua, a custom PID autopilot controller (Meger et al., 2014) is utilized. Given a target orientation offset for a chosen codebit, the autopilot controller causes the Aqua to

rotate, stopping when the IMU reading indicates the Aqua is within 5 degrees of the target angles. To prevent the Aqua from drifting and accidentally appearing to execute an undesired motion, the controller maintains the neutral orientation for 3 seconds before a new motion is attempted, where the neutral orientation is considered to be the orientation at which the Aqua starts executing a gesture.

Trajectories in simulation

An important evaluation of our method involves decoding messages from a robot as it moves around its environment in regular operation. This evaluation ensures that messages aren't missed while performing basic navigation and ensures the decoder's ability to discern regular operation from messaging. In order to test our visual decoder's performance in simulation, we generate a dataset of 10 synthetic videos showcasing the Aqua executing gestures intermittently as it explores the custom-designed underwater world described in Sec. 3.2.1. Each recording features 1 codeword repeated 5 times at random over the course of approximately 2 minutes of navigation.

To generate this data, we place a simulated camera approximately 2 meters behind a simulated Aqua model. Default settings in Unreal Engine cause the camera to translate and rotate perfectly with the object it is following, so we introduce artificial lag to the camera's rotation to simulate the delay that would occur in a real trial as either a human or robot attempts to follow a gesturing Aqua. The videos recorded by the simulated camera are stored in a ROS bag along with ground truth information, including the 6-DoF pose and bounding box of the target Aqua and the pose of the camera.

4.3.2 Static Visual Decoding

In order to simplify our deployment and encoding alphabet, we forego the roll axis and generate codebits by using 3 bins with $\theta_y = 60°$ and $\theta_p = 60°$, restricting the orientation space to $\{-90°, 90°\}$. The corresponding codebit list is shown in Tab. 4–1. We assign the neutral codebit to be $\beta_z = \beta_5$.

To derive the transmission cost of each codebit, we plot the probability of codebits in Fig. 4–1 and a randomly sampled subset of angle errors relative to their respective angle value in Fig. 3–4.

The dynamic programming algorithm we implement from (Golin and Rote, 1998) optimally encodes n messages in $O(n^{C+2})$ time and $O(n^{C+1})$ space where C is the highest integer cost assigned to a codebit. We restrict our evaluation on a simpler cost list $c = [1, 1, 1, 2, 2, 2, 3, 3]$ for codebits $[\beta_4, \beta_6, \beta_8, \beta_2, \beta_3, \beta_3, \beta_9, \beta_1, \beta_7]$ respectively. This cost list is an integer cost list which is reflective of the order of the transmission costs as opposed to their values. We generate the optimal prefix-free code-tree for $r = 8$ and $n = 15$, as shown in Fig. 4–2 in roughly 4h of runtime.

To measure our decoding performance, we evaluate our decoder on the Unreal and real underwater gestures datasets described in Sec. 4.3.1. We generate confusion matrices for each dataset and summarize precision and recall values in the matrices in Tab. 4–2 and Tab. 4–3.

On synthetic data, the visual decoder achieves a mean precision of **0.96** and mean recall of **0.91**. Note that the requirement for the robot to return to a neutral codebit β_z results in some missed detections of certain messages, as can be seen in the *False negatives* (FNs) column.

Table 4–2: Confusion matrix for codewords executed in Unreal. Class-specific recall values are highlighted in grey.

	$\beta_4\beta_4$	$\beta_4\beta_2$	$\beta_4\beta_1$	$\beta_4\beta_7$	β_2	β_1	β_3	β_7	**FNs**
$\beta_4\beta_4$	0.94	0.	0.	0.	0.	0.	0.	0.	0.06
$\beta_4\beta_2$	0.04	0.82	0.	0.	0.08	0.	0.	0.	0.06
$\beta_4\beta_1$	0.02	0.06	0.84	0.	0.	0.04	0.	0.	0.04
$\beta_4\beta_7$	0.	0.	0.	0.98	0.	0.	0.	0.02	0.
β_2	0.	0.	0.	0.	0.94	0.	0.	0.	0.06
β_1	0.	0.	0.	0.	0.04	0.86	0.	0.	0.10
β_3	0.	0.	0.	0.	0.	0.	1.	0.	0.
β_7	0.	0.	0.	0.	0.	0.	0.	0.90	0.10
Precision	0.94	0.93	1.00	1.00	0.89	0.96	1.00	0.98	

Table 4–3: Confusion matrix for codewords executed in the pool. Class-specific recall values are highlighted in grey.

	$\beta_4\beta_4$	$\beta_4\beta_2$	β_2	β_1	β_6	β_8	**FNs**
$\beta_4\beta_4$	0.67	0.	0.	0.	0.	0.17	0.17
$\beta_4\beta_2$	0.	0.67	0.17	0.	0.	0.	0.17
β_2	0.	0.	0.73	0.	0.	0.	0.27
β_1	0.	0.	0.38	0.25	0.	0.	0.38
β_6	0.	0.	0.	0.	0.83	0.	0.17
β_8	0.	0.	0.	0.	0.	0.82	0.18
Precision	1.00	1.00	0.67	1.00	1.00	0.95	

On real data, our visual decoder achieves a mean precision of **0.94** and mean recall of **0.66**. An explanation for the particularly worse performance of the system in the real pool on codebit β_1 is the imperfect execution of it by the physical robot. Codebit β_1 featuring both yaw and pitch variations were found more likely to overshoot and undershoot on pitch. Fine-tuning of the autopilot controller for such tasks can help mitigate these errors. Most notably, codebit β_1 executions tend to not pitch enough and were at times more closely executed as codebit β_2. Slight overshoot in the yaw axis also lead to more false negatives than expected as the robot skipped

the neutral codebit at times which is supposed to signal the end or transition of a codeword. These errors in executions are typical of real systems deployed in the field. A way to tackle these limitations is to use codebits that are more spread out in the orientation space of the robot to allow some room for error.

4.3.3 Visual Decoding on Swimming Trajectories

We evaluate our system on synthetic swimming trajectories of the Aqua in order to better understand the performance of the visual decoder in a deployment setting. The dataset, described in Sec. 4.3.1, consists of typical swimming trajectories with messages communicated at random times. The goal of this evaluation is to ensure the reliability of the code even when the robot performs a variety of swimming poses. We summarize the precision/recall values on these trajectories in Tab. 4–4. Common false negatives are codebits β_2 and β_8 which represent basic left and right yaw configurations. As shown in Fig. 4–1, these codebits have a high probability of occurrence in regular deployment and our simplified cost structure did not fully capture this cost. Using the more refined transmission cost defined in Sec. 4.1 would help mitigate this issue and ensure these codebits are used less often individually. One can also introduce a cut-off on the codebit probability $p(\beta_i)$ term of the transmission cost and not rely on codebits with high probability.

Table 4–4: Precision/Recall of the decoded messages on synthetic trajectories.

Trajectory	**Codeword**	**Counts**	**Precision**	**Recall**
1	$\beta_4\beta_4$	5	0.83	1.
2	$\beta_4\beta_2$	5	0.83	1.
3	$\beta_4\beta_1$	5	0.67	0.80
4	$\beta_4\beta_7$	5	0.71	1.
5	β_2	5	0.83	1.
6	β_1	5	0.71	1.
7	β_8	5	0.67	0.80
8	β_3	5	1.	1.
9	β_7	5	0.83	1.
10	β_8	5	0.63	1.
		Mean	**0.77**	**0.96**

CHAPTER 5
Conclusion

5.1 Summary

In this thesis, we presented a tracking-by-detection method for tracking the 3D pose of an autonomous underwater vehicle with an aim to improve and enable multi-robot convoying and developed a vision-based communication system between robots in radio-denied environments to robustify robot convoying.

Our tracking method relies on a 3D object detection and pose estimation multi-output convolutional neural network that jointly predicts the target robot's presence in the image, its 3D orientation and the bounding box that encapsulates the target. Combining this information with the robot's known scale and the camera intrinsics, we compute an estimate of the 3D translation of the robot in order to obtain the full 6-degree-of-freedom pose. We trained exclusively on synthetic training data generated in Unreal engine in order to bypass the tedious task of 3D pose manual annotations. Our system demonstrates the ability to transfer its performance from the learned model to a real underwater dataset and achieves a 23.51° mean rotational error over the entire dataset. Using our pose estimation network, we then apply a Kalman-filter on the pose and bounding box and evaluate our system on a variety of

synthetic and real trajectories. In particular, we restrict various axes of rotation on the synthetic trajectories in order to isolate the errors across the axes. Our system achieves a mean translation RMSE of 0.79m and mean orientation RMSE of 23.09° over all the trajectories.

We then leverage our pose estimation network to develop a visual communication system that uses optimal variable-length prefix-free codes reliant on the robot performing various full-body based pose configurations or gestures. This communication system minimizes the likelihood of false positive detection through a defined transmission cost function that seed the selection of the encoding pose configurations used. We demonstrate our system's ability to decode messages on synthetically generated static and swimming sequences with a mean precision and recall of 0.96 and 0.91 respectively, and on real data with a mean precision and recall of 0.94 and 0.66.

5.2 Limitations

It is important to note that our pose estimation network is only trained on a limited range of poses as described in Sec. 3.2.1, specifically the half-sphere where the Aqua robot is facing forward away from the camera. We impose this restriction as we are mainly focused on convoying applications where the back of the robot is in the view at all times, as seen in Fig. 5–1. In order to generalize the pose estimation network, some work needs to be done to ensure symmetry is handled correctly in symmetrical robots such as the Aqua. In addition, this restriction extends to the visual decoder whereas the bins are predefined and assumed to be from the same half-sphere mentioned.

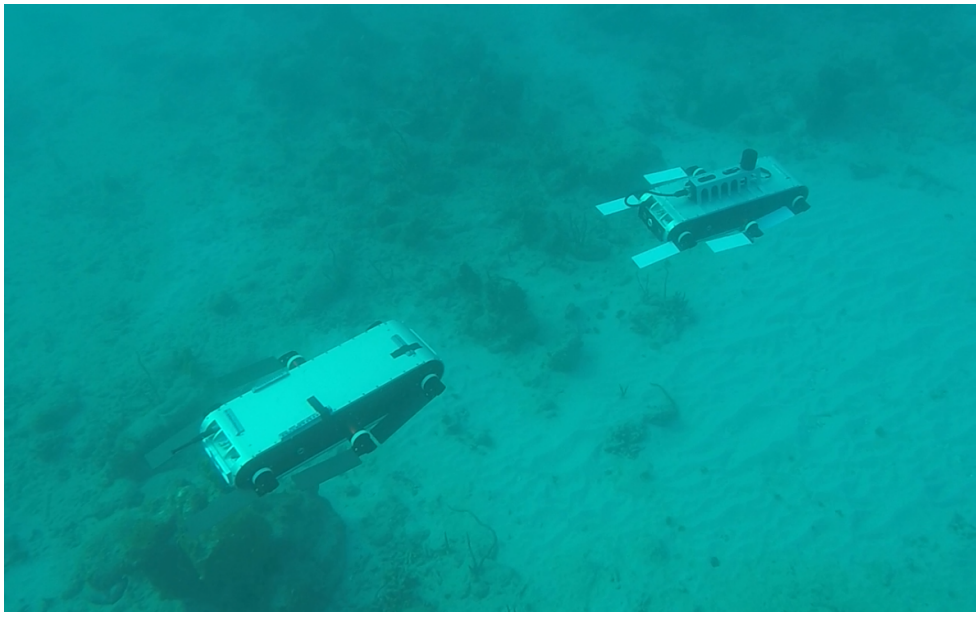


Figure 5–1: A convoy of two Aqua robots showcasing the typical view of a follower Aqua.

5.3 Future Work

While we focus our system on full-body gestures, the pose estimation network can be extended to detect pose configurations of different subsets of a robot. For example, in a dual-arm mobile manipulator platform such as the Baxter robot (Guizzo and Ackerman, 2012) by Rethink Robotics, a visual decoder could be applied only to the arm of the robot to recognize human-like arm-based gesturing.

As mentioned above, augmenting the visual decoder to be fully viewpoint invariant is left as future work.

Another interesting extension is the more formal support of detection and pose estimation on images with a multitude of robots present. Leveraging the 3D pose detected on different robots can allow us to track each robot independently and

correctly decode messages from more than one robot and increase the collaboration potential.

For Aqua, we expect to use this technique in underwater multi-robot convoying missions to signal important messages and achieve more robust tracking, as well as in more complex joint activity scenarios where diver intervention is preferably kept at a minimum. More generally, the aforementioned limitations and suggestions for improvement present many future avenues for exploiting visual gesturing for human-friendly inter-robot communication across many robotic settings and domains. In particular, communicating intent will be one key factor in the better integration of robots as they continue to augment the human workforce and enabling body-language is one exciting step in that direction.

References

Andriluka, M., S. Roth, and B. Schiele (2008). People-tracking-by-detection and people-detection-by-tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8. IEEE.

Baillieul, J. and K. Özcimder (2012, June). The control theory of motion-based communication: Problems in teaching robots to dance. In *2012 American Control Conference (ACC)*, pp. 4319–4326.

Bao, C., Y. Wu, H. Ling, and H. Ji (2012). Real time robust l1 tracker using accelerated proximal gradient approach. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1830–1837. IEEE.

Bradford, P., M. J. Golin, L. L. Larmore, and W. Rytter (2002). Optimal prefix-free codes for unequal letter costs: Dynamic programming with the monge property. *Journal of Algorithms 42*(2), 277 – 303.

Chandrasekhar, V., W. K. Seah, Y. S. Choo, and H. V. Ee (2006). Localization in underwater sensor networks: survey and challenges. In *Proceedings of the 1st ACM international workshop on Underwater networks*, pp. 33–40. ACM.

Collet, A., D. Berenson, S. S. Srinivasa, and D. Ferguson (2009). Object recognition and full pose registration from a single image for robotic manipulation. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 48–55. IEEE.

Corke, P., C. Detweiler, M. Dunbabin, M. Hamilton, D. Rus, and I. Vasilescu (2007). Experiments with underwater robot localization and tracking. In *Robotics and Automation, 2007 IEEE International Conference on*, pp. 4556–4561. IEEE.

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

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Doniec, M., C. Detweiler, I. Vasilescu, and D. Rus (2010, Oct). Using optical communication for remote underwater robot operation. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4017–4022.

Dudek, G. and M. Jenkin (2010). *Computational principles of mobile robotics*. Cambridge university press.

Dudek, G., M. Jenkin, E. Milios, and D. Wilkes (1995, Aug). Experiments in sensing and communication for robot convoy navigation. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, Volume 2, pp. 268–273 vol.2.

Dudek, G., J. Sattar, and A. Xu (2007, April). A visual language for robot control and programming: A human-interface study. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 2507–2513.

Dumitrescu, S. (2006, March). Faster algorithm for designing optimal prefix-free codes with unequal letter costs. In *Data Compression Conference (DCC'06)*, pp. 1 pp.–444.

EpicGames (1998-2018). Unreal engine. https://www.unrealengine.com/en-US/what-is-unreal-engine-4.

Feichtenhofer, C., A. Pinz, and A. Zisserman (2017). Detect to track and track to detect. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3038–3046.

Feil-Seifer, D. and M. J. Mataric (2005). Defining socially assistive robotics. In *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*, pp. 465–468. IEEE.

Fidler, S., S. Dickinson, and R. Urtasun (2012). 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *Advances in neural information processing systems*, pp. 611–619.

Giesbrecht, J. L., H. K. Goi, T. D. Barfoot, and B. A. Francis (2009). A vision-based robotic follower vehicle. In *Unmanned Systems Technology XI*, Volume 7332, pp. 73321O. International Society for Optics and Photonics.

Golin, M. J. and G. Rote (1998, Sept). A dynamic programming algorithm for constructing optimal prefix-free codes with unequal letter costs. *IEEE Transactions on Information Theory 44*(5), 1770–1781.

Guizzo, E. and E. Ackerman (2012, October). The rise of the robot worker. *IEEE Spectrum 49*(10), 34–41.

Hong, S., T. You, S. Kwak, and B. Han (2015). Online tracking by learning discriminative saliency map with convolutional neural network. *CoRR abs/1502.06796*.

Huffman, D. A. (1952, Sept). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE 40*(9), 1098–1101.

Izadinia, H., Q. Shan, and S. M. Seitz (2017). Im2cad. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 2422–2431. IEEE.

Jepson, A. D., D. J. Fleet, and T. F. El-Maraghi (2003, Oct). Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*(10), 1296–1311.

Jones, A. and S. Andersson (2013, June). A motion-based communication system. In *2013 American Control Conference*, pp. 365–370.

Jugessur, D. and G. Dudek (2000, June). Local appearance for robust object recognition. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, Volume 1, pp. 834–839 vol.1.

Karp, R. (1961). Minimum-redundancy coding for the discrete noiseless channel. *IRE Transactions on Information Theory 7*(1), 27–38.

Kendall, A., M. Grimes, and R. Cipolla (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 2938–2946.

Koreitem, K., J. Li, I. Karp, T. Manderson, and G. Dudek (2019). Underwater communication using full-body gestures and optimal variable-length prefix codes. In *IEEE International Conference on Robotics and Automation (IN REVIEW)*.

Koreitem, K., J. Li, I. Karp, T. Manderson, F. Shkurti, and G. Dudek (2018, Oct.). Synthetically trained 3d visual tracker of underwater vehicles. In *MTS/IEEE OCEANS*, Charleston, SC, USA.

Kortenkamp, D., E. Huber, R. P. Bonasso, et al. (1996). Recognizing and interpreting gestures on a mobile robot. In *Proceedings of the National Conference on Artificial*

Intelligence, pp. 915–921.

Kristan, M., J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder (2015). The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 1–23.

Landgraf, T., R. Rojas, H. Nguyen, F. Kriegel, and K. Stettin (2011, 08). Analysis of the waggle dance motion of honeybees for the design of a biomimetic honeybee robot. *PLOS ONE 6*(8), 1–10.

Leonard, J. J. and H. F. Durrant-Whyte (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on robotics and Automation 7*(3), 376–382.

Li, C., Q. Tran, M. Zia, G. Hager, and M. Chandraker (2017). Deep supervision with shape concepts for occlusion-aware 3d object parsing. In *CVPR*.

Lim, J. J., H. Pirsiavash, and A. Torralba (2013). Parsing ikea objects: Fine pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2992–2999.

Lowe, D. G. (2004, November). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*(2), 91–110.

Lugo, J. J., A. Masselli, and A. Zell (2013, Sept). Following a quadrotor with another quadrotor using onboard vision. In *2013 European Conference on Mobile Robots*, pp. 26–31.

Manderson, T. and G. Dudek (2018, Oct.). Gpu-assisted learning on an autonomous marine robot for vision based navigation and image understanding. In *MTS/IEEE OCEANS*, Charleston, SC, USA.

Manderson, T., J. Gamboa Higuera, R. Cheng, and G. Dudek (2018). Vision-based autonomous underwater swimming in dense coral for combined collision avoidance and target selection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Manderson, T., I. Karp, and G. Dudek (2018). Aqua underwater simulator. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop*.

Manz, M., T. Luettel, F. von Hundelshausen, and H.-J. Wuensche (2011). Monocular model-based 3d vehicle tracking for autonomous vehicles in unstructured environment. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2465–2471. IEEE.

Meger, D., F. Shkurti, D. C. Poza, P. Giguère, and G. Dudek (2014). 3d trajectory synthesis and control for a legged swimming robot. In *Proceedings of the IEEE International Conference on Robotics and Intelligent Systems (IROS)*.

Movshovitz-Attias, Y., T. Kanade, and Y. Sheikh (2016). How useful is photo-realistic rendering for visual learning? In *European Conference on Computer Vision*, pp. 202–217. Springer.

Nakata, T., T. Sato, T. Mori, and H. Mizoguchi (1998). Expression of emotion and intention by robot body movement. In *Proceedings of the 5th international conference on autonomous systems*.

Nishimura, H. and M. Schwager (2018). Active motion-based communication for robots with monocular vision. In *Proceedings 2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, pp. 2948–2955.

Peng, X., B. Sun, K. Ali, and K. Saenko (2015). Learning deep object detectors from 3d models. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1278–1286.

Quigley, M., K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Quixel (2018). Quixel megascans. `https://megascans.se/`.

Raghunathan, D. and J. Baillieul (2008, Oct.). Relative motion of robots as a means for signaling. *2173*.

Rekleitis, I., G. Dudek, and E. Milios (2003, Sept.). Probabilistic cooperative localization and mapping in practice. In *IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, pp. 1907–1912. IEEE.

Ren, W. and R. W. Beard (2004). Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints. *IEEE Transactions on Control Systems Technology 12*(5), 706–716.

Roumeliotis, S. I. and I. M. Rekleitis (2004). Propagation of uncertainty in cooperative multirobot localization: Analysis and experimental results. *Autonomous Robots 17*(1), 41–54.

Sattar, J., E. Bourque, P. Giguere, and G. Dudek (2007, May). Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction. In *Fourth Canadian Conference on Computer and Robot Vision (CRV '07)*, pp. 165–174.

Sattar, J., G. Dudek, O. Chiu, I. Rekleitis, P. Giguère, A. Mills, N. Plamondon, C. Prahacs, Y. Girdhar, M. Nahon, and J.-P. Lobos (2008, September). Enabling autonomous capabilities in underwater robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, Nice, France.

Schneiderman, H., M. Nashman, A. J. Wavering, and R. Lumia (1995a, Nov). Vision-based robotic convoy driving. *Machine Vision and Applications 8*(6), 359–364.

Schneiderman, H., M. Nashman, A. J. Wavering, and R. Lumia (1995b). Vision-based robotic convoy driving. *Machine Vision and Applications 8*(6), 359–364.

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

Simonyan, K. and A. Zisserman (2014). Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556*.

Song, S. and J. Xiao (2016). Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 808–816.

Srinivasan, M. V. (2010). Honeybee communication: A signal for danger. *Current Biology 20*(8), R366 – R368.

Su, H., C. R. Qi, Y. Li, and L. J. Guibas (2015). Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2686–2694.

Sun, B. and K. Saenko (2014). From virtual to reality: Fast adaptation of virtual object detectors to real domains. In *BMVC*, Volume 1, pp. 3.

Sutantyo, D. and P. Levi (2015, Jul). Decentralized underwater multi-robot communication using bio-inspired approaches. *Artificial Life and Robotics 20*(2), 152–158.

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

Tribou, M. J., A. Harmat, D. W. Wang, I. Sharf, and S. L. Waslander (2015). Multi-camera parallel tracking and mapping with non-overlapping fields of view. *The International Journal of Robotics Research 34*(12), 1480–1500.

von Frisch, K. (1993). *The dance language and orientation of bees*. Belknap Press. Harvard University Press.

Wang, W., J. Liu, G. Xie, L. Wen, and J. Zhang (2017). A bio-inspired electrocommunication system for small underwater robots. *Bioinspiration and Biomimetics 12*(3), 036002.

Welch, G. and G. Bishop (1995). An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA.

Xiang, Y., W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese (2016). Objectnet3d: A large scale database for 3d object recognition. In *European Conference Computer Vision (ECCV)*.

Yu, Q., T. B. Dinh, and G. Medioni (2008). Online tracking and reacquisition using co-trained generative and discriminative trackers. In *European conference*

on computer vision, pp. 678–691. Springer.