Underwater Communication Using Full-Body Gestures and Optimal Variable-Length Prefix Codes

Karim Koreitem[1], Jimmy Li[1], Ian Karp[1], Travis Manderson[1] and Gregory Dudek[1]

***Abstract*— In this paper we consider inter-robot communication in the context of joint activities. In particular, we focus on convoying and passive communication for radio-denied environments by using whole-body gestures to provide cues regarding future actions. We develop a communication protocol whereby information described by codewords is transmitted by a series of actions executed by a swimming robot. These action sequences are chosen to optimize robustness and transmission duration given the observability, natural activity of the robot and the frequency of different messages. Our approach uses a convolutional network to make core observations of the pose of the robot being tracked, which is sending messages. The observer robot then uses an adaptation of classical decoding methods to infer a message that is being transmitted. The system is trained and validated using simulated data, tested in the pool and is targeted for deployment in the open ocean. Our decoder achieves .94 precision and .66 recall on real footage of robot gesture execution recorded in a swimming pool.**

I. 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. 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, hand gesturing by scuba divers or cyclists signalling their trajectory to drivers and fellow cyclists. 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 setting.

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 [1], [2] 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. In this paper, we focus our application and experiments in an underwater setting, where robot communication is naturally more constrained and other communication methods require additional hardware.

[1]The authors are affiliated with the Mobile Robotics Lab, the Center for Intelligent Machines (CIM), and the School of Computer Science at McGill University in Montréal, Canada. {karimkor, jimmyli, ianfkarp, travism, dudek}@cim.mcgill.ca

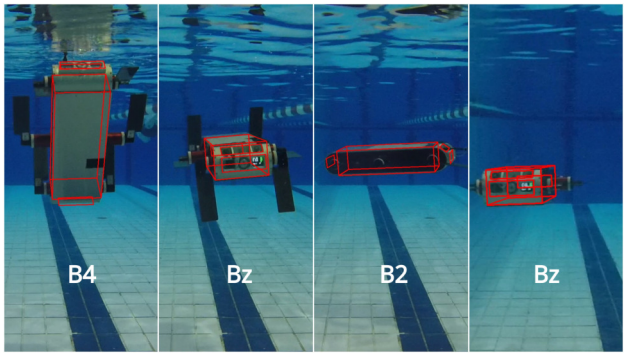

Fig. 1. An Aqua robot performing codeword $\beta_4\beta_2$ in the pool with neutral codebits β_z in between.

We regress 3D orientation of the robot by training a CNN exclusively on synthetic images using CAD renderings of the robot in a synthetic underwater environment designed in Unreal game development engine [3]. This approach eliminates the need of labour-intensive manual labelling of real underwater images and allows us to benefit from domain randomization techniques [4] to robustify our model.

We validate our system on the Aqua family of amphibious hexapod robots [5] shown in Fig. 1. 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 [6] which relies on an onboard 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) [7] framework. Our visual tracker and decoder relies on images captured by the front-facing RGB camera and runs at frame rate on the robot's laptop-grade dual-core Intel i3 CPU and Nvidia GPU (Jetson TX2) [8].

We evaluate our system both in simulation using a synthetic marine environment designed in a custom Aqua simulator [9] built on Unreal and in a real underwater context from both marine and pool settings. Our evaluation demonstrates successful generalization of the synthetically trained pose estimator to real images.

II. Related Work

A. Visual Tracking

In our prior work, we have presented underwater tracking-by-detection [10] systems that operated on both 2D bounding box detection [11] and 3D pose estimation [12]. In both

approaches, detection of the target in every frame is integrated using filtering. Other approaches to tracking include the use of CNNs in 2D [13], sparse vectors [14] and other techniques [15], typically based on 2D image models. As in prior work [12], [16], 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 [17], [17]–[21].

To avoid the tedious task of data labelling, there has been significant work on using synthetic images for training [4], [22]–[25]. We take a similar approach by training our network on synthetic data generated in Unreal. We also vary backgrounds and textures in our synthetic data which helps our learned model generalize to real images without needing to train on them.

B. Robot Communication and Visual Communication

Several authors have considered the utility and nature of gesture-based communications in robotics [26], [27]. There has also been some prior work on allowing robots to communicate via body movement [28] or mutual observation [29], [30]. In [31], 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.

In marine environments, radio communications are often impractical or impossible but several authors have examined interesting alternatives [32]–[34]. In contrast, whole-body motions and activities is widely employed in the animal kingdom and has also inspired several robotics efforts [31], [35]–[41]

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.

C. 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 [42]–[44]. Such an encoding represents a generalization of the classical Huffman coding problem [1] 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 alphabet $\{\cdot, -\}$. In our setting, the variable cost is also an excellent way of penalizing pose configurations that are energy intensive, harder to reliably detect and more ambiguous during day-to-day operation of the robot.

Karp [42] 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 [44], the authors restrict the letter costs to a binary set. In [43], 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.

III. Method

Our method 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 a CNN-based orientation regressor to detect the 3D orientation of the robot to in turn decode the codeword.

A. 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*. This set of codebits or alphabet corresponds to the set of pose configurations the robot can perform.

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:

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

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. We can then define the cost of a code as the expected transmission cost of a codeword:

$$C(W) = \sum_{i \leq n} T(cw_i) \cdot p_i \tag{2}$$

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

A list of definitions is included here for reference:

- **Codebit** β_i: a particular pose configuration.
- **Alphabet** Σ: a set of codebits.
- **Codeword** cw_i: an ordered list of codebits.
- **Code** W: a set of codewords.

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. I.

TABLE I

EXAMPLE LIST OF 9 CODEBITS GENERATED FROM BINNING OF THE YAW AND PITCH AXES WITH $\theta_p = \theta_y = 60^\circ$ 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:

$$T(\beta_i) = p(\beta_i) \cdot \bar{e}(\beta_i) \cdot d(\beta_i) \tag{3}$$

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.

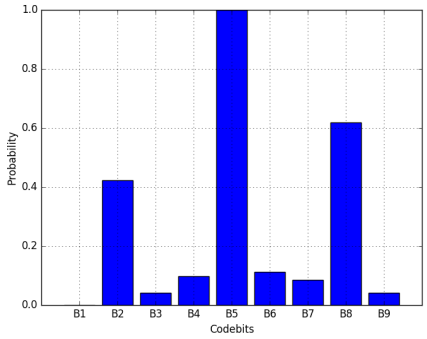

Fig. 2. A histogram showing the codebits probabilities captured from 5 minutes of footage of regular operation of the robot. We discourage the use of high probability codebits in message encoding to prevent the false detection of gestures during regular operation.

Given our list of codebits and their associated costs, we implement the optimal prefix-free dynamic programming algorithm presented by [43] to obtain the code-tree that minimizes the total cost of the prefix-free code. An example code tree is presented in Fig. 3.

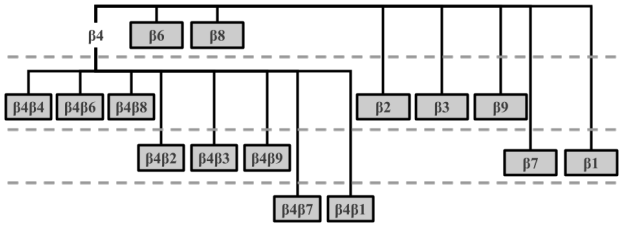

Fig. 3. An example optimal prefix-free code tree using the codebits from Tab. I with $r = 8$ codebits and $n = 15$ messages. The leaves of the tree (highlighted in gray) represent the final codewords that make up the code. The cost associated with each codebit is defined in the following list: $c = [1, 1, 1, 2, 2, 2, 3, 3]$ 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.

B. Visual Decoder

Using the codewords from the tree generated in Sec. III-A, 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. 1 shows Aqua executing the codeword $\{\beta_4\beta_z\beta_2\beta_z\}$

In order to get an orientation estimate, we rely on our prior work in [12] and train a (CNN) as an orientation regressor on monocular images of a model of the Aqua generated synthetically in Unreal. A description of our training and test datasets are presented in Sec. IV. This regressor is initialized with the VGG16 architecture [45] which has been shown to perform well on visual tasks. We use VGG weights pre-trained on ImageNet and augment the network for orientation regression of a quaternion. The network outputs the orientation estimate $q = (w, x, y, z)$ of the robot with respect to the observing camera. The loss function is defined as the L2-norm of the orientation quaternion:

$$L_q = \frac{1}{2n} \sum_x ||q_{gt} - q||^2 \quad (4)$$

Our network generalizes to real images without ever needing to train on them, foregoing the need for any human labelling of images. During tracking, we apply a Kalman-Filter tracker on the observer to integrate the orientation estimate.

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 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. 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 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 $[-10°, 10°]$.

IV. Dataset

A. Pose Estimation

Our pose estimation dataset contains a mixture of synthetic images generated with a custom Aqua simulator [9] built on Unreal and real, manually annotated underwater images collected during field trials at McGill University's Bellairs Research Institute in Barbados. We opt to restrict the training data to exclusively rely on synthetic images and do not include any real images in our training process. As seen in [12], using only photo-realistic synthetic images for our training set reduces the need for tedious manual annotation while resulting in a trained network that can accurately generalize to real data.

1) Unreal synthetic dataset: The synthetic dataset used to train our pose estimator consists of $50,000$ images created in Unreal with 45 variations in lighting angles and intensities on 4 sets of backgrounds, including a simulated custom-designed underwater world, a simulated pool, a fixed plane populated with random textures, and a white background. The Unreal Engine is one of the most faithful photorealistic simulation engines available. The simulated robot is displayed with 2 different robot chassis materials (matte and shiny) as well as 2 attachment configurations, one with and one without a downward facing camera attachment.

The simulated environments used in this paper were identical to those used in [12], utilizing professionally made photo-scanned assets and textures to increase realism [46].

The Aqua CAD model within each image is first randomly placed within the camera frame between $[0.5m, 2.0m]$ away from the simulated camera, then given a random orientation within $[-85°, 85°]$ on the three rotation axes (roll, pitch, yaw). Samples from the synthetic training data are presented in Fig. 4.

Fig. 4. Image samples from synthetic training data generated using Unreal.

2) Real underwater dataset: Our test set is comprised of 1000 real images collected during underwater field trials off the West coast of Barbados. The images are captured using a following robot's on-board cameras and diver-held cameras at variable distances and angles, mainly looking at the back of the Aqua. Using a custom-built annotator, we manually annotated the 6-DoF pose of the robot in each of these images, which allows the user to mark keypoints on the robot assigned from the CAD model. The human annotator then iteratively fits a wireframe to the robot using its known dimensions.

B. Visual Encoding

To evaluate our visual encoder, we prepare a dataset containing the Aqua executing codewords in both real pool trials and in the Aqua simulator [9]. 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. 3.

1) 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. III 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. 5.

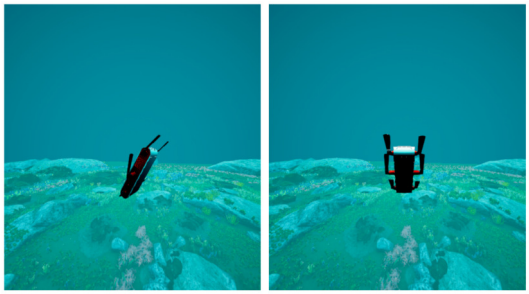

Fig. 5. Synthetic Aqua robot positioned according to codebit β_1 (left) and β_4 (right).

2) Real underwater pool gestures: The testing dataset used to evaluate visual encoding in a 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 Table IV.

In order to execute the gestures corresponding to codebits on the physical Aqua, a custom PID autopilot controller [6] 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.

3) Unreal trajectories: 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. IV-A.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.

V. Experimental Results

Fig. 6. Pair of Aqua robots following one another at sea and positioned for gesture-based communication.

A. Pose Estimation

Our orientation regression network is evaluated in isolation by evaluating mean angle errors on a real underwater test set. A table summarizing the angle errors from prior work in [12] is reproduced here in Tab. II in order to showcase the orientation regression performance.

TABLE II

Base metrics evaluated over the real underwater test set collected in Barbados as per Sec. IV-A.2.

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

B. 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. I. 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. 2 and a randomly sampled subset of angle errors relative to their respective angle value in Fig. 7.

The dynamic programming algorithm we implement from [43] 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 $[\beta_4, \beta_6, \beta_8, \beta_2, \beta_3, \beta_9, \beta_1, \beta_7]$ respectively. This cost list is an integer cost list which is reflective of the order of the

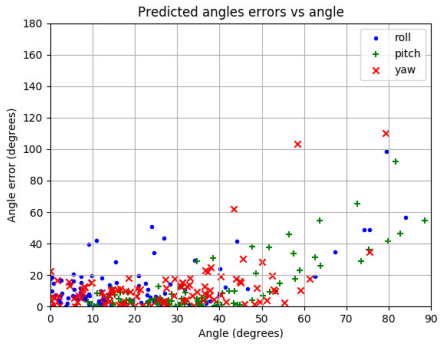

Fig. 7. Angle errors vs angle values for yaw/pitch/roll from a subset of the test set. This plot was first presented in [12].

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. 3.

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

TABLE III

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	

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 IV

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.

C. 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. IV-B.3, 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. V. Common false negatives are codebits β_2 and β_8 which represent basic left and right yaw configurations. As shown in Fig. 2, 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. III-A 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 V

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**

VI. CONCLUSION

We have presented a method for vision-based communication between robots in radio-denied environments. The method allows a robot to encode sequences of pose configurations (codebits) to convey a message. Our method uses optimal variable-length prefix codes to encode these codebits while minimizing the likelihood of false positive detection. The following robot decodes the transmitted message by using a pose estimation CNN. We demonstrate our technique on synthetically generated tracking sequences with a mean precision and recall of 0.96 and 0.91 respectively, and on real data with 0.94 and 0.66. The system runs in real-time on the Aqua robot underwater. We expect to use this technique in our own work of underwater multi-robot convoying using the Aqua robots to signal important messages and achieve more robust tracking.

REFERENCES

[1] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sept 1952.

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

[3] E. Games, "Unreal engine," https://www.unrealengine.com/en-US/what-is-unreal-engine-4, 1998-2018.

[4] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "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*. IEEE, 2017, pp. 23–30.

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

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

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

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

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

[10] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3038–3046.

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

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

[13] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," *CoRR*, vol. abs/1502.06796, 2015. [Online]. Available: http://arxiv.org/abs/1502.06796

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

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

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

[17] W. Ren and R. W. Beard, "Trajectory tracking for unmanned air vehicles with velocity and heading rate constraints," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 5, pp. 706–716, 2004.

[18] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[19] M. J. Tribou, A. Harmat, D. W. Wang, I. Sharf, and S. L. Waslander, "Multi-camera parallel tracking and mapping with non-overlapping fields of view," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1480–1500, 2015.

[20] G. Dudek and M. Jenkin, *Computational principles of mobile robotics*. Cambridge university press, 2010.

[21] H. Schneiderman, M. Nashman, A. J. Wavering, and R. Lumia, "Vision-based robotic convoy driving," *Machine Vision and Applications*, vol. 8, no. 6, pp. 359–364, 1995.

[22] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "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*, 2015, pp. 2686–2694.

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

[24] B. Sun and K. Saenko, "From virtual to reality: Fast adaptation of virtual object detectors to real domains." in *BMVC*, vol. 1, no. 2, 2014, p. 3.

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

[26] D. Feil-Seifer and M. J. Mataric, "Defining socially assistive robotics," in *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*. IEEE, 2005, pp. 465–468.

[27] D. Kortenkamp, E. Huber, R. P. Bonasso *et al.*, "Recognizing and interpreting gestures on a mobile robot," in *Proceedings of the National Conference on Artificial Intelligence*, 1996, pp. 915–921.

[28] T. Nakata, T. Sato, T. Mori, and H. Mizoguchi, "Expression of emotion and intention by robot body movement," in *Proceedings of the 5th international conference on autonomous systems*, 1998.

[29] I. Rekleitis, G. Dudek, and E. Milios, "Probabilistic cooperative localization and mapping in practice," in *IEEE International Conference on Robotics and Automation*. Taipei, Taiwan: IEEE, Sept. 2003, pp. 1907–1912.

[30] S. I. Roumeliotis and I. M. Rekleitis, "Propagation of uncertainty in cooperative multirobot localization: Analysis and experimental results," *Autonomous Robots*, vol. 17, no. 1, pp. 41–54, 2004.

[31] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "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*, vol. 2, Aug 1995, pp. 268–273 vol.2.

[32] D. Sutantyo and P. Levi, "Decentralized underwater multi-robot communication using bio-inspired approaches," *Artificial Life and Robotics*, vol. 20, no. 2, pp. 152–158, Jul 2015. [Online]. Available: https://doi.org/10.1007/s10015-015-0201-5

[33] W. Wang, J. Liu, G. Xie, L. Wen, and J. Zhang, "A bio-inspired electrocommunication system for small underwater robots," *Bioinspiration and Biomimetics*, vol. 12, no. 3, p. 036002, 2017. [Online]. Available: http://stacks.iop.org/1748-3190/12/i=3/a=036002

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

[35] 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*, April 2007, pp. 2507–2513.

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

[37] A. Jones and S. Andersson, "A motion-based communication system," in *2013 American Control Conference*, June 2013, pp. 365–370.

[38] M. V. Srinivasan, "Honeybee communication: A signal for danger," *Current Biology*, vol. 20, no. 8, pp. R366 – R368, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S096098221000240X

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

[40] D. Raghunathan and J. Baillieul, "Relative motion of robots as a means for signaling," vol. 2173, 10 2008.

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

[42] R. Karp, "Minimum-redundancy coding for the discrete noiseless channel," *IRE Transactions on Information Theory*, vol. 7, no. 1, pp. 27–38, 1961.

[43] M. J. Golin and G. Rote, "A dynamic programming algorithm for constructing optimal prefix-free codes with unequal letter costs," *IEEE*

Transactions on Information Theory, vol. 44, no. 5, pp. 1770–1781, Sept 1998.

[44] P. Bradford, M. J. Golin, L. L. Larmore, and W. Rytter, "Optimal prefix-free codes for unequal letter costs: Dynamic programming with the monge property," *Journal of Algorithms*, vol. 42, no. 2, pp. 277 – 303, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0196677402912137

[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[46] Quixel, "Quixel megascans," https://megascans.se/, 2018.