From Interaction to Integration: Advancing Optimal Human-Robot Interfaces for Underwater Manipulation

Paulo Padrao, Jose Fuentes, Leonardo Bobadilla
Knight Foundation School of Computing
and Information Sciences
Florida International University
Miami, FL 33199
Email: ppadraol@fiu.edu

Tero Kaarlela
Centria University
of Applied Sciences
Ylivieska, Finland
Email: tero.kaarlela@centria.fi

Alfredo Bayuelo
National University of Colombia
Bogotá, Colombia
Email: ajbayuelos@unal.edu.co

***Abstract*—Underwater operations involve interaction with the environment, but such interactions can be hazardous or detrimental to the underwater ecosystem. Therefore, it is essential to develop teleoperation systems that prioritize user comfort while minimizing environmental impact. In this study, we build upon previous research to enhance teleoperation systems specifically designed for underwater manipulation. We introduce robot components that enable interaction with the environment, particularly focusing on the robot's gripper. Through the utilization of hand movements, which are translated using Convolutional Neural Networks, we demonstrate the ability to achieve delicate and precise control over the robot's gripper motion. To validate our approach, we provide simulations that accurately represent the behavior of the real hardware involved in the teleoperation system.**

I. Introduction

Underwater robots are becoming increasingly essential tools for studying, monitoring, and managing coastal and estuaries environments. These environments are critical for numerous applications, including coastal conservation, coral restoration, and oil rig maintenance.

Moreover, emerging applications include a large degree of freedom manipulators [17, 2] and underwater humanoids [5]. These platforms motivate the need for more intuitive Human-Robot Interfaces that can capture natural motions that a remote operator can do to transmit these motions to a remote underwater complex robot.

In this short paper, we will extend our work Padrao et al. [15] where we proposed and showed that intuitive teleoperation interfaces are also optimal. The contributions of this paper are the following:

- Extended our problem formulation to include the robot's equipment that can be utilized to interact with the environment. In this case, we consider the robot's gripper;
- We propose an extension that decouples the gripper from the rest of the robot, as we could require soft movements that can only be performed by using the user's hands;
- Simulation results are provided for evaluation.

II. Related Work

The physical qualities of the water as a communication media prevents using Radio Frequency (RF) waves as a wireless communication method of teleoperation. Instead, acoustic and optical communications have been researched and utilized [16]. Both enable wireless underwater communication with either low bandwidth or limited usability. Acoustic modems are limited in bandwidth, allowing only low-resolution and low-quality video transmission [1]. While optical modems enable higher bandwidth than acoustic modems, the trade-off is the requirement for line-of-sight between the transmitter and the receiver. Maintaining line-of-sight between the ROV and the above-surface transmitter requires sophisticated tracking electronics and actuators [10]. In any of these modalities, the transmission of information can be limited and must be carefully measured.

Underwater teleoperation relies on robotic platforms such as ROVs and autonomous underwater vehicles (AUVs). These platforms are designed to be robust, durable, and capable of operating effectively in harsh and challenging conditions. This paper uses BlueROV2 [3], an open-source ROV platform provided in a six-thruster vectored configuration. BlueROV2 has been utilized in a wide variety of applications such as inspection [18], autonomous navigation [4], and education [8] of underwater robotics. QGroundControl (QGC) is the default open-source remote control and mission planning software for BlueROV2 [6]. BlueROV2 and QGC were utilized in the presented solution because both are open solutions, enabling hardware and software customization.

Different simulators are available for the BlueROV2 hardware, such as Orca4 [12] and BlueSim [7]. Orca4, a set of ROS2 packages, provides AUV support for BlueROV2. The Ardupilot ROS2 module offers low-level support for thruster control, and the Gazebo simulator provides a high-level interface for monitoring and controlling the BlueROV2. To create a three-dimensional pose of the ROV, Orca4 uses the real-time SLAM library ORB SLAM2 [14].Orca4 [12] is

a set of ROS2 packages to provide autonomous underwater vehicle support for the BlueROV2. The Ardupilot ROS2 module provides low-level support to control thrusters, and the Gazebo simulator provides a high-level interface to monitor and control the BlueROV2. Orca4 utilizes real-time SLAM library ORB SLAM2 [14] to create a three-dimensional pose of the ROV.

In this paper, we use the BlueSim simulator provided by Bluerobotics Inc. BlueSim is implemented with the open-source game engine Godot [11] and enables users to connect and control a simulated BlueROV2 utilizing QGC. In addition, BlueSim allows the user to visualize the environment by providing two virtual camera views of the simulated underwater environment. Together with the BlueSim, we use Google Mediapipe framework [13] which includes the Convolution Neural Network (CNN) implementations allowing robots to recognize objects, facial expressions, and hand gestures.

III. Methods

In this work, we consider an extension of our work considered in Padrao et al. [15]; this work considers the task of visual-based teleoperation of an underwater vehicle. In this scenario, two agents are involved, the person teleoperating the robot and the robot itself. Both have a workspace, an action space, and a state space. For easiness, let us remind the necessary notation. Consider $\mathcal{W}_o \subset \mathbb{R}^3$ and $\mathcal{W}_r \subset \mathbb{R}^3$ to be the workspaces for the operator and the robot, respectively; $\mathcal{C}_o$ and $\mathcal{C}_r$ be their configuration spaces. We denote by $\mathcal{U}$ the set of controls applicable to the robot and $\mathcal{A}$ the set of actions the human operator can perform. The robot's dynamics is ruled by the relation given by the function $f : \mathcal{C}_r \times \mathcal{U} \longrightarrow \mathcal{C}_r$

$$\dot{x} = f(x, u). \tag{1}$$

Since a person is making use of the teleoperation system, it is required to map the user's actions to the robot's actions through a map $g : \mathcal{C}_r \times \mathcal{A} \longrightarrow \mathcal{U}$ so that the robot is affected by actions taken by the user.

$$\dot{x} = f(x, g(x, a)). \tag{2}$$

So g is intended to translate the user's commands into the robot's commands. For multiple reasons, this function g should comply with several principles to capture stability and comfort. Also, they aim to preserve part of the configuration space structure such as *Continuity*, *Linearity*, *Consistency* and *Reachability* described in detail in Hauser [9], Padrao et al. [15]. Define a_θ and a_ψ to be the head pitch and head yaw commands, respectively, and u_θ and u_ψ to be the robot camera tilt command and yaw command of the robot base. Also, let v_o and v_r be the linear forward velocities of the operator and the robot in $[m/s]$, respectively. We consider that the depth of the operator and ROV can be set directly by the action variables a_z and u_z in $[m]$, respectively. Also, we include the hand actions that are intended to manage the robot's gripper. The user's actions a_{close} and a_{open} and the robot's actions u_{close} and u_{open} indicate how opened or closed the user's hands and the robot's gripper respectively. Then, we define the action space of the operator as

$$\begin{aligned}\mathcal{A} = (a_{\theta\,\min}, a_{\theta\,\max}) \times (a_{\psi\,\min}, a_{\psi\,\max}) \times (v_{o\,\min}, v_{o\,\max}) \\ \times (a_{z\,\min}, a_{z\,\max}) \times (a_{close}, a_{open})\end{aligned} \tag{3}$$

and the action space of the robot as

$$\begin{aligned}\mathcal{U} = (u_{\theta\,\min}, u_{\theta\,\max}) \times (u_{\psi\,\min}, u_{\psi\,\max}) \times (v_{r\,\min}, v_{r\,\max}) \\ \times (u_{z\,\min}, u_{z\,\max}) \times (u_{close}, u_{open}).\end{aligned} \tag{4}$$

Finding an appropriate function, denoted as g, to translate user commands into the robot's actions is a crucial aspect of developing an effective teleoperation system. In this context, we examine a general formulation where the suitability of function g is determined by minimizing a customized functional, denoted as $J(g)$:

$$J(g) = \int_0^T L(x, g, Dg), dt, \tag{5}$$

Here, L represents a cost function, and D^n denotes the n^{th} derivative of g. The time interval $[0, T]$ signifies the duration during which the teleoperation task is executed. It is worth noting that this approach is not limited to the presented formulation. It is possible to incorporate task-related constraints, incorporate non-integral terms, and explore more comprehensive versions of (5). For instance, in our recent work Padrao et al. [15], we consider a functional, denoted as (6), which addresses the challenge of moving the robot from an initial state $x_{initial}$ to a desired state $x_{final} = x(T)$ while the robot adheres to the dynamics specified in (2). In this context, the function g is considered to be a linear transformation represented by a matrix G, such that the robot's command $u(t)$ and the user's action $a(t)$ are related through $u(t) = g(a(t)) = Ga(t)$. This representation offers several advantages as it satisfies principles such as linearity and continuity directly by adopting this structure for the function g.

Specifically, the functional seeks to find the optimal map G and the user's optimal actions $a(t)$ by minimizing the following expression:

$$\begin{aligned}\min_{g,a} \quad & \alpha||x_{final} - x(T)||^2 + \beta \int_0^T a(t)^\top M a(t) dt \\ & + \gamma \int_0^T ||\dot{x}(t)|| dt + \delta\, dist(G, O(2)). \\ \text{S.t.} \quad & \dot{x}(t) = \begin{bmatrix} \cos(\theta(t)) & 0 \\ \sin(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix} u(t), \quad x(0) = x_{initial}.\end{aligned} \tag{6}$$

In this formulation, M is a positive-definite matrix, $O(2)$ represents the set of orthogonal 2×2 matrices, x_{final} denotes the target point, and the coefficients α, β, γ, and δ are non-negative regularization coefficients that determine the relative importance of each term.

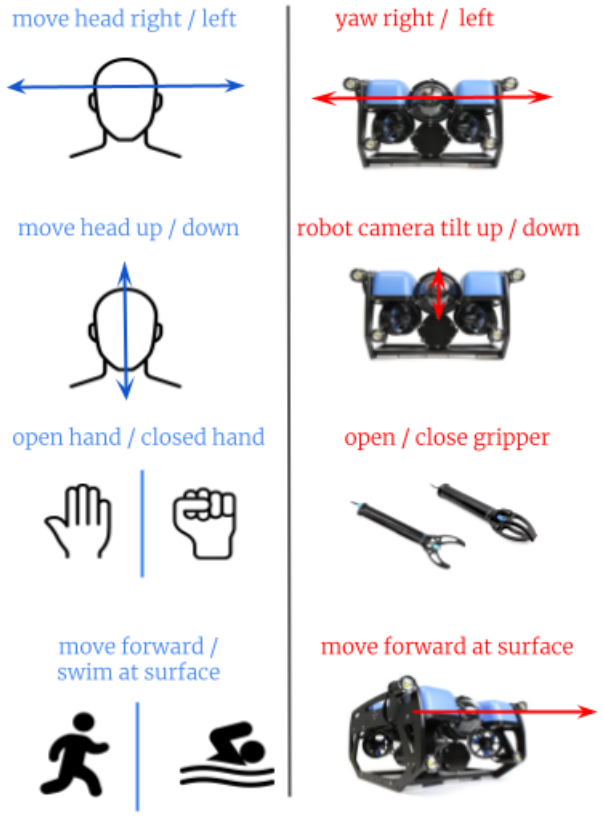

Fig. 1. An example of human-robot action space translation

An illustrative example of the translation between human and robot actions is depicted in Fig.1. This mapping is considered intuitive and natural, as many systems employ similar approaches to handle comparable scenarios. Our previous work Padrao et al. [15] demonstrated that this assignment, which involves body movements but excludes hand movements, not only possesses these desirable attributes but also satisfies the optimality conditions described in (6). Building upon this result, our current work extends the approach to include hand movements for operating the robot's integrated gripper, thereby enabling user interaction with the environment.

We have separated the task of managing the gripper from the task of driving the robot, as we believe that the former necessitates delicate movements that cannot be adequately achieved solely through the body movements depicted in Fig.1. In this particular case, since we have two user hand movements, namely opening and closing the hand, mapped to two robot actions, namely opening and closing the gripper, it is evident that in order to maintain the properties that ensure comfortable and intuitive usage, the mapping described in Fig.1 satisfactorily fulfills these requirements.

IV. Results

This work employed a Software-in-the-Loop (SIL) approach. Instead of using actual hardware, the SIL configuration utilized the BlueSim hardware simulator [7]. For the development and configuration of the software components, the SIL configuration incorporated the BlueSim hardware simulator as a substitute for real hardware. The BlueSim simulator emulates the BlueROV2 hardware [3] and includes a virtual camera unit, which allows for testing and refining the system. Once fully operational, the proposed architecture is illustrated in Figure 2.

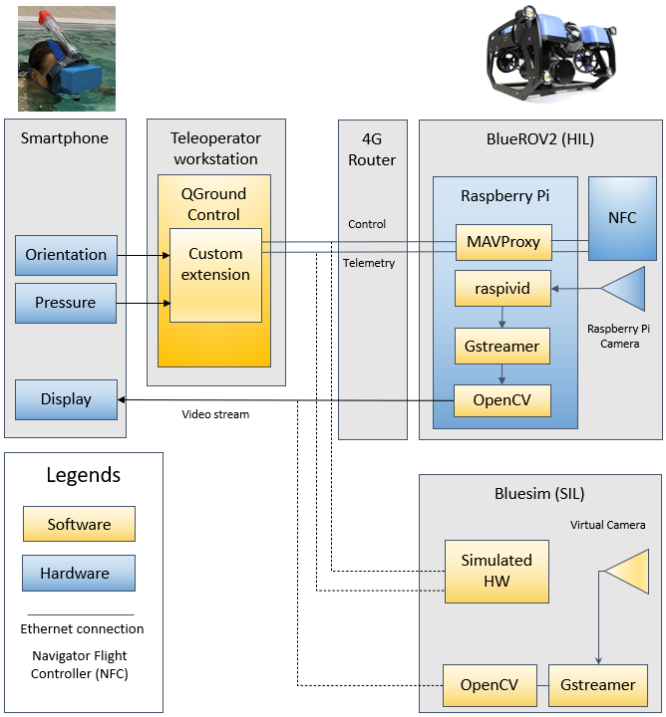

Fig. 2. Proposed architecture for human-robot interface [15]

By integrating the Google Mediapipe framework [13] with our system, we established a connection between recognized hand gestures and the BlueSim gripper. This integration allowed us to effortlessly translate the identified gestures into precise commands that directed the actions of the BlueSim gripper, resulting in an intuitive user experience.

For gesture recognition, our architecture consisted of a 6-layer sequential model. The input layer is a one-dimensional array with a length of 21 multiplied by 2. That covers all hand landmarks as shown in Figure 3. The second layer is a 20%-dropout layer used for regularization and to prevent overfitting. The third layer is a fully connected layer with 20 units and a ReLU activation function. Then, another dropout layer follows with a dropout rate of 40%, and the subsequent dense layer has ten units and also uses the ReLU activation function. Finally, the output layer has the number of classes as its units (in this case 2), and it uses the softmax activation function. This layer produces output probabilities for each class, representing the likelihood of the input belonging to each class. In total, our model has 1,092 parameters. For model compilation, our loss function consisted of the sparse categorical cross-entropy, and we used a stochastic gradient descent method based on the Adam optimization. We trained our model for 1,000 epochs

with a batch size of 128 and obtained an accuracy of 0.981. Hand gesture detection results are shown in Figure 4, and the integration with BlueSim can be found at https://youtu.be/EXK2kUNe4rA.

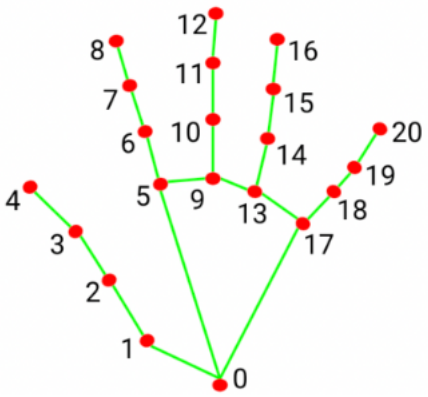

Fig. 3. Hand landmarks [13]

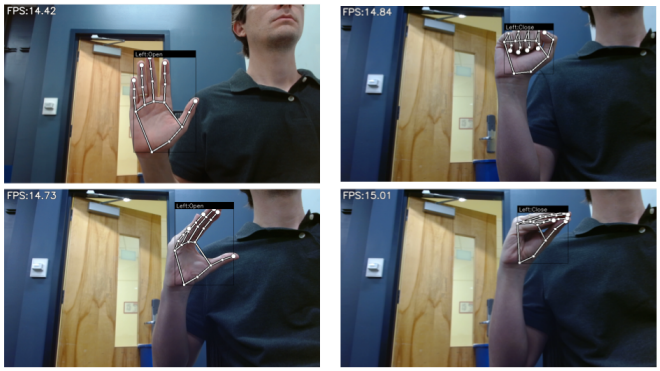

Fig. 4. Hand gesture detection results. The first column illustrates two distinct configurations for detecting open hands while the second column describes potential configurations associated with closed hands

V. Conclusion

This paper continues our work aiming to control an ROV using natural body movements. Previously we proposed a novel concept for controlling ROV directional movements by capturing and translating teleoperator head motions into control commands of the ROV; now, the previous work is extended by enabling the teleoperator to control ROV gripper states using hand gestures. The proposed system utilizes Google Media Pipeline to identify the open and closed states of the teleoperator hand, and a Python script translates recognized states into corresponding control commands of the gripper.

The proposed solution enables to control of an ROV gripper using hand gestures. The system proposed is scalable and, in addition to controlling a robot gripper, can be utilized to control other systems requiring binary control. In the future Hardware-in-the-loop (HIL) should be enabled, to control physical ROV and gripper in addition to simulated ones. Furthermore, haptics for the grasping would provide a realistic teleoperation experience for the teleoperator and allow for performing complex tasks such as pick-and-place of underwater objects.

Acknowledgments

References

[1] Alexander Barbie, Niklas Pech, Wilhelm Hasselbring, Sascha Flogel, Frank Wenzhoefer, Michael Walter, Elena Shchekinova, Marc Busse, Matthias Turk, Michael Hofbauer, and Stefan Sommer. Developing an underwater network of ocean observation systems with digital twin prototypes—a field report from the baltic sea. *IEEE Internet Computing*, PP:1–1, 03 2021. doi: 10.1109/MIC.2021.3065245.

[2] Andreas Birk, Tobias Doernbach, Christian Mueller, Tomasz Łuczynski, Arturo Gomez Chavez, Daniel Koehntopp, Andras Kupcsik, Sylvain Calinon, Ajay K Tanwani, Gianluca Antonelli, et al. Dexterous underwater manipulation from onshore locations: Streamlining efficiencies for remotely operated underwater vehicles. *IEEE Robotics & Automation Magazine*, 25(4):24–33, 2018.

[3] Bluerobotics Inc. Bluerov2. https://bluerobotics.com/store/rov/bluerov2/, 2023. Accessed 2 February 2023.

[4] Juan Chen, Caiming Sun, and Aidong Zhang. Autonomous navigation for adaptive unmanned underwater vehicles using fiducial markers. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9298–9304, 2021. doi: 10.1109/ICRA48506.2021.9561419.

[5] Changhyun Chung and Motomu Nakashima. Development of a swimming humanoid robot for research of human swimming. *Journal of Aero Aqua Bio-mechanisms*, 3(1):109–117, 2013.

[6] Dronecode Project, Inc. Qgroundcontrol. http://qgroundcontrol.com/, 2023. Accessed 2 February 2023.

[7] William Galvani and Patrick Pereira. Bluesim. https://github.com/bluerobotics/bluesim, 2023. Accessed 23 February 2023.

[8] Ashiria Goel, Colin Szeto, Mabel Szeto, Rishi Veerepalli, and Eesh Vij. Leveraging competitive robotics experience to spread marine education. In *OCEANS 2021: San Diego – Porto*, pages 1–9, 2021. doi: 10.23919/OCEANS44145.2021.9705971.

[9] Kris Hauser. Recognition, prediction, and planning for assisted teleoperation of freeform tasks. *Autonomous Robots*, 35, 11 2013. doi: 10.1007/s10514-013-9350-3.

[10] Hemani Kaushal and Georges Kaddoum. Underwater optical wireless communication. *IEEE Access*, 4:1518–1547, 2016. doi: 10.1109/ACCESS.2016.2552538.

[11] Juan Linietsky and Ariel Manzur. Godot engine. https://github.com/godotengine/godot, 2022. Accessed 27 February 2023.

[12] Juan Linietsky and Ariel Manzur. Orca4. https://github.com/clydemcqueen/orca4, 2022. Accessed 12 November 2022.

[13] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for perceiving and processing reality. In *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019*, 2019. URL https://mixedreality.cs.cornell.edu/s/NewTitle_May1_MediaPipe_CVPR_CV4ARVR_Workshop_2019.pdf.

[14] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. doi: 10.1109/TRO.2015.2463671.

[15] Paulo Padrao, Jose Fuentes, Tero Kaarlela, Alfredo Bayuelo, and Leonardo Bobadilla. Towards optimal human-robot interface design applied to underwater robotics teleoperation, 2023.

[16] V. Ranadive and T. Sheridan. Video framerate, resolution and grayscale tradeoffs for undersea telemanipulator control. In *OCEANS 81*, pages 1222–1222, 1981. doi: 10.1109/OCEANS.1981.1151520.

[17] Satja Sivčev, Joseph Coleman, Edin Omerdić, Gerard Dooly, and Daniel Toal. Underwater manipulators: A review. *Ocean engineering*, 163:431–450, 2018.

[18] Anne Wendt, Henrich Preuss, Wito Kleinhempel, and Helge Renkewitz. Frankenstein goes swimming: Software architecture of a modified bluerov2 heavy for underwater inspection and maintenance. In *OCEANS 2022, Hampton Roads*, pages 1–5, 2022. doi: 10.1109/OCEANS47191.2022.9977166.