Journal of *Marine Science and Engineering*

MDPI

*Article*

# Underwater Object Recovery Using a Hybrid-Controlled ROV with Deep Learning-Based Perception

**Inés Pérez-Edo, Salvador López-Barajas, Raúl Marín-Prades and Pedro J. Sanz ***

Interactive Robotic Systems Lab, Jaume I University, 12071 Castellón de la Plana, Spain; al416992@uji.es (I.P.-E.); barajas@uji.es (S.L.-B.); rmarin@uji.es (R.M.-P.)
* Correspondence: sanzp@uji.es

**Abstract**

The deployment of large remotely operated vehicles (ROVs) or autonomous underwater vehicles (AUVs) typically requires support vessels, crane systems, and specialized personnel, resulting in increased logistical complexity and operational costs. In this context, lightweight and modular underwater robots have emerged as a cost-effective alternative, capable of reaching significant depths and performing tasks traditionally associated with larger platforms. This article presents a system architecture for recovering a known object using a hybrid-controlled ROV, integrating autonomous perception, high-level interaction, and low-level control. The proposed architecture includes a perception module that estimates the object pose using a Perspective-n-Point (PnP) algorithm, combining object segmentation from a YOLOv11-seg network with 2D keypoints obtained from a YOLOv11-pose model. In addition, a Natural Language ROS Agent is incorporated to enable high-level command interaction between the operator and the robot. These modules interact with low-level controllers that regulate the vehicle degrees of freedom and with autonomous behaviors such as target approach and grasping. The proposed system is evaluated through simulation and experimental tank trials, including object recovery experiments conducted in a 12 $\times$ 8 $\times$ 5 m test tank at CIRTESU, as well as perception validation in simulated, tank, and harbor scenarios. The results demonstrate successful recovery of a black box using a BlueROV2 platform, showing that architectures of this type can effectively support operators in underwater intervention tasks, reducing operational risk, deployment complexity, and mission costs.

**Keywords:** hybrid-controlled ROV; natural language agent; search and rescue; underwater manipulation



Academic Editor: Weicheng Cui

Received: 19 December 2025
Revised: 12 January 2026
Accepted: 16 January 2026
Published: 18 January 2026



## 1. Introduction

Underwater operations are of relevance in Search and Recovery (S&R) missions, aimed at locating and retrieving objects of interest such as flight recorders [1], submerged archaeological material [2], or biological samples [3]. Thanks to these interventions, it is possible to obtain objects or materials of high value that contribute to cultural preservation, research, and safety.

In this context, remotely operated vehicles (ROVs) began to be used, platforms equipped with cameras and sonars, connected to the surface through an umbilical cable that provides real-time power and data transmission. This solution makes it possible to address the growing number of underwater operations and to access areas where diver

intervention, although effective, involves risks and limitations in immersion time [4]. Although ROVs have proven to be effective tools, their configuration is designed for real-time teleoperation, creating a dependence on both the support vessel and the operator. For this reason, autonomous underwater vehicles (AUVs) emerged, capable of navigating and inspecting large areas without the need for human teleoperation [5]. The most recent step in this evolution is the development of autonomous intervention vehicles (I-AUVs), which have expanded the capabilities of underwater robotics by combining advanced onboard decision making with one or more manipulators, enabling the execution of complex underwater tasks without continuous human supervision.

Over the last decades, autonomous underwater manipulation has gained significant relevance in research. A clear indication of this growing interest is the increasing number of survey papers published in this field. For instance, a comprehensive review of underwater manipulators is presented in [6], where different systems are analyzed according to their actuation type (electric or hydraulic), application domain (commercial or research), and mechanical design and control strategies. Another relevant survey can be found in [7], which focuses on complete intervention autonomous underwater vehicle (I-AUV) systems and discusses the main challenges associated with underwater manipulation tasks, using the Girona 500 I-AUV equipped with the ECA arm as a representative example. More specifically, manipulation at the end-effector level has been addressed in [8], where a detailed review of underwater grippers for specimen collection compares different design solutions and operating principles, while a complementary perspective is provided in [9], where a taxonomy of underwater manipulation actions is proposed and analyzed, contributing to a more structured understanding of manipulation capabilities in subsea environments; these trends toward increased system integration, autonomy, and reduced reliance on physical tethers are exemplified by recent experimental platforms such as the OPTIHROV project, which has demonstrated cableless underwater manipulation using both single-arm [10] and dual-arm [11] configurations based on visible light and acoustic wireless communications, while maintaining the human operator in the control loop.

The use of large ROVs or AUVs typically requires support vessels, crane systems, and specialized personnel, increasing logistics and complicating their deployment. In contrast, there are currently lightweight robots capable of reaching great depths and performing tasks similar to those of large ROVs. Currently, market-ready ROVs are being used as low-cost AUVs due to their modularity and the possibility of equipping them with autonomous navigation capabilities [12]. In this context, several works have demonstrated the use of lightweight ROVs for the development of autonomous underwater systems. Some representative examples include the BlueROV2 [13], which has been used for autonomous coordination in underwater pipeline assembly [14] and for autonomous navigation tasks in the inspection of aquaculture cages [15]. Similarly, the Blueye Pioneer has also been employed in applications such as large-scale inspection of offshore mooring systems using low-cost autonomous underwater drones [16].

For this reason, lightweight underwater robots require perception and manipulation strategies that are effective while remaining simple and low-cost. In recent works, many underwater interventions use ArUco markers for pose estimation and guidance for manipulation [14]. In a monocular camera has also been used without explicitly estimating the target's 6D pose, instead detecting markers or pixel-segmented elements [16,17]. However, in the literature it is not easy to find cases where a monocular camera is used to estimate the pose of an underwater object. For this task depth cameras and multibeam and multisonar sensors can also be used. Alternatively, other works have explored the use of depth cameras, multibeam systems, or multisonar sensors, which provide additional geometric information, but usually involve higher cost, complexity, and energy consumption.

At a higher level of system operation, Large Language Models (LLMs) have begun to be used to interact with robotic systems at a high level. These models allow the operator to express tasks in natural language, and the LLM acts as an agent capable of interpreting the user's intent and generating the instructions to carry it out. Recent examples such as AquaChat [18], oriented toward the inspection of aquaculture infrastructures, and OceanChat [19], focused on the high-level piloting of underwater vehicles, demonstrate how LLMs can function as a mechanism that translates complex commands into concrete operational actions. Through APIs that enable the exchange of natural-language messages, these agents facilitate more intuitive modes of operation and reduce the need to program or specify low-level commands.

The main contribution of this work relies on the recovery of a known object using a lightweight hybrid-controlled ROV equipped with a gripper rigidly fixed to the body frame. To address this problem, several modules were integrated. These include a perception module capable of estimating the 3D pose of a known object using a monocular camera, a natural-language ROS agent that allows the operator to send high-level commands to the robot, and low-level controllers responsible for executing the approach and grasping maneuvers. This paper is organized as follows. Section 2 presents the kinematics of the hybrid-controlled ROV, the control architecture, the perception module, and the natural-language ROS agent. Section 3 presents the mechatronics integration of the hybrid-controlled ROV, the simulation environment, and the case study considered in this work. Section 4 reports the performance of the perception module, the natural-language ROS agent, and the approach and grasping controllers. Finally, conclusions and future work are provided in Section 5.

## 2. Methodology

### *2.1. Kinematics*

The kinematics of the Autonomous Underwater Vehicle are presented in this subsection. It starts by introducing the reference frames and definitions, and then proceeds to describe the kinematics of position and velocity.

#### 2.1.1. Reference Frames and Definitions

Figure 1 shows the frames of the hybrid-controlled ROV and Table 1 describe the frames.

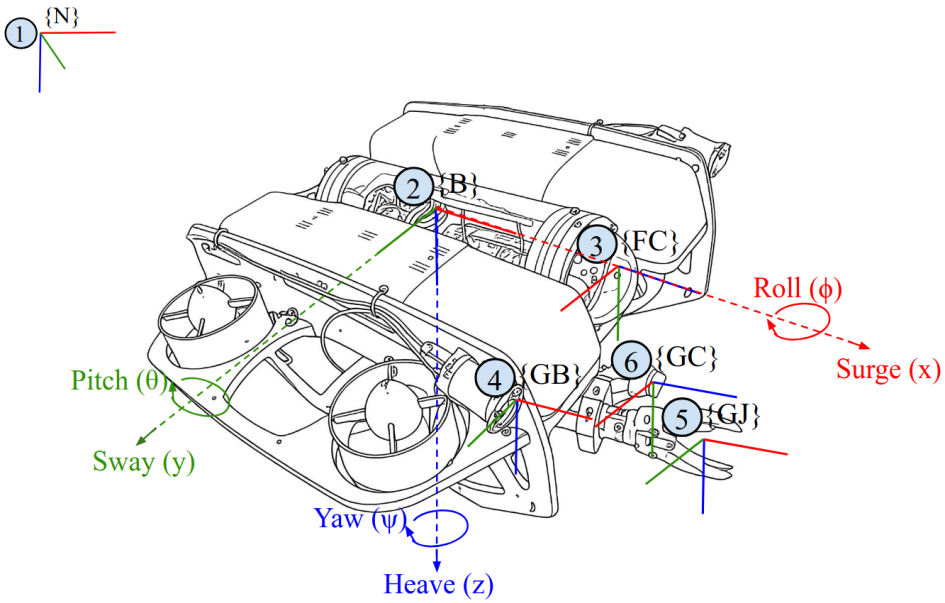


**Figure 1.** Kinematic frames of the hybrid-controlled ROV.

**Table 1.** Reference frames used in the hybrid-controlled ROV.

| # | Frame Name | Description |
|---|---|---|
| 1 | NED | North-East-Down reference frame |
| 2 | ROV_base_link | Main body frame of the ROV |
| 3 | Front_camera | Front-facing camera on the ROV |
| 4 | Gripper_base_link | Base frame of the Newton Gripper mounted on the ROV |
| 5 | Gripper_jaws_link | Jaws frame of the Newton Gripper |
| 6 | Gripper_camera | Camera mounted at the Gripper |

The motion of the robot is described using six degrees of freedom (DoF), comprising three translational and three rotational components. The translational motions correspond to surge, sway, and heave, which represent displacements along the body-fixed $X$, $Y$, and $Z$ axes and are denoted by the position variables $x$, $y$, and $z$, with corresponding linear velocities $u$, $v$, and $w$. The rotational motions include roll, pitch, and yaw, defined as rotations about the $X$, $Y$, and $Z$ axes and represented by the orientation angles $\phi$, $\theta$, and $\psi$, with associated angular velocities $p$, $q$, and $r$, respectively. It is important to notice that although surge, sway and heave are defined in the body frame, the position variables x, y, z are expressed in the NED frame.

Using this notation the vector pose of the robot is given by

$$\boldsymbol{\eta} = \begin{bmatrix} \eta_1^T & \eta_2^T \end{bmatrix}^T = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T. \tag{1}$$

2.1.2. Kinematics of Position

The position and orientation of the ROV's base link (Frame B) relative to the North-East-Down reference frame (Frame N) are described through the vector $\boldsymbol{\eta}$ introduced in Equation (1). Using this vector, the corresponding homogeneous transformation matrix is formulated, as shown in Equation (2).

$${}^{N}\mathbf{T}_{B}(\boldsymbol{\eta}) = \begin{bmatrix} {}^{N}\mathbf{R}_{B}(\eta_2) & \eta_1 \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2}$$

where the orientation of Frame $\{B\}$ relative to Frame $\{N\}$ is parameterized by the Euler angles, from which the corresponding rotation matrix is obtained as

$${}^{N}\mathbf{R}_{B}(\boldsymbol{\eta_2}) = \mathbf{R}_z(\psi)\,\mathbf{R}_y(\theta)\,\mathbf{R}_x(\phi) \tag{3}$$

Considering the sequence of rotations around the $x$, $y$, and $z$ axes, the overall rotation matrix mapping Frame $\{B\}$ to Frame $\{N\}$ is obtained as a function of $\boldsymbol{\eta}_2$ as

$${}^{N}\mathbf{R}_{B}(\eta_2) = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{4}$$

The spatial configuration of the gripper jaws, defined by Frame $\{GJ\}$, relative to the vehicle base frame $\{B\}$ is characterized by the homogeneous transformation

$${}^{B}\mathbf{T}_{GJ} = \begin{bmatrix} \mathbf{R}_{3\times 3} & \mathbf{p}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{5}$$

in this equation **R** represents the rotation and **p** is the translation vector. Using both transforms (Frame {N} to {B} and Frame {B} to {GJ}) the gripper jaws frame relative to the NED coordinate frame can be derived as follows

$$^{N}\mathbf{T}_{GJ}(\boldsymbol{\eta}) = {}^{N}\mathbf{T}_{B}(\boldsymbol{\eta}) \cdot {}^{B}\mathbf{T}_{GJ} \tag{6}$$

with this transform the pose of the gripper jaws relative to the word can be represented as

$$\boldsymbol{\eta_{GJ}} = \begin{bmatrix} \eta_{GJ_1} & \eta_{GJ_2} \end{bmatrix}^T = \begin{bmatrix} x_{GJ} & y_{GJ} & z_{GJ} & \phi_{GJ} & \theta_{GJ} & \psi_{GJ} \end{bmatrix}^T \tag{7}$$

2.1.3. Kinematics of Velocity

The velocities of the gripper jaws frame relative to the world can be computed using the body velocities, for this the Jacobian of the vehicle $\mathbf{J_v}$ is used to relate the body velocities and the NED frames velocities as shown below

$$\dot{\boldsymbol{\eta}} = \mathbf{J_v}(\eta_2)\boldsymbol{\nu} \tag{8}$$

here, $\boldsymbol{\nu}$ is the body velocity vector, $\dot{\boldsymbol{\eta}}$ represent represents the time derivative of the pose expressed in the NED frame. {N} and $\mathbf{J_v}(\eta_2)$ represent the Jacobian of the vehicle that is described as

$$\mathbf{J_v}(\eta_2) = \begin{bmatrix} ^{N}\boldsymbol{R}_B(\eta_2) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & J_{\nu_2}(\eta_2) \end{bmatrix} \tag{9}$$

where $^{N}\boldsymbol{R}_B(\eta_2)$ was described in Equation (4) and $J_{\nu_2}(\eta_2)$ represents the angular transformation matrix from the body frame to the NED frame, this matrix can be expressed as

$$J_{\nu_2}(\eta_2) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \tag{10}$$

The gripper jaws frame (Frame {GJ}) is rigidly attached to the vehicle base link (Frame {B}) through the fixed transformation shown in Equation (5). Because the transform is constant, the angular velocity of the gripper frame is identical to that of the vehicle body. However, the linear velocity of the gripper origin differs due to the contribution generated by the angular motion of the ROV around the offset vector $^{B}r_{BGJ}$.

The linear velocity of the gripper jaws expressed in the body frame can be written using rigid-body velocity propagation as

$$^{B}\nu_{GJ} = {}^{B}\nu_B + {}^{B}\omega_B \times {}^{B}r_{BGJ} = {}^{B}\nu_B - S(^{B}r_{BGJ})\,{}^{B}\omega_B, \tag{11}$$

where $S(\cdot)$ denotes the skew-symmetric matrix operator and $\omega_B$ represent the angular body velocity.

Transforming the linear velocity to the NED frame yields

$$^{N}\nu_{GJ} = {}^{N}R_B(\eta_2)\left(\begin{bmatrix} I_{3\times3} & -S(^{B}r_{BGJ}) \end{bmatrix}\begin{bmatrix} ^{B}\nu_B \\ ^{B}\omega_B \end{bmatrix}\right). \tag{12}$$

Since the orientation of the gripper jaws frame changes according to the same angular velocity as the body, the Euler angle rates are obtained through

$$\dot{\eta}_{GJ_2} = J_{\nu_2}(\eta_2)\,{}^{B}\omega_B. \tag{13}$$

Combining the linear and angular components, the complete velocity vector of the gripper jaws frame in the NED frame can be expressed compactly as

$$\dot{\boldsymbol{\eta}}_{GJ} = \mathbf{J}_v^{GJ}(\eta_2)\,\boldsymbol{\nu}, \tag{14}$$

where the Jacobian of the gripper jaws frame is given by

$$\mathbf{J}_v^{GJ}(\eta_2) = \begin{bmatrix} {}^{N}R_B(\eta_2) & -{}^{N}R_B(\eta_2)\,S({}^{B}r_{BGJ}) \\ \mathbf{0}_{3\times3} & J_{\nu_2}(\eta_2) \end{bmatrix}. \tag{15}$$

Equation (15) shows that the Jacobian of the gripper jaws differs from the vehicle Jacobian only by the additional linear velocity term induced by the rotation of the body around the fixed offset ${}^{B}r_{BGJ}$. This term captures the kinematic contribution of the ROV's angular motion to the position of the end-effector.

### 2.2. Controllers

A PID controller was used to control the ROV in the different phases of the intervention. The controller was chosen due to its simplicity and its low dynamic response required for the intervention. Also, it ensures sufficient stability at the low speed ranges of the ROV. The control law is shown bellow:

$$\boldsymbol{u}(t) = K_p\,\boldsymbol{e}(t) + K_i \int_0^t \boldsymbol{e}(\tau)\,d\tau + K_d\dot{\boldsymbol{e}}(t) \tag{16}$$

where $\boldsymbol{u}(t) \in \mathbb{R}^6$ is the control input vector composed of linear and angular components, and $K_p$, $K_i$, and $K_d \in \mathbb{R}^{6\times6}$ are diagonal gain matrices for the proportional, integral, and derivative terms, respectively.

The tracking error is defined as

$$\boldsymbol{e}(t) = \begin{bmatrix} \boldsymbol{e}_p(t) \\ \boldsymbol{e}_\eta(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_p(t) - \boldsymbol{y}_p(t) \\ \boldsymbol{r}_\eta(t) - \boldsymbol{y}_\eta(t) \end{bmatrix} \tag{17}$$

where $\boldsymbol{e}_p(t) \in \mathbb{R}^3$ represents the position error and $\boldsymbol{e}_\eta(t) \in \mathbb{R}^3$ represents the orientation error. The vectors $\boldsymbol{r}_p(t) = [x_r\ y_r\ z_r]^T$ and $\boldsymbol{y}_p(t) = [x\ y\ z]^T$ denote the desired and measured positions in the NED frame, respectively, while $\boldsymbol{r}_\eta(t) = [\phi_r\ \theta_r\ \psi_r]^T$ and $\boldsymbol{y}_\eta(t) = [\phi\ \theta\ \psi]^T$ denote the desired and measured orientations.

Specifically for the grasping controller, the forward (surge) command is modulated according to the lateral, vertical, and yaw alignment. The effective surge command is given by

$$u_x^{cmd} = \alpha\,u_x, \qquad \alpha = 1 - \min\left(1, \max_{j\in\{y,z,\psi\}} \frac{|e_j|}{t_j}\right), \tag{18}$$

where $t_j$ are predefined tolerance bounds. This strategy allows a behavior in which the ROV first achieves lateral and angular alignment before advancing towards the target, improving stability and reducing the risk of undesired contacts during low-speed intervention maneuvers.

### 2.3. Perception

The perception module constitutes a fundamental component of the autonomous grasping system, as it is responsible for providing the visual information necessary to interpret the environment and locate the object of interest. In this work, only a monocular camera is used as the visual sensor, which implies working with purely projective informa-

tion and facing limitations in terms of depth and loss of structural information, especially in demanding environments such as the underwater medium. These restrictions affect the system differently depending on the stage of the mission and the content visible in the image. When the object appears fully within the field of view, it is possible to apply global detection techniques and obtain a coherent description of its structure. In contrast, in close-range stages, where visibility is often partial, obtaining the global context of the object becomes more difficult and therefore more localized processing is required, focused exclusively on the grasping area.

For this reason, the proposed perception system is organized into two differentiated phases according to the visual conditions and the information available in each stage of the approach.

#### 2.3.1. Global Object Perception: Object Detection, Keypoints and 6D Pose Estimation

In this phase, having the complete object within the field of view is crucial to estimate its pose. However, this situation also introduces the added difficulty that the image includes multiple environmental elements visible in the scene that act as visual noise. The overall global object perception workflow of this stage is illustrated in Figure 2. To address this stage, neural networks based on YOLO [20] were employed, which allow an initial detection of the known object and isolate it through a region of interest (ROI), thus eliminating most of the irrelevant information. On this ROI, a YOLO-pose model is applied, responsible for predicting labelled keypoints of recognized points of the object. These 2D correspondences subsequently allow the application of a Perspective-n-Point (PnP) [21] procedure to obtain a complete estimation of the 6D pose of the object with respect to the camera.

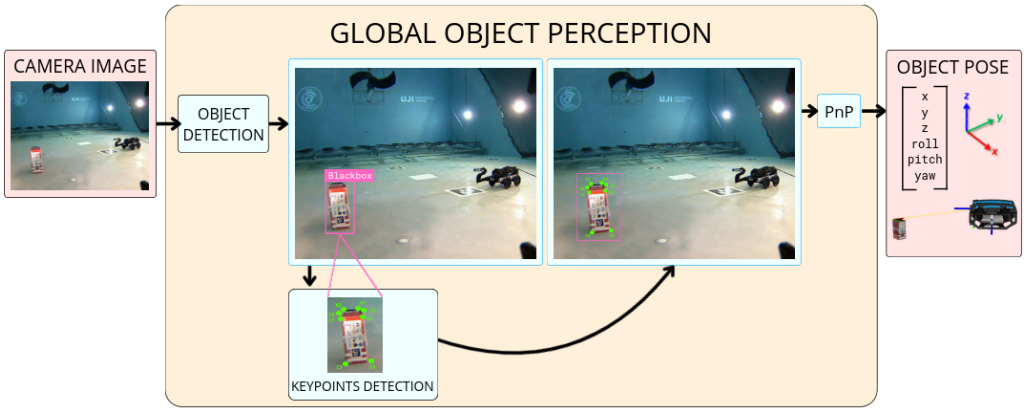


**Figure 2.** Overview of the global object perception module.

Object Detection and ROI Extraction

First, the detection of the object of interest is performed using the full camera image through a YOLOv11 model specifically trained for the class corresponding to the object to be grasped. The detector generates several bounding boxes with their confidence intervals, and the one with the highest score is selected as long as it exceeds a predefined minimum threshold. From the predicted bounding box, a region of interest (ROI) is generated, centered on that bounding box and enlarged by 10% with respect to its original size to ensure that the object remains fully contained even in the presence of small detection inaccuracies.

This ROI is obtained by cropping the original image, and the displacement of its upper-left corner $(ox, oy)$ is recorded, indicating the translation between the coordinate system of the ROI and that of the original image. This cropped region is used as input in the keypoint estimation stage, significantly reducing the influence of environmental elements and the computational cost.

Neural Network Keypoint Estimation

Once the region of interest has been extracted, a YOLOv11-pose model trained to predict a labeled set of 2D keypoints associated with specific features of the object, such as corners, edges or manipulation elements, is applied. The model produces, for each keypoint, its coordinates in the cropped ROI image. Since the predicted keypoints are defined in the ROI coordinate system, it is necessary to transform them to the original image coordinate system before using them in the pose estimation. To do this, the displacement introduced by the cropping is corrected by adding the ROI origin offset $(ox, oy)$ to each keypoint $(u_i^{\mathrm{ROI}}, v_i^{\mathrm{ROI}})$, thus obtaining their coordinates in the full image as

$$u_i^{\mathrm{img}} = u_i^{\mathrm{ROI}} + o_x, \quad v_i^{\mathrm{img}} = v_i^{\mathrm{ROI}} + o_y$$

In this way, the valid keypoints are expressed in the same reference system as the camera image and can be used as coherent 2D correspondences in the subsequent 3D pose estimation stage.

Pose Estimation via PnP

Once the 2D keypoints predicted by the neural network have been obtained and each prediction has been associated with its corresponding label $(p_i \in \mathbb{R}^2)$, it becomes possible to build the correspondences required to estimate the pose of the object. To do this, it is necessary to know the specific point of each label in the object geometry whose three-dimensional position $(X_i \in \mathbb{R}^3)$ must be known beforehand and defined in a 3D model of the object. In this way, a set of pairs $(p_i, X_i)$, where $p_i = (u_i, v_i)$ are the 2D coordinates of the keypoint in the image, and $X_i$ is its corresponding 3D point in the reference frame of the object.

With these 2D–3D correspondences, the pose estimation problem is formulated as a *Perspective-n-Point* (PnP) problem. The goal of PnP is to determine the rigid transformation formed by the rotation $R \in SO(3)$ and the translation $t \in \mathbb{R}^3$, which aligns the three-dimensional model of the object with its projection onto the image plane. This relationship is governed by the pinhole camera model and explicitly depends on the camera intrinsic matrix $K$, which encodes fundamental parameters of the optical system such as the focal lengths and the principal point. The projection model is expressed as:

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]\mathbf{X}_i, \tag{19}$$

where $\lambda_i$ is a scale factor associated with the depth of each point. Solving the PnP problem consists of finding the parameters $R$ and $t$ that minimize the reprojection error between the observed image positions and those projected by the geometric model.

In the proposed implementation, pose estimation is performed using a robust PnP-based approach with RANSAC (Random Sample Consensus), whose objective is to obtain a reliable solution even in the presence of erroneous keypoint predictions. This method evaluates different pose hypotheses generated from subsets of 2D–3D correspondences and selects the one that maximizes the number of consistent correspondences according to a reprojection error threshold. In cases where RANSAC [22] does not provide a valid solution, an iterative refinement method is applied that progressively adjusts the rotation and translation parameters to minimize this error. As a result, a complete and accurate estimation of the 6D pose of the object with respect to the camera is obtained.

#### 2.3.2. Local Grasp Perception: Segmentation and Grasp Points Determination

At short distances, the object is no longer completely visible within the camera's field of view, which causes strategies based on global pose estimation to lose reliability. At this stage, perception must focus exclusively on the region associated with grasping. For this reason, a specific short-range perception module based on segmentation techniques is employed, whose objective is to accurately isolate the functional region intended for grasping and to extract, directly in image coordinates, the necessary information to successfully perform the grasping maneuver, as summarized in Figure 3.

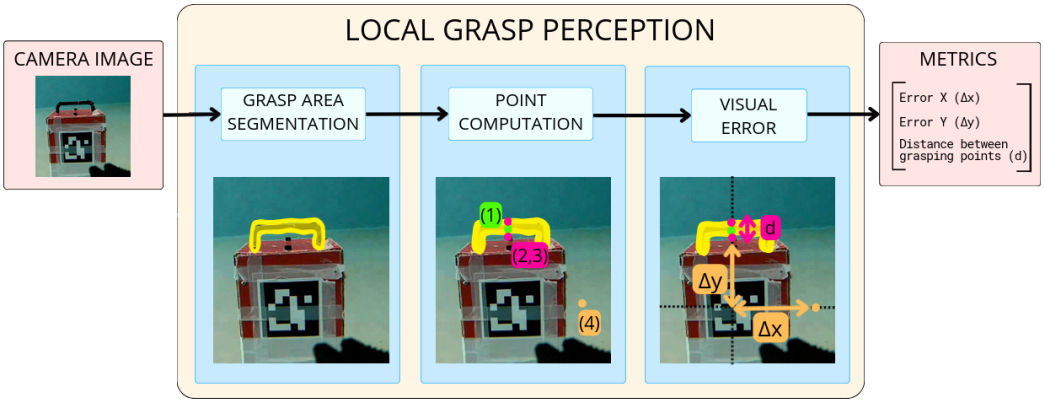


**Figure 3.** Overview of the local grasp perception object perception module. The numbered points indicate: (1) segment centroid, (2,3) grasping points, and (4) visual reference point. The final metrics are the alignment errors $(\Delta x, \Delta y)$ and the distance between grasp points $d$.

To locate the grasping region, segmentation is performed using YOLO. Unlike the long-range perception phase, where the complete geometry of the object is considered, at this stage the objective is to extract only the part that is relevant for manipulation.

Once the grasping element mask is obtained, the geometric centroid of the segment is computed in image coordinates. This centroid is used as a robust visual reference for the position of the manipulable area. Simultaneously, a fixed target point is defined in the image, representing the desired position of the center of the segmented region during the approach phase. The alignment error between the centroid of the segment and the target point defined on the image gives rise to the terms $(\Delta x, \Delta y)$, which indicate the horizontal and vertical displacement required to correct the relative position between the robot and the object during the final approach.

To compute the grasping points, the second-order central moments of the segmented mask $(\mu_{20}, \mu_{02}, \mu_{11})$ are used, from which the orientation of the principal axis of the region is estimated:

$$\theta = \frac{1}{2} \arctan\left(\frac{2\,\mu_{11}}{\mu_{20} - \mu_{02}}\right). \tag{20}$$

From this angle, an orthogonal axis is defined, which corresponds to the closing direction of the gripper. Along this direction, a line is constructed that passes through the centroid of the region and intersects the contour of the object.

Subsequently, the intersections between this line and the contour of the segmented mask are computed. From all the intersections obtained, the two that are farthest apart from each other along this line are selected, being interpreted as the extremes of the grasping region. These two points define the grasping points in image coordinates. The Euclidean distance between the two grasp points,

$$d = \left\| \mathbf{p}_{\text{grasp}}^{(1)} - \mathbf{p}_{\text{grasp}}^{(2)} \right\|, \tag{21}$$

where $\mathbf{p}_{\text{grasp}}^{(1)}$ and $\mathbf{p}_{\text{grasp}}^{(2)}$ represent the two extreme grasp points in image coordinates, is used as a measure of the apparent size of the object, serving as an indirect visual reference of the proximity to the object during the final approach phase.

This perception module was designed to align the robot frontally with the handle to initiate the grasp from this configuration, assuming that the object is initially presented in a vertical pose. Nevertheless, a successful recovery may still be achieved as long as the orientation of the handle is compatible with the approach direction of the gripper. Since the gripper does not provide rotational degrees of freedom, recovering objects with arbitrary orientations would require the use of an underwater vehicle–manipulator system (UVMS). In cases where the object is partially buried or tilted, the vehicle would need to compensate by introducing initial roll or pitch offsets, which could be estimated from a PnP-based model during the global object perception phase. In the current implementation, the robot assumes zero initial roll and pitch. Regarding occlusions, as the vehicle approaches the handle, the fisheye camera generally provides an almost complete view of the handle at the grasping stage; however, if occlusions occur, the computed grasping points would no longer be valid.

### *2.4. Natural Language ROS Agent*

This section describes the natural language interaction module developed to enable high-level control of the robotic system. The main objective of this module is to facilitate communication between the operator and the robot through instructions expressed in natural language, eliminating the need to interact directly with low-level commands or internal system configurations. To this end, the system incorporates an agent based on language models that allows transforming instructions expressed in natural language into executable actions within the robot's ROS architecture, as illustrated in Figure 4.

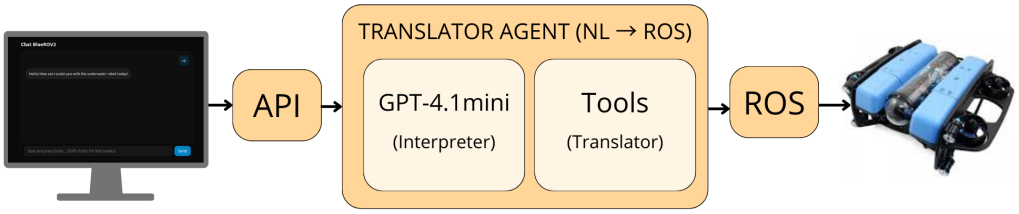


**Figure 4.** Architecture of the natural language translation agent to ROS for robot control.

The commands entered by the user are sent through an API to an agent that integrates the GPT-4.1 mini language model [23], which is responsible for interpreting the semantic intent of the instruction and generating a structured response in terms of high-level actions.

This response is not directly translated into low-level control commands, but rather into the selection of a predefined system actions, also known as tools. These actions act as an intermediate abstraction layer, completely decoupling human interaction from the internal control of the robot. Based on this selection, the agent is responsible for translating the high level orders into concrete operations within the ROS architecture, such as starting or stopping nodes, enabling or disabling functional modules, or publishing messages on specific topics that govern the behavior of the system.

In this way, the agent acts as a bridge between natural language and the robotic system, enabling high level, intuitive, and flexible interaction, without requiring the user to have knowledge of the internal structure of ROS or low level control mechanisms.

To ensure safe, predictable, and deterministic behavior, the set of available actions is strictly limited to a closed set of predefined tools. This ensures that the language model can only activate previously validated behaviors, preventing unwanted or potentially dangerous executions within the robotic system.

### 2.5. System Architecture

The system architecture is organized into three main blocks—perception, control, and the robotic platform—all integrated through ROS and ArduSub, as shown in Figure 5. The perception block processes the images captured by the system cameras and extracts the visual information required for each mission phase, publishing its results in ROS.

The control layer uses this perceptual information to generate the actions required to fulfill the user's requests through the ROS-based agent, producing motion commands in the form of ROS messages. These commands are translated into PWM signals by a PyMAVLink-based bridge, which acts as an interface between ROS and the ArduSub firmware. The firmware operates in Depth Hold mode, meaning that depth control as well as pitch and yaw attitude control are handled internally by the firmware itself, while the external system is responsible for defining high level motion references.

Finally, the BlueROV2 executes the generated commands by controlling the thrusters and the grasping system. It is important to mention that the autonomous execution is only active while the operator continuously presses and holds an enable button. If the operator releases this button, the Autonomous Relay module immediately interrupts the autonomous behavior and returns control to the operator, regardless of the current state of the grasping sequence as it is shown in Figure 5.

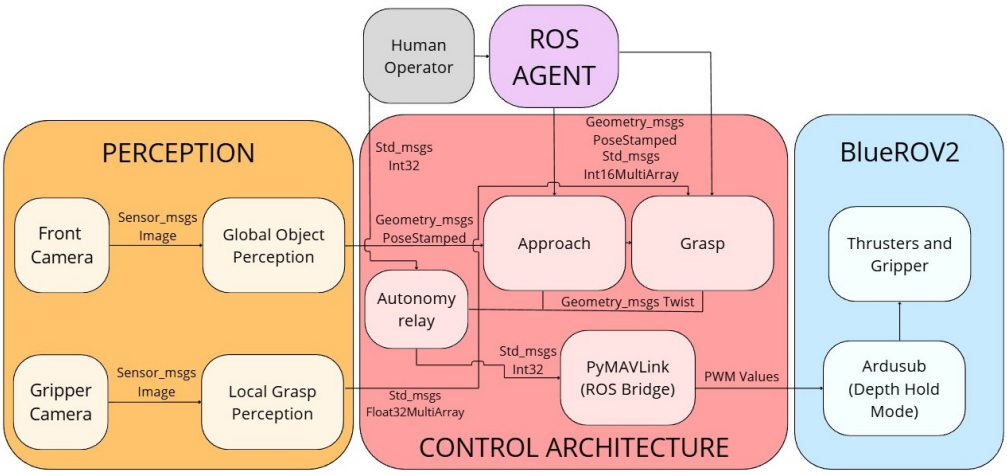


**Figure 5.** Block diagram of the proposed architecture integrating a ROS agent with ArduSub for autonomous underwater manipulation.

## 3. Experimental Setup

This section presents the experimental configuration adopted to validate the proposed system. First, the mechatronics integration of the robotic platform used is described. Next, the simulation environment employed for preliminary validation is presented. Finally, a real practical case is introduced to evaluate the performance of the system.

### 3.1. Mechatronics Integration

For this experiment, the BlueROV2 [24] was used in the heavy (1) configuration, which adds two thrusters to the standard configuration, providing the vehicle with 6 DoF, as illustrated in Figure 6.

The system is connected to the ground control station via an umbilical cable (2), which is responsible for data transmission between the vehicle and the ground control station. It is important to note that all perception and planning processing is executed on the ground control station, rather than onboard the vehicle, which allows the use of vision models and algorithms with higher computational requirements without compromising the system's

capabilities. At the control station, an ASUS ROG Flow 14 (ASUSTeK Computer Inc., Taipei, Taiwan) equipped with Ubuntu 20.04, featuring an AMD Ryzen 7 processor (Advanced Micro Devices, Inc., Santa Clara, CA, USA), 16 GB of RAM, and an NVIDIA GeForce GTX 1650 GPU and ROS Noetic. Under this configuration, the total end-to-end latency from image acquisition to control command publication is approximately 170 ms. The video stream is encoded in H.264 and transmitted through the Fathom Ethernet tether, introducing a transmission-related delay of about 40–50 ms, including encoding, decoding, and buffering. On the GCS, the YOLO-based perception module requires approximately 40 ms per frame, while the PnP-based pose estimation takes around 90 ms. Finally, the control computation and ROS publication introduce a negligible delay of 0.6 ms.

The front camera (3) corresponds to a Low-Light HD USB Camera [25], integrated into a waterproof housing for operation in an underwater environment. The gripper camera (4) corresponds to the DeepWater Exploration exploreHD USB Camera [26]. Having a camera in this position has the advantage of providing information on how the grasping action is carried out, which is especially useful in manipulation processes, as it offers a more precise observation of the interaction between the manipulator and the object.

Finally, the gripper (5) is the Newton Subsea [27], designed to operate in underwater environments. This actuator allows the opening and closing of the jaws, enabling simple object grasping.

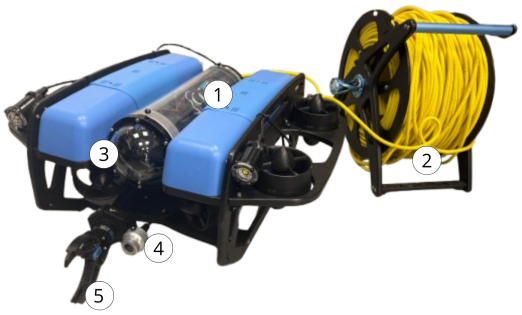


**Figure 6.** (1) BlueROV2 Heavy, (2) umbilical tether, (3) frontal camera, (4) gripper camera, (5) Newton Subsea Gripper.

### *3.2. Simulation*

The simulator used was Stonefish [28], designed to realistically simulate the behavior of robots in underwater environments. One of its main advantages is that it computes hydrodynamic forces and underwater contact according to the shape of each object, providing more realistic simulations. In addition, it includes its own rendering system that realistically represents the ocean, the atmosphere, and the underwater world, including effects such as light refraction and reflection underwater. The Figure 7 shows the simulation environment used in this work.

To reproduce the behavior of the real BlueROV2, a simulation architecture is used that integrates Stonefish, ROS, and ArduSub SITL [29] in a closed-loop control cycle. Stonefish acts as the physical engine of the system: it simulates the vehicle dynamics, the aquatic environment, and the hydrodynamic interactions, generating at each iteration sensor data equivalent to what the real robot would produce. This data (IMU, GPS, odometry, thruster setpoints) is published in ROS through the topics of the stonefish_ros [30] package, where it acquires the same format and structure as the physical sensors of the BlueROV2.

An intermediate node, called the bridge, receives these signals and assembles them into a structured JSON packet, following the specification required by ArduSub. SITL returns these PWM values to the bridge in a compact binary format. The node decodes and normalizes them, then publishes them in ROS as a Float64MultiArray on the

/bluerov/thrusters_setpoints topic, from which Stonefish updates the dynamic state of the vehicle by applying the forces associated with each thruster. Thanks to this, the flight controller operates as if it were onboard a physical ROV, using the simulated input to compute the PWM commands sent to the thrusters.

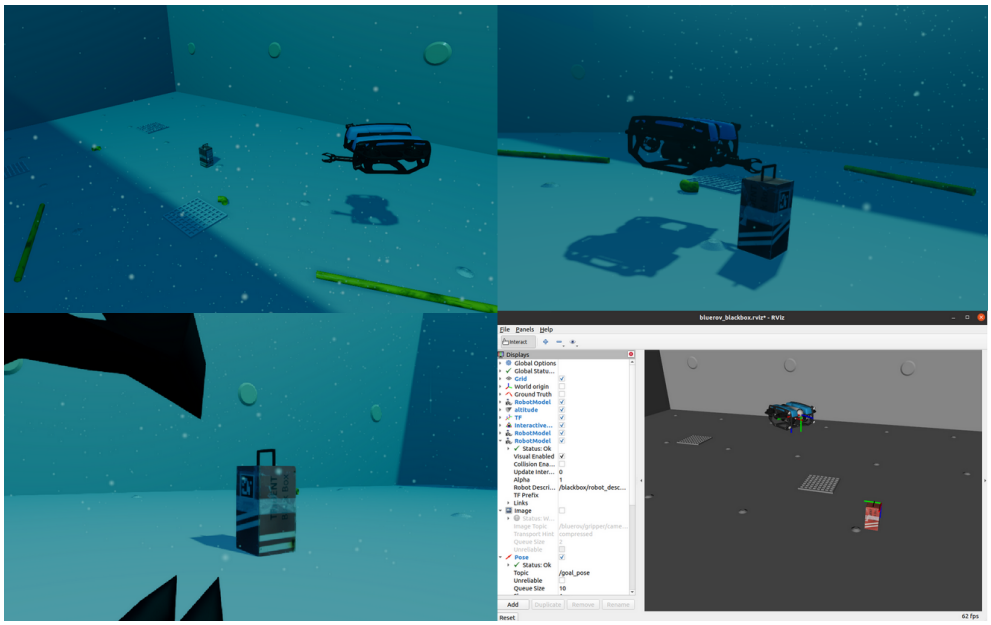

**Figure 7.** Upper images: external views of the simulation. Bottom left: gripper camera view. Bottom right: RViz visualization.

This workflow, shown in Figure 8 ensures that the simulated vehicle responds to commands exactly as the real robot would, allowing the ROV to be operated from any ROS interface or from PyMAVLink [31] without additional modifications, thus providing a realistic, reproducible, and fully compatible testing environment with the final system.

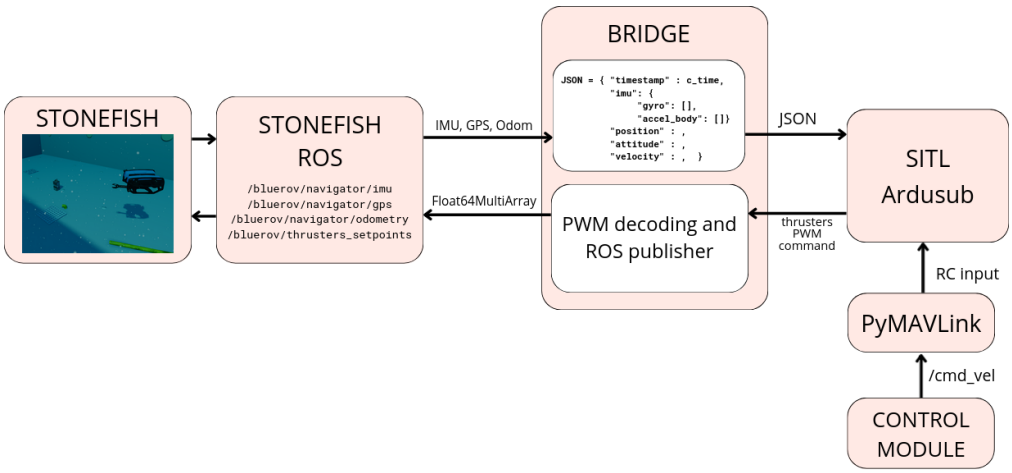


**Figure 8.** Data Flow Between ROS, ArduSub SITL, and Stonefish Simulator.

### *3.3. Case Study*

To experimentally validate the proposed system, a case study was conducted focusing on the autonomous recovery of a prototype aircraft black box.

The experiments were carried out in the 12 × 8 × 5 m test tank at CIRTESU, where a workspace was set up including the BlueROV2 Heavy equipped with the described perception and grasping system, the ground control station responsible for visual processing

and mission supervision, and the test object positioned at the bottom of the tank. This controlled environment allows for safe and repeatable evaluation of the performance of the full pipeline, including global object perception, local grasp perception, and grasp execution.

Figure 9 shows an overview of the experimental setup, including the vehicle, the location of the processing computer, and the position of the target inside the tank.

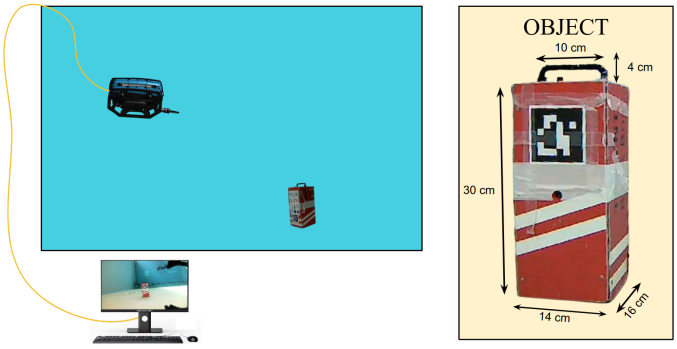


**Figure 9.** The experimental setup is illustrated on the left, whereas the object is shown on the right.

## 4. Results

This section presents the results obtained when evaluating the operation of the system. On the one hand, the performance of the perception modules, as well as the response of the controller during the approach and execution of the grasp. In addition, the behavior of the natural language agent is examined, showing how it interacts with these modules, coordinating their activation and managing the mission flow based on instructions expressed in natural language. The following subsections present these results in a structured manner. A video of the simulation demonstration is available at https://youtu.be/o-8nKwHwujA and the video of the real tank experiment is available at https://youtu.be/GTfr_6CrBok (both accessed on 15 January 2026).

### *4.1. Perception and Pose Estimation*

The validation of this module was carried out following a gradual strategy ranging from simulation to real conditions at sea. Initially, the perception models were trained and evaluated in a simulated environment, where visual conditions can be controlled and precise annotations are available. Next, the model was tested directly in the test tank to verify its performance in a real environment. A subsequent fine-tuning process was conducted to adapt the simulation-trained models to the real domain and improve their robustness. Finally, only the perception and pose estimation was evaluated in a real marine environment, specifically in Castellón harbor, at a depth of 6 m, validating its ability to operate under significantly more demanding conditions, Figure 10 shows an image of the blackbox in that environment.

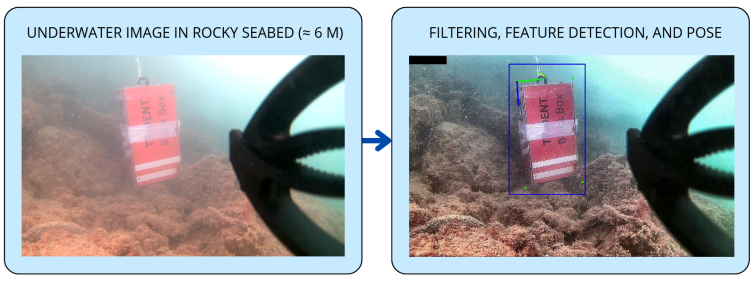


**Figure 10.** Underwater image before and after visual preprocessing and pose estimation in a real marine environment.

#### 4.1.1. Simulation

For object detection, a YOLOv11n model was used, while keypoint estimation was performed using YOLOv11n-pose. Both models were trained on a dataset of 450 images for 80 epochs. Figure 11 shows a comparison between the pose estimated by the PnP method and the ground-truth pose obtained from the simulation.

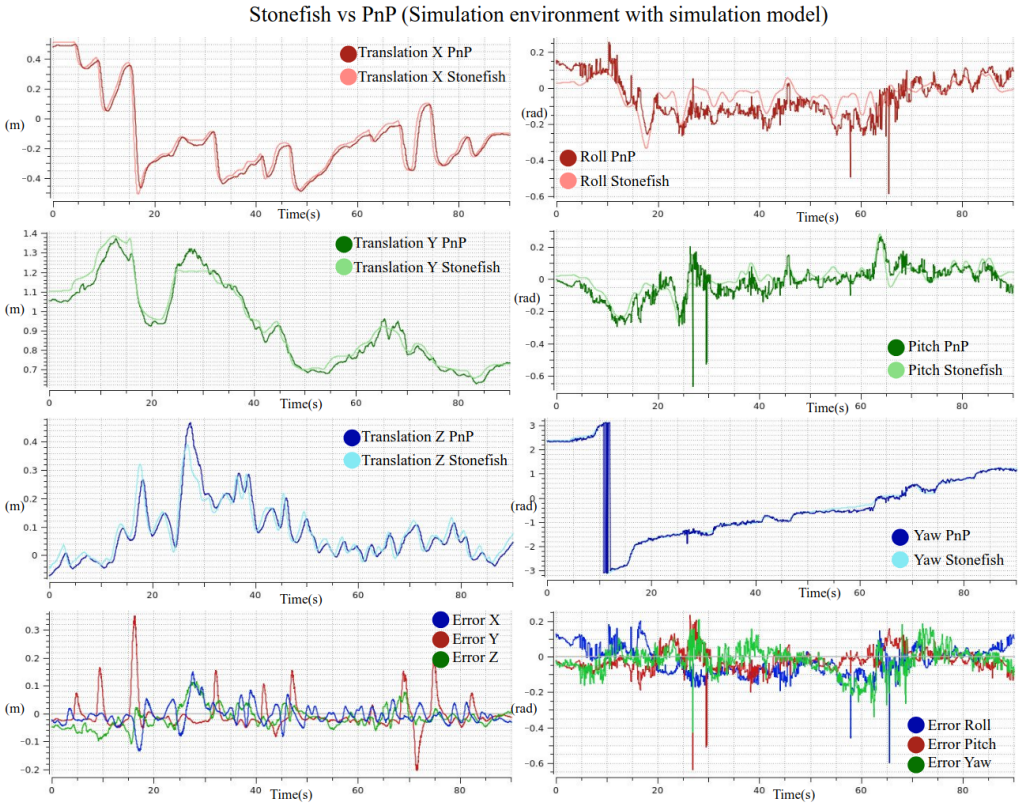


**Figure 11.** Comparison between the pose estimated by the PnP method and the ground-truth pose from the simulation. The plots show the linear components x, y, z, the angular components roll–pitch–yaw and the errors all expressed in the camera frame and representing the position and orientation of the box with respect to the camera.

#### 4.1.2. Simulation to Real

The system trained and tested in simulation is now evaluated using the real robot in a water tank. For ground truth, ArUco markers are placed on the black box. Figure 12 shows a comparison between the pose estimated by the PnP method and the ArUco markers.

#### 4.1.3. Real Fine-Tuning

To improve performance under real conditions, the model trained in simulation was refined through fine-tuning using a dataset of 150 images captured in the tank environment. This dataset included samples captured under different illumination conditions and visibility levels, including low-visibility scenarios.

This real-image dataset was collected under varying illumination conditions, backgrounds, and with the target object placed at different positions and orientations with respect to both the tank and the robot, in order to increase visual diversity and reduce the risk of overfitting. Fine-tuning was performed using a very low learning rate to ensure a smooth adaptation of the simulation-trained model. The objective was not to specialize the network to the experimental setup, but to promote the generalization of the features learned in simulation to real-world conditions. The adjustment was performed starting from the simulation-trained model, using a reduced learning rate (0.0004), which allowed

the weights to be progressively adapted without overwriting the previously acquired knowledge. The training was carried out for 40 epochs, freezing the initial feature extraction layers. Figure 13 shows a comparison between the pose estimated by the PnP method and the ArUco markers.

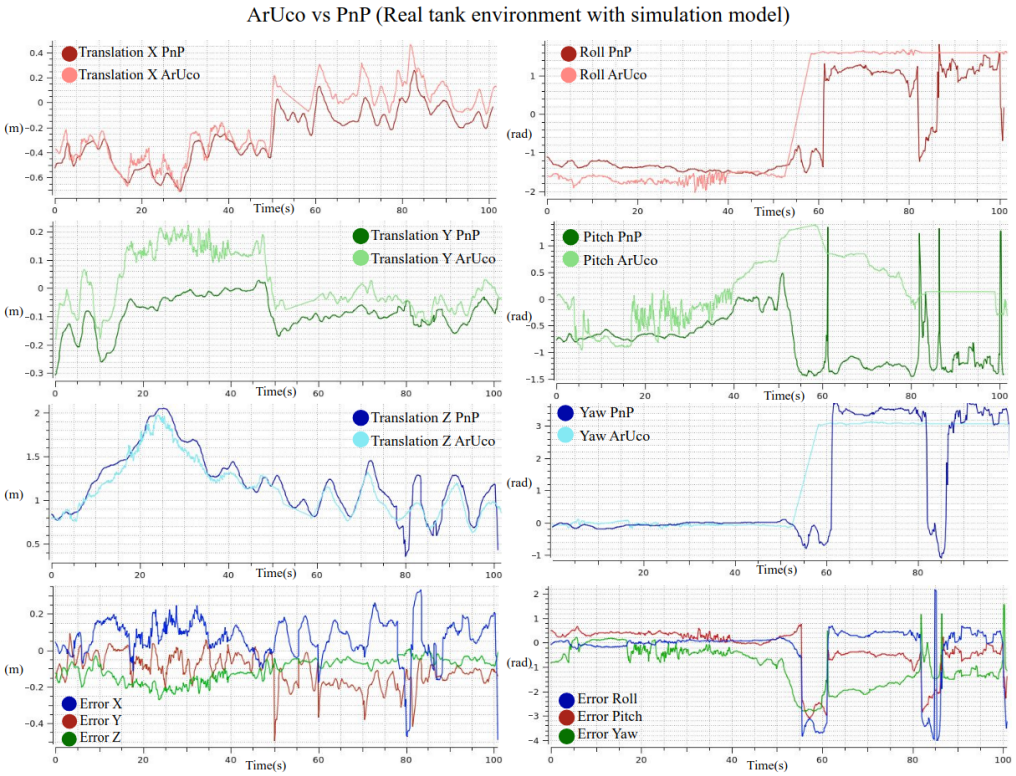


**Figure 12.** Comparison between the pose estimated by the PnP method and the ground-truth pose obtained from ArUco markers. The plots show the linear components x, y, z, the angular components roll–pitch–yaw and the errors, all expressed in the camera frame and representing the position and orientation of the box with respect to the camera.

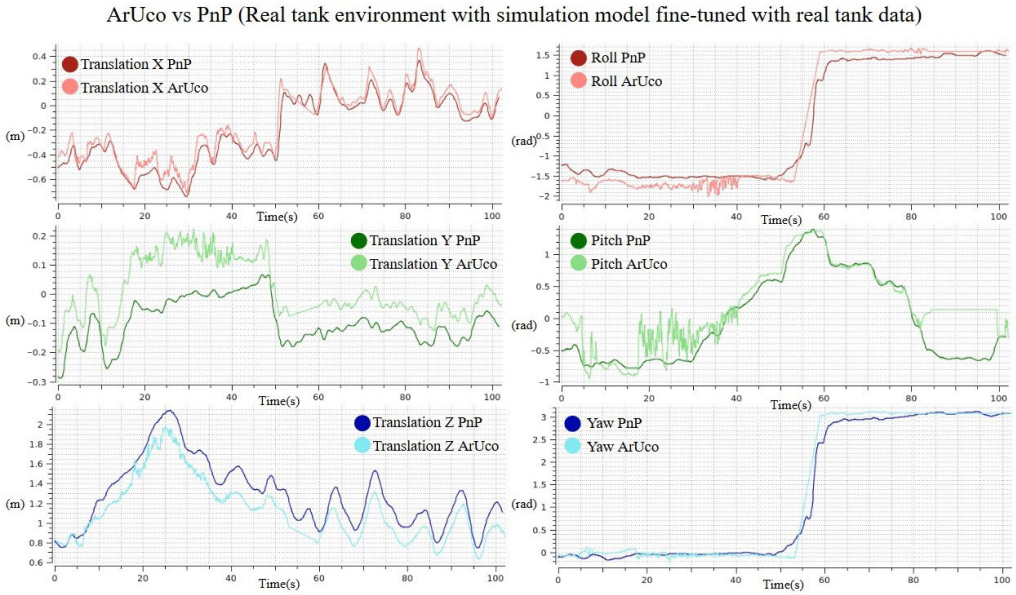


**Figure 13.** Comparison between the pose estimated by the PnP method using the real fine-tuning and the ground-truth pose obtained from ArUco markers. The plots show the linear components x, y, z and the angular components roll–pitch–yaw, all expressed in the camera frame and representing the position and orientation of the box with respect to the camera.

#### 4.1.4. Fine-Tuned Model Trained in the Tank and Tested at Sea

As a final experiment, the model refined through fine-tuning is evaluated in a real marine environment, with a rocky seabed and at an approximate depth of 6 m. Since under these conditions image quality is degraded by turbidity and low contrast, a visual preprocessing step is applied to each frame prior to inference. Specifically, the implemented filter combines white balance through channel equalization, gamma correction to enhance shadow information, local contrast enhancement using CLAHE [32] (Contrast Limited Adaptive Histogram Equalization) on the luminance channel, and mild sharpening. This preprocessing aims to improve the visibility of relevant edges and textures without modifying the model architecture, which improves detection stability in degraded underwater environments, as illustrated in Figure 10. For further details on underwater image enhancement and dehazing techniques, the reader is referred to [33], which presents previous research carried out at the IRS Lab.

Figure 14 shows a comparison between the pose estimated by the PnP method and the ArUco markers.

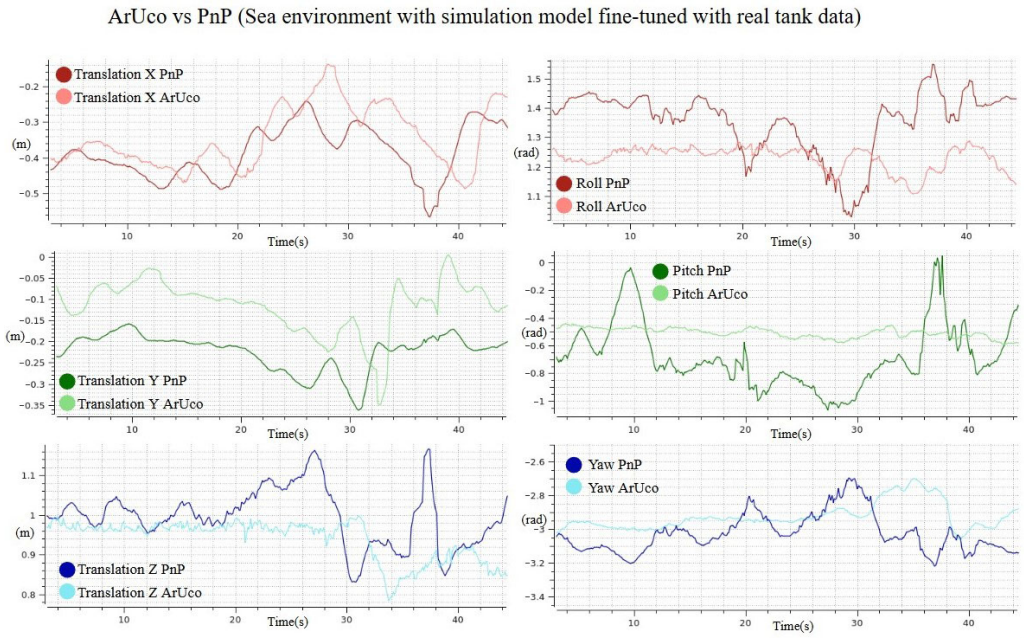


**Figure 14.** Comparison between the pose estimated by the PnP method using the fine-tuning and the ground-truth pose obtained from ArUco markers in harbor conditions. The plots show the linear components x, y, z and the angular components roll–pitch–yaw, all expressed in the camera frame and representing the position and orientation of the box with respect to the camera.

Tables 2–7 summarize the translation and rotation errors obtained under the different experimental configurations (simulation, real-world tank environment, and marine environment with and without image preprocessing), comparing the estimated pose with the ground truth, and are quantified using the mean value, the standard deviation, and the root mean square error.

**Table 2.** Translation error in $x$ [m].

| Metric | Sim. | Real + Sim. | Real FT | Sea | Sea (Filtered) |
|---|---|---|---|---|---|
| Avg | −0.0071 | −0.1041 | −0.0567 | −0.1060 | −0.0588 |
| Std | 0.0551 | 0.0833 | 0.0515 | 0.0184 | 0.0762 |
| RMSE | 0.0556 | 0.1334 | 0.0766 | 0.1076 | 0.0962 |

**Table 3.** Translation error in $y$ [m].

| Metric | Sim. | Real + Sim. | Real FT | Sea | Sea (Filtered) |
|---|---|---|---|---|---|
| Avg | −0.0151 | −0.1082 | −0.1102 | −0.0981 | −0.1157 |
| Std | 0.0372 | 0.0628 | 0.0463 | 0.0264 | 0.0546 |
| RMSE | 0.0402 | 0.1251 | 0.1195 | 0.1015 | 0.1279 |

**Table 4.** Translation error in $z$ [m].

| Metric | Sim. | Real + Sim. | Real FT | Sea | Sea (Filtered) |
|---|---|---|---|---|---|
| Avg | −0.0044 | 0.0704 | 0.1667 | 0.0622 | 0.0867 |
| Std | 0.0351 | 0.1191 | 0.0735 | 0.0189 | 0.1143 |
| RMSE | 0.0354 | 0.1383 | 0.1822 | 0.0651 | 0.1435 |

**Table 5.** Rotation error in roll [rad].

| Metric | Sim. | Real + Sim. | Real FT | Sea | Sea (Filtered) |
|---|---|---|---|---|---|
| Avg | −0.0245 | −0.1802 | 0.0393 | −1.3421 | 0.1556 |
| Std | 0.0803 | 0.6938 | 0.2054 | 0.8094 | 0.1315 |
| RMSE | 0.0839 | 0.7168 | 0.2091 | 1.5673 | 0.2037 |

**Table 6.** Rotation error in pitch [rad].

| Metric | Sim. | Real + Sim. | Real FT | Sea | Sea (Filtered) |
|---|---|---|---|---|---|
| Avg | −0.0221 | −0.8502 | −0.2285 | −0.2020 | −0.0195 |
| Std | 0.0615 | 0.7257 | 0.3043 | 0.9473 | 0.4311 |
| RMSE | 0.0654 | 1.1178 | 0.3806 | 0.9686 | 0.4316 |

**Table 7.** Rotation error in yaw [rad].

| Metric | Sim. | Real + Sim. | Real FT | Sea | Sea (Filtered) |
|---|---|---|---|---|---|
| Avg | −0.0232 | 0.2581 | −0.0296 | 1.5686 | −0.1419 |
| Std | 0.0741 | 0.7297 | 0.0952 | 0.9154 | 0.1663 |
| RMSE | 0.0777 | 0.7740 | 0.0997 | 1.8161 | 0.2186 |

### *4.2. Approach Phase*

The obtained results show a consistent behavior of the pose estimation system across the different evaluated scenarios. Under simulation conditions, the model achieves the best metrics, reflecting a precise and stable estimation. When the model trained in simulation is directly applied to real data, an increase in error is observed, particularly in the rotational components, highlighting the impact of the gap between the simulated and real domains. The fine-tuning process using real data effectively mitigates this effect, reducing both the mean error and the variability of the estimation. In the marine environment, the system maintains satisfactory performance despite visual degradation; this favorable behavior may be attributed to the fact that the sequences acquired in this scenario are shorter than in other environments and exhibit a smaller variation in the camera viewpoint relative to the object.

The grasping phase consists of bringing the robot to a distance of 1.5 m along the x-axis, while maintaining alignment in y, z, and yaw, so that the PnP-based perception system can operate reliably. The results of this phase can be seen in the Figures 15 and 16.

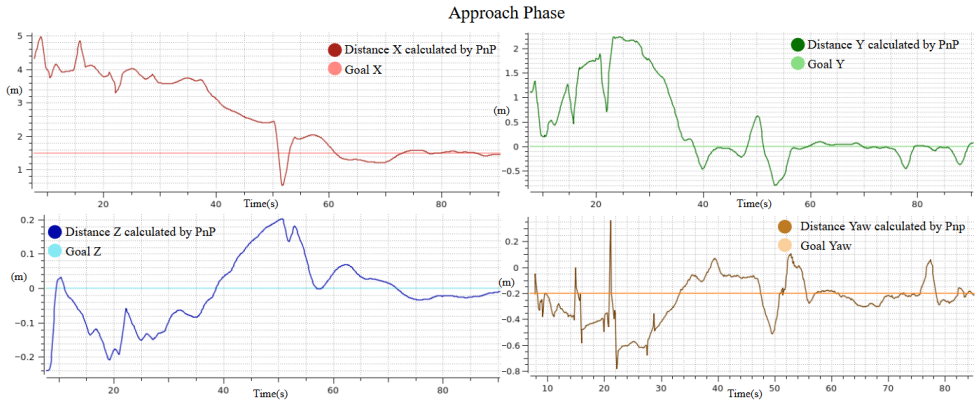


**Figure 15.** Evolution of the camera-to-blackbox relative pose during the approach phase, showing the PnP-estimated position (x, y, z) and yaw angle of the camera with respect to the blackbox, along with the target reference values used for alignment.

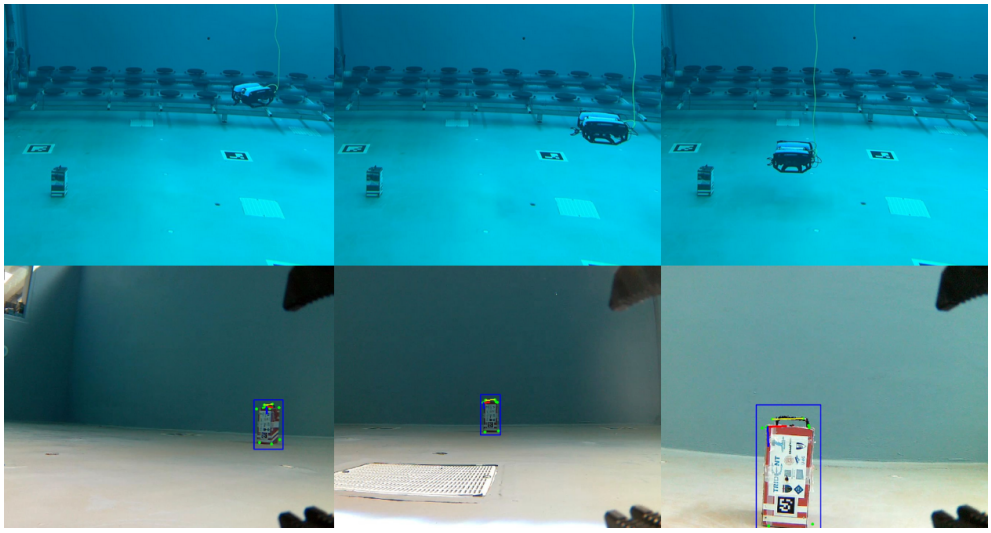

**Figure 16.** Sequence of images during the approach phase. The top row shows the robot's external view, and the bottom row shows the corresponding onboard camera images.

### *4.3. Grasping Phase*

This phase begins once the approach phase has been completed and the robot is positioned at 1.5 m along the x-axis from the box. The segmentation model is then used, and the robot advances along the x-axis while aligning itself using the pixel error computed by the perception system, until the distance between the grasping points exceeds 110 pixels. At this point, the gripper is closed, completing the grasp of the black box. The 110 pixels threshold was empirically calibrated for the specific object and camera setup used in these experiments and would need to be adjusted for objects with different sizes or geometries. The results of this phase can be seen in the Figures 17 and 18.

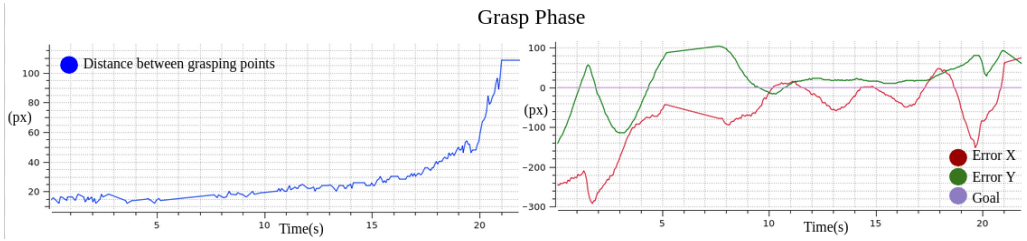


**Figure 17.** Alignment error metrics with respect to the target point in the x and y directions (**right**), and distance between the grasping points (**left**).

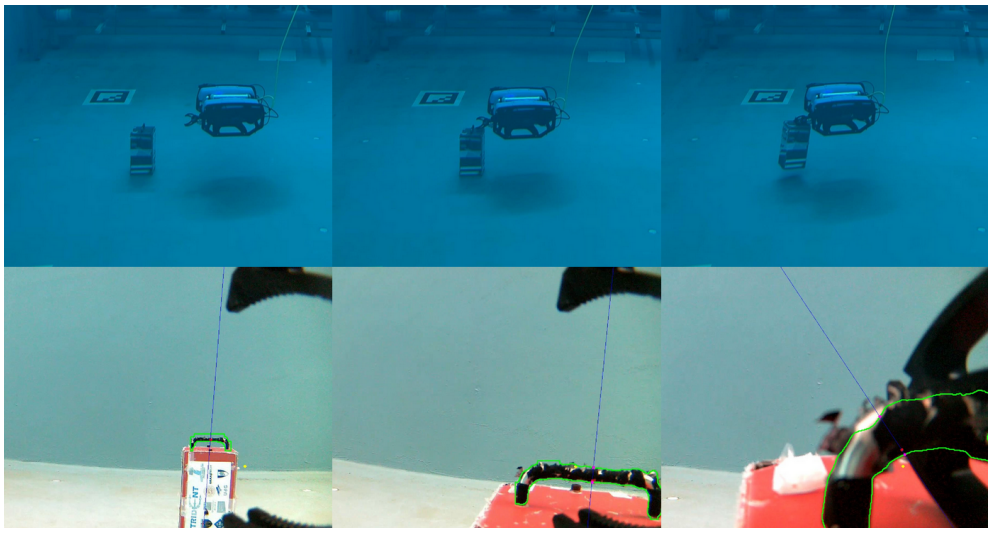

**Figure 18.** Sequence of images during the grasping phase. The top row shows the robot's external view, and the bottom row shows the corresponding onboard camera images.

This phase has been evaluated ten times both in simulation and in real-world experiments in order to assess the robustness and repeatability of the proposed grasping strategy. Each trial was considered successful when the robot was able to completely grasp the object. The quantitative results of these experiments, including the number of successful grasps in each scenario, are summarized in the Table 8.

**Table 8.** Grasping success rate in simulation and real-world experiments.

| Metric | Simulation | Real |
|---|---|---|
| Success rate | 8/10 | 7/10 |

### *4.4. ROS Agent*

This section evaluates the behavior of the natural language–based agent, focusing on how high-level instructions provided by the user are translated into executable actions, as illustrated in Figure 19.

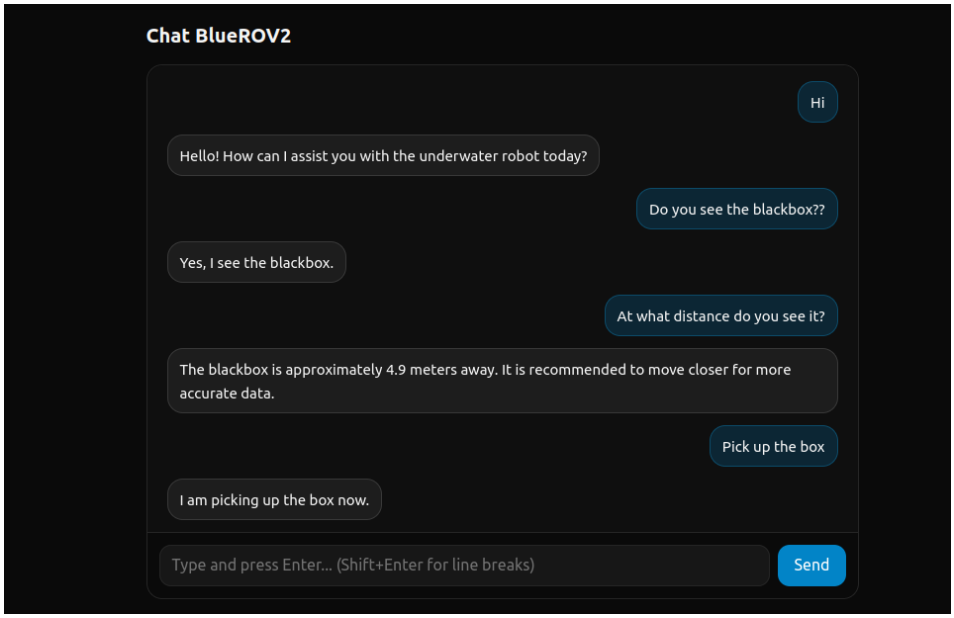


**Figure 19.** Overview of the natural language agent interface and its interaction workflow.

To enable natural language interaction, the agent is provided with a reduced set of tools that represent high-level actions previously defined within the system. These tools encapsulate functional behaviors such as initiating environment perception, activating object pose estimation, transitioning to the approach phase, or executing the grasping maneuver. Throughout the experiments, the agent automatically selects the most appropriate tool based on the instructions provided by the user, allowing the mission to be described through simple and intuitive commands, such as "do you see the object?", "align with the box", or "grasp the black box", without the need to specify low level commands.

As a representative example, the tool responsible for positioning the robot at a specific relative distance with respect to the detected object is described below, as illustrated in Figure 20. Given a natural language instruction, such as "position yourself 3 meters in front of the box," the agent interprets the user's intent and translates the command into a control reference expressed in the object camera frame. This reference is sent to the control system by publishing a `goal_pose` message, in which the desired position relative to the object is specified, for example $(x = 3; y = 0; z = 0)$, while keeping the remaining components at their nominal values. To avoid acting on all axes simultaneously unless explicitly requested by the user, the tool publishes an `axis_mask` vector that defines which degrees of freedom should be actively controlled and which can be ignored during motion execution. This mechanism allows the robot's behavior to be adapted to different high level commands; for instance, an instruction such as "align in y" activates only the corresponding axis and the yaw alignment for improved operation, using a mask of the form $[0, 1, 0, 0, 0, 1]$. In this way, a single tool enables flexible positioning behaviors that are consistent with the user's expressed intent.

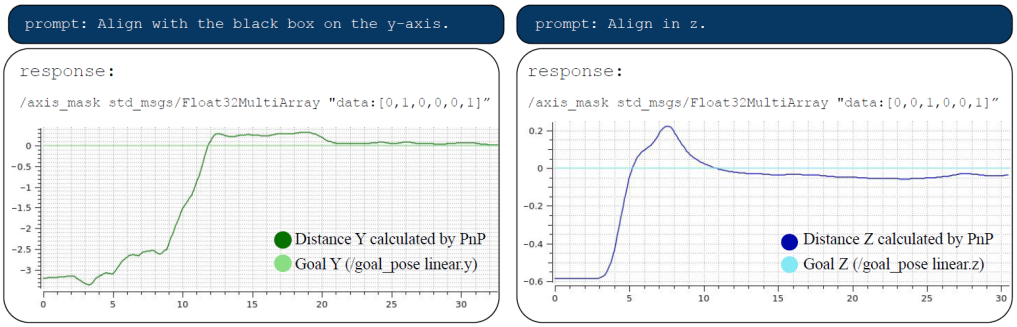


**Figure 20.** Examples of natural language prompts and the corresponding system responses.

It is important to note that the proposed natural language ROS agent is not intended to replace traditional graphical user interfaces based on buttons and sliders, but to act as a supervisory tool at the mission-command level. For instance, the agent could be integrated into conventional GUIs through voice recognition, providing an additional and intuitive interaction modality for real-world underwater intervention missions. Finally, the latency introduced by the agent is approximately 5.186 s depending on the input and the network connections. But, as the generated high-level commands are limited to triggering predefined actions and are not part of the real-time control loop the stability of the controller and the mission is not in risk.

## 5. Conclusions

This article has presented a hybrid-controlled ROV architecture for underwater object recovery. The proposed architecture combines a perception module capable of estimating the 6DoF pose of a known object using a monocular camera, neural network models for segmentation and feature detection, and a PnP-based pose estimation module. The perception system was validated under different scenarios. First, a model trained exclu-

sively with simulated images was evaluated in simulation, achieving an average Euclidean error of 1.73 cm. This same model was then tested in real tank experiments, where the average Euclidean error increased to 16.58 cm and a relatively high average rotational error of $-0.25$ rad was observed. After fine-tuning the model using real tank images, the average rotational error was reduced to $-0.0729$ rad. Finally, sea trials performed using the fine-tuned model showed good overall performance, with an average Euclidean error of 15.61 cm and an average rotational error of 0.1057 rad. These results demonstrate the feasibility of deploying the proposed perception system in real underwater environments when combined with the image filtering techniques described in this article.

The perception system presented in this article relies on a monocular camera for 6D pose estimation, which is particularly challenging in underwater environments due to light refraction and turbidity. Although promising results are obtained in simulation, in a water tank, and in a rocky underwater environment, this approach presents important limitations, mainly related to occlusions: when the target object is not fully visible, keypoints cannot be reliably detected and the PnP algorithm fails to estimate the object pose, an issue that is further aggravated by image noise and floating particles in the water. To improve the perception system, a stereo camera could be employed to estimate the object pose without relying on full object visibility; however, stereo-based approaches remain sensitive to illumination changes and require sufficient visual features in the scene or on the object to compute depth reliably. Alternatively, 6D pose estimation can be achieved using sonar-based systems, ranging from simple altimeters and multibeam sonars to modern 3D imaging sonars capable of generating point clouds, which, despite providing less detailed scene information, are more robust in turbid environments and offer higher overall robustness.

During the grasping stage, the proposed grasp strategy was evaluated both in simulation and in real water-tank experiments. The results show that, despite its simplicity, the PID-based controller exhibits good performance under low or negligible water-current conditions. However, when experiments were conducted under slightly higher currents generated by the tank filtration system, the grasp success rate decreased noticeably. In calm water conditions, the grasping success rate was approximately 70–80%, with the primary failure mode being the loss of the target from the camera field of view due to external disturbances, which triggered the abortion of the autonomous behavior. Overall, most failures are therefore attributed to the controller's limited ability to anticipate and compensate for higher water currents, rather than perception errors, which were observed to be minimal thanks to the robustness of the object detection model. Finally, a small number of failures were caused by mechanical constraints, as the available clearance for inserting the gripper into the box handle is limited; a flatter gripper geometry on the lower side could potentially improve the overall grasp success rate.

Regarding the use of a language model for high-level vehicle control, the experimental results and recorded videos validate the use of this technology as an intuitive human–robot interface, opening the possibility of incorporating microphones and speech-based interaction in future developments. In the experiments presented, the instructions were deliberately constrained to a limited and well-defined set of commands in order to ensure mission safety and protect the vehicle. Concerning the autonomous approach and grasping actions, the results show that these behaviors are robust, despite relying solely on monocular vision and being potentially sensitive to abrupt vehicle motions.

Finally, several directions for future work are identified. From a control perspective, the implementation of adaptive or nonlinear controllers would significantly improve system stability and accuracy, since controller gains should vary depending on the intervention phase and the distance to the target object. Another important research direction concerns

the localization of the real robot with respect to the world NED frame, which would enable the removal of the umbilical cable and represent a step towards increased autonomy. To increase this autonomy integrating an onboard GPU is feasible and has already been demonstrated on larger robotic platforms; however, the main drawback is the significantly higher power consumption, which would result in shorter mission durations.

The localization could be estimated using a downward-looking camera observing ArUco markers in the tank, or alternatively through sensor fusion approaches incorporating measurements from a DVL, depth sensor, USBL, IMU, AHRS, or INS. In terms of underwater manipulation, proposed improvements include enhancing the gripper design by integrating tactile or force sensors, or relocating the camera to the center of the gripper instead of the wrist. Additionally, the use of a manipulator with a higher number of degrees of freedom would substantially expand the range of possible intervention tasks. Finally, future work will focus on improving the detection models using images acquired in real marine environments, such as harbors, and on performing object recovery directly on the seabed. These steps would increase the robustness of the proposed algorithms and raise the technology readiness level of the system.

**Author Contributions:** Conceptualization, I.P.-E., S.L.-B., R.M.-P., and P.J.S.; methodology, I.P.-E., S.L.-B., R.M.-P., and P.J.S.; software, I.P.-E. and S.L.-B.; validation, I.P.-E., S.L.-B.; formal analysis, I.P.-E., S.L.-B.; investigation, I.P.-E., S.L.-B., R.M.-P., and P.J.S.; resources, R.M.-P., and P.J.S.; data curation, I.P.-E. and S.L.-B.; writing—original draft preparation, I.P.-E. and S.L.-B.; writing—review and editing, I.P.-E., S.L.-B., R.M.-P., and P.J.S.; visualization, I.P.-E., S.L.-B., R.M.-P., and P.J.S.; supervision, S.L.-B., R.M.-P., and P.J.S.; project administration, R.M.-P., and P.J.S.; funding acquisition, R.M.-P., and P.J.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This paper was partially supported by the Ministry of Science, Innovation and Universities of Spain, under PID2023-149910OB-C32 grant.

**Data Availability Statement:** Video of the simulation experiment is available at https://youtu.be/o-8nKwHwujA. Video of the real tank experiment is available at https://youtu.be/GTfr_6CrBok (both accessed on 15 January 2026). And the videos are also availiables in the Zenodo repository [34].

**Acknowledgments:** The authors would like to acknowledge the financial support provided by the TANDEM/CARMEN project under grant PID2023-149910OB-C32, which funds the doctoral FPI fellowship of Salvador López-Barajas. Additionally, the authors acknowledge the support of the Universitat Jaume I through the project with reference GACUJIMB/2024/12, which provided funding for the research grant of Inés Pérez-Edo. Finally, the authors would also like to thank Alejandro Solís, technician at CIRTESU, for his support in the mechatronic integration of the robotic system.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| AUV | Autonomous Underwater Vehicle |
| ROV | Remote Operated Vehicle |
| GCS | Ground Control Station |
| CIRTESU | Center for Robotics and Underwater Technologies Research |
| NED | North East Down |
| NL | Natural Language |
| SaR | Search and Rescue |
| CLAHE | Contrast Limited Adaptive Histogram Equalization |
| ROS | Robot Operating System |
| PnP | Perspective-n-Point |

## References


1. Bibel, G. *Beyond the Black Box: The Forensics of Airplane Crashes*; Johns Hopkins University Press: Baltimore, MD, USA, 2008. [CrossRef]
2. Benjamin, J.; O'Leary, M.; McDonald, J.; Wiseman, C.; McCarthy, J.; Beckett, E.; Morrison, P.; Stankiewicz, F.; Leach, J.; Hacker, J.; et al. Aboriginal artefacts on the continental shelf reveal ancient drowned cultural landscapes in northwest Australia. *PLoS ONE* **2020**, *15*, e0233912. Correction in *PLoS ONE* **2023**, *18*, e0287490. https://doi.org/10.1371/journal.pone.0287490. [CrossRef] [PubMed]
3. Van Der Heyde, M.; Alexander, J.; Nevill, P.; Austin, A.; Stevens, N.; Jones, M.; Guzik, M. Rapid detection of subterranean fauna from passive sampling of groundwater eDNA. *Environ. DNA* **2023**, *5*, 1055–1063. [CrossRef]
4. Tso, J.; Powers, J.; Kim, J. Cardiovascular considerations for scuba divers. *Heart* **2021**, *108*, 1084–1089. [CrossRef] [PubMed]
5. Yoon, S.; Qiao, C. Cooperative search and survey using autonomous underwater vehicles (AUVs). *IEEE Trans. Parallel Distrib. Syst.* **2010**, *22*, 364–379. [CrossRef]
6. Sivčev, S.; Coleman, J.; Omerdić, E.; Dooly, G.; Toal, D. Underwater manipulators: A review. *Ocean. Eng.* **2018**, *163*, 431–450. [CrossRef]
7. Ridao, P.; Carreras, M.; Ribas, D.; Sanz, P.J.; Oliver, G. Intervention AUVs: The next challenge. *Annu. Rev. Control* **2015**, *40*, 227–241. [CrossRef]
8. Mazzeo, A.; Aguzzi, J.; Calisti, M.; Canese, S.; Vecchi, F.; Stefanni, S.; Controzzi, M. Marine Robotics for Deep-Sea Specimen Collection: A Systematic Review of Underwater Grippers. *Sensors* **2022**, *22*, 648. [CrossRef] [PubMed]
9. Mazzeo, A.; Aguzzi, J.; Calisti, M.; Canese, S.; Angiolillo, M.; Allcock, A.L.; Vecchi, F.; Stefanni, S.; Controzzi, M. Marine Robotics for Deep-Sea Specimen Collection: A Taxonomy of Underwater Manipulative Actions. *Sensors* **2022**, *22*, 1471. [CrossRef] [PubMed]
10. López-Barajas, S.; Sanz, P.J.; Marín-Prades, R.; Echagüe, J.; Realpe, S. Network congestion control algorithm for image transmission—HRI and visual light communications of an autonomous underwater vehicle for intervention. *Future Internet* **2025**, *17*, 10. [CrossRef]
11. Pi, R.; Palomeras, N.; Carreras, M.; Sanz, P.J.; Oliver-Codina, G.; Ridao, P. OPTIHROV: Optically linked hybrid autonomous/remotely operated vehicle, beyond teleoperation in a new generation of underwater intervention vehicles. In Proceedings of the OCEANS 2023-Limerick, Limerick, Ireland, 5–8 June 2023.
12. Willners, J.S.; Carlucho, I.; Łuczyński, T.; Katagiri, S.; Lemoine, C.; Roe, J.; Stephens, D.; Xu, S.; Carreno, Y.; Pairet, È.; et al. From market-ready ROVs to low-cost AUVs. *arXiv* **2021**, arXiv:2108.05792.
13. Ge, Z.; Yang, F.; Lu, W.; Wei, P.; Ying, Y.; Peng, C. A Navigation System for ROV's Inspection on Fish Net Cage. *arXiv* **2025**, arXiv:2503.00482. [CrossRef]
14. López-Barajas, S.; Solis, A.; Marín-Prades, R.; Sanz, P.J. Towards Autonomous Coordination of Two I-AUVs in Submarine Pipeline Assembly. *J. Mar. Sci. Eng.* **2025**, *13*, 1490. [CrossRef]
15. Nguyen, D.T.; Elseth, C.L.; Øvstaas, J.R.; Arntzen, N.; Hamre, G.; Lillestøl, D.-B. Enabling scalable inspection of offshore mooring systems using cost-effective autonomous underwater drones. *Front. Robot. AI* **2025**, *12*, 1655242. [CrossRef] [PubMed]
16. Pi, R.; Cieślak, P.; Ridao, P.; Sanz, P.J. TWINBOT: Autonomous Underwater Cooperative Transportation. *IEEE Access* **2021**, *9*, 37668–37684. [CrossRef]
17. Sun, K.; Cui, W.; Chen, C. Review of Underwater Sensing Technologies and Applications. *Sensors* **2021**, *21*, 7849. [CrossRef] [PubMed]
18. Akram, W.; Ud Din, M.; Saad, A.; Hussain, I. AquaChat: An LLM-guided ROV framework for adaptive inspection of aquaculture net pens. *Aquac. Eng.* **2025**, *111*, 102607. [CrossRef]
19. Yang, R.; Hou, M.; Wang, J.; Zhang, F. OceanChat: Piloting Autonomous Underwater Vehicles in Natural Language. *arXiv* **2023**, arXiv:2309.16052. [CrossRef]
20. Khanam, R.; Hussain, M. YOLOv11: An Overview of the Key Architectural Enhancements. *arXiv* **2024**, arXiv:2410.17725. [CrossRef]
21. Pan, S.; Wang, X. A Survey on Perspective-n-Point Problem. In *Proceedings of the 40th Chinese Control Conference (CCC), Shanghai, China, 26–28 July 2021*; IEEE: Piscataway, NJ, USA, 2021; pp. 2396–2401. [CrossRef]
22. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
23. OpenAI. GPT-4.1 Mini Language Model. 2024. Available online: https://openai.com (accessed on 15 January 2026).
24. Blue Robotics. BlueROV2 Heavy Configuration Retrofit Kit. 2025. Available online: https://bluerobotics.com/store/rov/bluerov2-upgrade-kits/brov2-heavy-retrofit/ (accessed on 10 December 2025).
25. Blue Robotics. CAM-USB Low Light R1 Camera. 2025. Available online: https://bluerobotics.com/store/sensors-cameras/cameras/cam-usb-low-light-r1/ (accessed on 10 December 2025).

26. Blue Robotics. ExploreHD USB Deepwater Exploration Camera. 2025. Available online: https://bluerobotics.com/store/sensors-cameras/cameras/deepwater-exploration-explorehd-usb-camera/ (accessed on 10 December 2025).
27. Blue Robotics. Newton Gripper ASM-R2-RP. 2025. Available online: https://bluerobotics.com/store/thrusters/grippers/newton-gripper-asm-r2-rp/ (accessed on 10 December 2025).
28. Cieślak, P. Stonefish: An Advanced Open-Source Simulation Tool Designed for Marine Robotics, with a ROS Interface [Conference presentation]. In Proceedings of the OCEANS 2019 Conference, Marseille, France, 17–20 June 2019. [CrossRef]
29. ArduPilot Dev Team. SITL Simulator (Software in the Loop). 2024. Available online: https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html (accessed on 10 December 2025).
30. Cieślak, P. Stonefish: Open-Source Marine Robotics Simulator. 2025. Available online: https://stonefish-ros.readthedocs.io/en/latest/install.html (accessed on 10 June 2025).
31. Pymavlink Developers. pymavlink: Python MAVLink Library. 2025. Available online: https://github.com/ArduPilot/pymavlink (accessed on 10 December 2025).
32. OpenCV Histogram Equalization and Adaptive Histogram Equalization (CLAHE). 2025. Available online: https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html (accessed on 10 December 2025).
33. Perez, J.; Sanz, P.J.; Bryson, M.; Williams, S.B. A Benchmarking Study on Single Image Dehazing Techniques for Underwater Autonomous Vehicles. In Proceedings of the OCEANS 2017—Aberdeen, Aberdeen, UK, 19–22 June 2017; pp. 1–9. [CrossRef]
34. Marín, R.; Perez Edo, I.; Sanz, P.J.; López Barajas, S. Underwater Object Recovery Using a Hybrid-Controlled ROV with Deep Learning-Based Perception. *Zenodo* **2025**. [CrossRef]