A Generic Haptic Teleoperation Architecture Integrating Dobot X-Trainer Controllers Across YARP and ROS 2

Bartek Łukawski*[1], Weilun Ye[2], Edwin Daniel Oña[1], Juan G. Victores[1], Carlos Balaguer[1], Alberto Jardón[1], Qiang Li[2]

[1]*Departamento de Ingeniería de Sistemas y Automática, RoboticsLab*
Universidad Carlos III de Madrid, ROR: 03ths8210
Avenida de la Universidad 30, Leganés, Comunidad de Madrid, España, 28911
{blukawsk*, eona, jcgvicto, balaguer, ajardon}@ing.uc3m.es

[2]*Department of Computer Science and Technology, Arbeit Gruppe Dexterous Robotics (AGDR)*
Shenzhen Technology University, ROR: 04qzpec27
Lantian Road 3002, Pingshan District, Shenzhen, Guangdong, China, 518118
{2510263006@stumail.sztu.edu.cn, liqiang1@sztu.edu.cn}

***Abstract*—We extend our previously published hardware- and middleware- agnostic generic teleoperation architecture to integrate the Dobot X-Trainer input controllers, adding haptic feedback to the overall control scheme. YARP and ROS 2 middleware frameworks are leveraged to enable seamless cross-middleware interoperability across diverse robotic platforms and their simulated environments and models, such as those evaluated in this work: the TEO humanoid, the ABB CRB 15000-5 "GoFa" industrial cobot, and Dobot collaborative arms. Haptic feedback is achieved by converting external force-torque measurements into motor command adjustments on the position-controlled X-Trainer input hardware. The architecture's flexibility and performance are validated across simulated environments and physical systems.**

***Index Terms*—software architecture, manipulator arm, collaborative robotics, haptic teleoperation, ROS 2, YARP**

I. Introduction

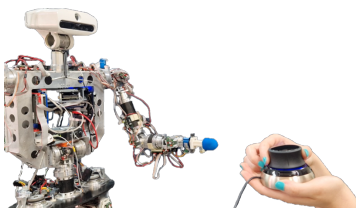
Our previous experience with teleoperation frameworks revealed that most robot manufacturers and open-source developers aim to target a single robot architecture, brand, or even individual model. It was our aim to fill this gap with a generic teleoperation framework which could be easily extended to cover specific hardware needs, either on the robotic platform side, or on the input device to be used to control it.

While investigating the addition of haptic feedback into this framework, the Dobot X-Trainer system came to our attention. However, we observed that the manufacturer's SDK does not offer ROS 2 nodes (or any other high-level interface) to directly access the device's encoder measurements nor its motor control, besides a low-level Python script to translate memory addresses into serial commands understood by the Dynamixel motors mounted on the device.

In this context, we decided to integrate the X-Trainer controllers into our open-source teleoperation architecture and explore its haptic capabilities.

II. Background

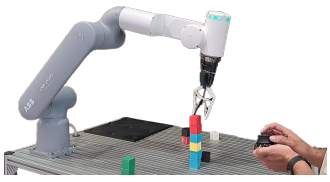
This work iterates on previous developments by RoboticsLab UC3M in the context of teleoperation software architectures. The foundations were laid in an earlier project introducing a generic controller for low-cost devices based on the YARP middleware, targeting several robotic platforms (real and simulated) and a collection of commercial-grade peripherals [1]. Building upon this, a Cartesian controller framework bridging YARP and the popular ROS 2 middleware was described, implemented and tested on two manipulator platforms [2] (see Figures 1a and 1b), and ultimately extended to a middleware- and hardware-agnostic architecture [3]. Haptic teleoperation with the Phantom Omni (rebranded as Phantom Touch) by 3D Systems was also explored [4].



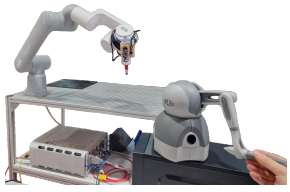
(a) Teleoperation of TEO [2]



(b) Teleoperation of AMOR [2]



(c) Teleoperation of ABB GoFa with a 6-axis 3D mouse [3]



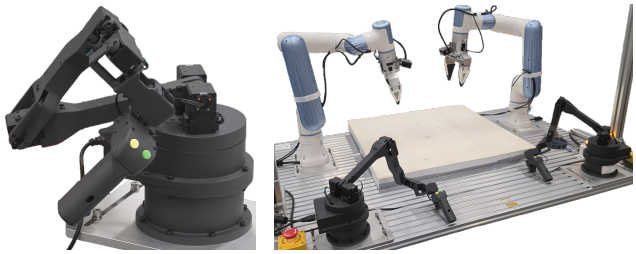
(d) Teleoperation of ABB GoFa with haptic feedback [4]

Fig. 1: Relevant previous works by RoboticsLab UC3M.

III. Materials and Tools

A. Dobot X-Trainer

The Dobot's X-Trainer system is depicted in Figure 2. It comprises two controllers based on Dynamixel actuators (Figure 2a) and two 6-axis collaborative robots (Figure 2b). Each controller features six joints and seven motors, with two motors actuating the second joint, together with two buttons and an analog trigger. All motors can be position-controlled and integrate an absolute encoder; therefore, depending on the state of the motors, the controllers can be operated in two ways: either as sensing devices providing joint measurements (motor torque off), or as haptic controllers that act upon an external signal (motor torque on).



(a) X-Trainer controller (b) X-Trainer AI full system rig

Fig. 2: Dobot X-Trainer AI data collection and training system.

B. Robot Platforms

Besides Dobot's collaborative robots, the TEO humanoid robot and ABB's CRB 15000-5 "GoFa" cobot, used in previous research, are also adopted in this work. TEO is a full-sized 28-DoF humanoid developed by RoboticsLab UC3M, featuring 6-DoF position-, velocity- and torque-controlled manipulators [5] (Figure 3a). On the other hand, the "GoFa" industrial collaborative robot is a commercial 6-axis arm developed by ABB, with intrinsic safety measures and support for kinesthetic teaching (Figure 3b). Custom or commercial tools, along with a force-torque sensor, can be installed on both robots, and simulator support is available: open-source Gazebo and OpenRAVE models for TEO, and a proprietary ABB RobotStudio environment and models for "GoFa".



(a) TEO humanoid robot (b) ABB CRB 15000-5 "GoFa"

Fig. 3: Robot platforms.

IV. Proposed Architecture

The proposed architecture for the teleoperation framework with an X-Trainer controller and haptic feedback is depicted in Figure 4. It is composed of a modular, distributed network of software building blocks leveraged by the YARP and ROS 2 middleware frameworks [6], [7]. YARP excels in platform-independent, low-level hardware access and motion control, whereas ROS 2, especially in its current, second iteration, has grown to a de facto standard in the robotics community, with much wider adoption and a rich open-source package ecosystem. It is therefore our choice to implement specific blocks adhering to YARP, which historically has been the pillar of RoboticsLab's codebase, while integrating ROS 2 bridges in order to gain interoperability with a higher variety of robots and controllers. The adoption of both middleware frameworks as the basis of our developments paves the way for a flexible and reusable framework that can be easily extended to different input devices, robot platforms (along with their associated virtual models and environments), and even a different middleware, as shown in our previous works.

The human operator and the robot (with an optional force-torque sensor to enable haptic feedback) are the main agents of the proposed architecture. A chain of interconnected components, such as device drivers for data acquisition and motion control, Cartesian and joint controllers, and the main application, connect both ends of the teleoperation control scheme. Some of those blocks (or "devices" in YARP terminology), responsible for computations or data acquisition, among others, are "wrapped" by other blocks solely devoted for opening a communication channel and exposing their programming interface and methods over the selected network, either YARP or ROS 2, using a server-client paradigm. These latter components are referred to as the *network wrapper server* (NWS) and *network wrapper client* (NWC).

Both network wrapper pairs and the exposed interface fulfill a specific goal. In this work, we present a device driver for the X-Trainer controllers (one instance per each) that exposes joint control through the motor interface (and associated network wrappers), while integrating with our generic teleoperation application through a joypad interface (and wrappers). The former allows querying encoder values and command the Dynamixel motors, whereas the latter binds the same encoder measurements in a more convenient C++ interface.

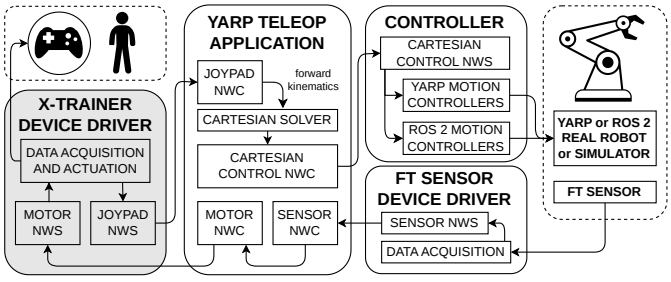


Fig. 4: Proposed teleoperation software architecture. The new component introduced in this work has been highlighted.

V. Implementation

A. Kinematic Description of the X-Trainer Controllers

In order to send Cartesian motion commands to the robot, the joint values across the X-Trainer controller's six axes need to be processed and converted to Cartesian poses through forward kinematics. A generic Cartesian solver device developed by us, based on the Orocos Kinematics and Dynamics Library (KDL), is used to load the kinematic description of the controllers and perform the calculations.

Table I depicts the Denavit-Hartenberg (DH) parameters (standard convention) of both left and right X-Trainer controllers (directions of rotation depend on the configuration stored in the ROM). Figure 5 depicts the coordinate frames associated with the Denavit-Hartenberg parameters, highlighting the base (S_0) and end-effector (S_6) frames.

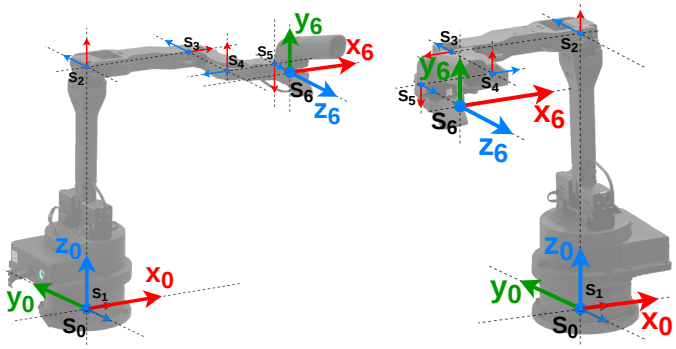
TABLE I: Denavit-Hartenberg parameters (distance in mm).

(a) Left controller

N	θ	D	A	α
1	0	0	0	$\frac{\pi}{2}$
2	$\frac{\pi}{2}$	0	196	π
3	$\frac{\pi}{2}$	0	154	0
4	$-\frac{\pi}{2}$	−83	0	$\frac{\pi}{2}$
5	π	−84	0	$-\frac{\pi}{2}$
6	$\frac{\pi}{2}$	31	0	0

(b) Right controller

N	θ	D	A	α
1	0	0	0	$\frac{\pi}{2}$
2	$\frac{\pi}{2}$	0	196	π
3	$-\frac{\pi}{2}$	0	154	0
4	$\frac{\pi}{2}$	−83	0	$-\frac{\pi}{2}$
5	π	−84	0	$\frac{\pi}{2}$
6	$\frac{\pi}{2}$	31	0	0



(a) Left controller (b) Right controller

Fig. 5: Coordinate frames associated to the DH parameters.

B. Teleoperation Control Scheme

Figure 6 depicts the control scheme adopted in the teleoperation framework with X-Trainer controllers and a force-torque sensor for haptic feedback. It leverages the network wrapper paradigm to introduce both YARP and ROS 2 elements, therefore gaining the necessary flexibility to conform to robot platforms that "speak" either of these middleware frameworks. In particular, the diagram is particularized for the YARP-based TEO humanoid and the ROS 2-based ABB "GoFa" cobot, for which a custom node was developed by us wrapping ABB's Externally Guided Motion (EGM) feature [8]. However, this generic scheme is also valid for other platforms (such as Dobot's arms integrated in the AI X-Trainer toolkit).

The main components of this scheme are:

1) The teleoperation application, connecting the X-Trainer device with the robot controllers. Communication with the former is performed through YARP's joypad network wrappers to acquire the latest encoder measurements. This information is passed to the Cartesian solver to perform forward kinematics and obtain the end-effector pose using the Denavit-Hartenberg parameters loaded during initialization. Depending on the robot, either YARP or ROS 2 network wrappers are used to send these coordinates to the Cartesian control device. On the other hand, force-torque measurements are received through the corresponding NWC and transformed to produce a feedback response, forwarded through a motor NWC to the X-Trainer's devices joint controller NWS.
2) For YARP-based platforms, a Cartesian control block that uses the kinematic description of the robot to perform inverse kinematics over the Cartesian pose sent by the main application, and forwards the result to the corresponding YARP joint controller.
3) A YARP-based joint controller implementing the motor interfaces supported by the robot or simulated environment (e.g., Gazebo, OpenRAVE).
4) The actuation, i.e., the real robot or simulator (e.g., TEO or the ABB "GoFa" cobot, or their virtual counterpart), and optional peripherals such as a force-torque sensor.
5) The newly developed X-Trainer data acquisition and motion control devices, which exposes both modalities through the corresponding YARP NWS devices and communicates through them with the main application.
6) For ROS 2-based platforms, a Cartesian controller mimicking YARP's interface with their topic and service equivalents in the ROS 2 ecosystem. For the ABB "GoFa" robot, this is our custom EGM-based node.
7) A data acquisition device for the force-torque sensor (if haptic feedback is expected) communicating the real device with the main application.
8) Particularly for the ABB "GoFa" cobot, the proprietary controller shipped for real robots and also embedded in the RobotStudio simulator. This component communicates with EGM servers through Google's Protobuf and UDP protocols.

VI. Experiments

A set of experiments has been conducted on three platforms: TEO, ABB "GoFa", and a Dobot's cobot arm. Due to limited hardware availability, simulated models have been used for TEO and "GoFa" (on Gazebo Sim and ABB RobotStudio environments, respectively), and the Dobot arms lacked a force-torque sensor. An auxiliary application was prepared to emulate the contact of the robot's end-effector against a virtual object, e.g., an easel, as shown in Figure 7a together with the "GoFa" and a drawing tool attached to it. Figure 7b depicts two 3D trajectories: the actual one followed by the robot's TCP, and another above it representing the measurements of a virtual sensor in the normal direction to the surface.

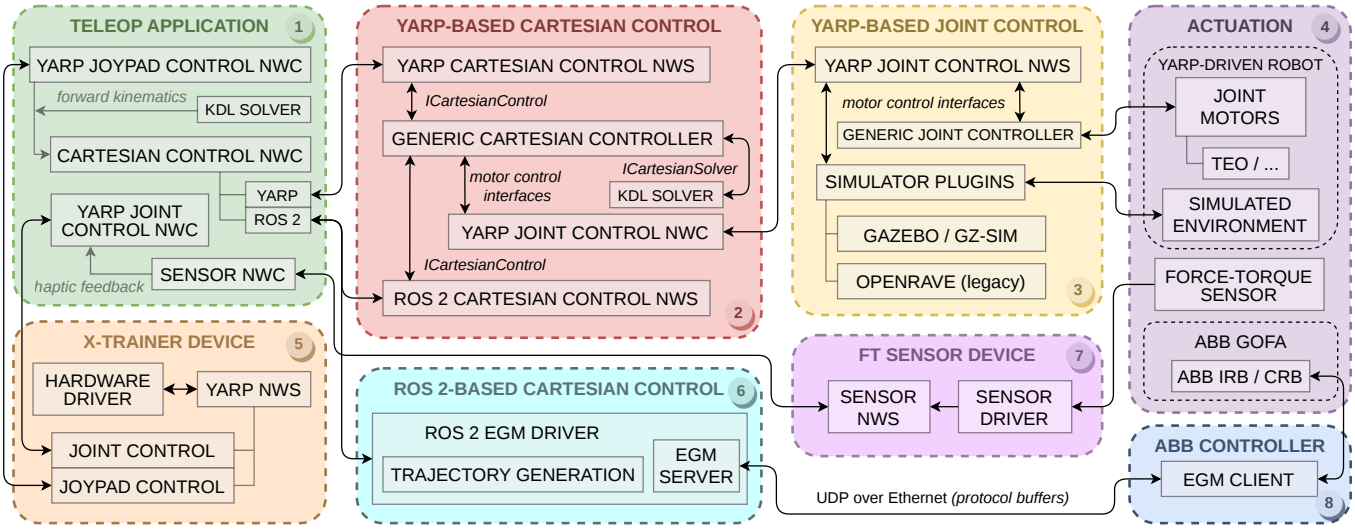
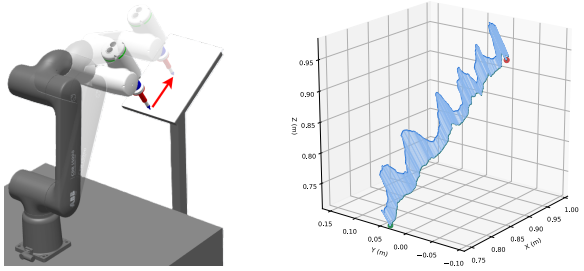


Fig. 6: Component diagram of the teleoperation application for YARP- and ROS 2-based robot platforms. Adapted from [3].



(a) Direction of motion

(b) TCP trajectory (below) and virtual sensor measurements (above), same perspective

Fig. 7: Teleoperation task with haptic feedback on the "GoFa".

Since the Dynamixel motors on the X-Trainer controllers are position-controlled, mode switches need to occur between torque on/off states, i.e., haptic feedback is enabled occasionally whenever contact is detected by the force-torque sensor. In order to avoid frequent mode switches, a configurable threshold was added to slightly extend the effect of the feedback, forcing the end-effector away from the detected obstacle along the surface normal at the point of contact, at a predefined distance. This repulsion has been observed to slightly undermine the manual performance of teleoperation tasks, yet to be precisely measured in future works.

VII. Conclusions

In this work, a generic teleoperation architecture, based on YARP and ROS 2 and leveraging our modular control scheme of distributed software components, was extended to permit haptic feedback in Dobot's X-Trainer controllers. Future work seeks to integrate a JR3 force-torque sensor in real setups [9]. The associated source code has been published on GitHub.[1]

[1]Main account: https://github.com/roboticslab-uc3m; relevant repositories: "kinematics-dynamics" (kinematics and dynamics solvers and applications), "teleop-yarp-devices" (YARP software components devoted to teleoperation), "rl_cartesian_controllers" (ROS 2 packages for Cartesian control and teleoperation), "abb_egm_driver" (EGM-based ROS 2 driver).

Acknowledgments

This research has been financed by "iRoboCity2030-CM, Robótica Inteligente para Ciudades Sostenibles" (TEC-2024/TEC-62) and "FotoArt5.0-CM, Laboratorios inteligentes para la ciencia del futuro" (TEC-2024/TEC-308), both funded by "Programas de Actividades I+D en Tecnologías de la Comunidad de Madrid"; "ROBOASSET: Intelligent robotic systems for assessment and rehabilitation in upper limb therapies" (PID2020-113508RB-I00, financed by AEI/10.13039/501100011033); "iREHAB: AI-powered Robotic Personalized Rehabilitation" (DTS22/00105, financed by Instituto de Salud Carlos III (ISCIII)); the Natural Science Foundation of Top Talent of SZTU (Grant No. GDRC202411); UC3M Mobility Grants (UC3M's Own Research Program); and EU structural funds.

References

[1] B. Łukawski, J. G. Victores, and C. Balaguer, "A generic controller for teleoperation on robotic manipulators using low-cost devices," Jornadas de Automática, CEA, 44, 2023. DOI: 10.17979/spudc.9788497498609.785

[2] B. Łukawski, M. Rebollo, Á. Gilabert, J. G. Victores, C. Balaguer, and A. Jardón, "YARP Cartesian controller layers over ROS 2 for teleoperation and web applications," Jornadas de Automática, CEA, 46, 2025. DOI: 10.17979/ja-cea.2025.46.12252

[3] B. Łukawski, E. D. Oña, J. G. Victores, C. Balaguer, and A. Jardón, "A YARP-driven generic Cartesian control architecture for hardware-agnostic teleoperation in ROS 2 ecosystems," Int. Conf. on Mechatronics, Robotics and Automation (ICMRA), IEEE, 2026. [in press]

[4] Á. Santos-García, B. Łukawski, D. Copaci, J. G. Victores, and C. Balaguer, "Bilateral haptic teleoperation bridging YARP and ROS 2 ecosystems," Simposios del Comité Español de Automática, CEA, 2(2), 2026. DOI: 10.64117/simposioscea.v2i2.216

[5] C. Pérez Martínez, P. Pierro, S. Martínez, L. Pabón, M. Arbulú, and C. Balaguer, "RH-2: an upgraded full-size humanoid platform," Mobile Robotics, World Scientific, 2009, pp. 471–478. DOI: 10.1142/9789814291279_0058

[6] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: Yet Another Robot Platform," Int. Journal of Advanced Robotic Systems, 3(1), 8, 2006, pp. 43–48. DOI: 10.5772/5761

[7] M. Quigley et al., "ROS: an open-source Robot Operating System," ICRA workshop on open source software, 3, 2009. Corpus ID: 6324125

[8] B. Łukawski, E. D. Oña, A. Jardón, J. G. Victores, and C. Balaguer, "Development of educational applications with ABB GoFa collaborative robot using Externally Guided Motion," Int. Conf. on Control, Automation and Diagnosis (ICCAD), IEEE, 2025. DOI: 10.1109/ICCAD64771.2025.11099395

[9] B. Łukawski, A. Olano-Díaz, A. González-Villasante, E. D. Oña, J. G. Victores, and A. Jardón, "External force-torque sensor feedback on the ABB GoFa collaborative robot," Simposios del Comité Español de Automática, CEA, 2(2), 2026. DOI: 10.64117/simposioscea.v2i2.215