Modular Software Framework for Upper-Body Telepresence and Control with Underactuated Robotic Hands via ROS 2 and YARP

Bartek Łukawski*[1], Miguel Alache-Manchado[2], Antonio Balboa-Parra[2],
Edwin Daniel Oña[1], Juan G. Victores[1], Carlos Balaguer[1], Alberto Jardón[1]
Departamento de Ingeniería de Sistemas y Automática, RoboticsLab
Universidad Carlos III de Madrid, ROR: 03ths8210
Avenida de la Universidad, 30 (edificio Sabatini), 28911 Leganés (Madrid), España
[1]{blukawsk*, eona, jcgvicto, balaguer, ajardon}@ing.uc3m.es
[2]{miguel.alache, a.balboa}@alumnos.uc3m.es

***Abstract*—Integrating underactuated anthropomorphic hands into complex teleoperation and telepresence systems often requires bridging heterogeneous middleware ecosystems and developing robot-agnostic control modules. In this paper, we present a modular software framework that orchestrates upper-body teleoperation and telepresence using underactuated hands and complementary vision systems. The architecture seamlessly coexists across ROS 2 and YARP, leveraging the rich ROS 2 ecosystem alongside established low-level YARP devices. As a core contribution, we introduce a generic control system featuring joint-coupling solvers and modular building blocks applicable to diverse robotic platforms. We integrate the low-cost Dextra underactuated hand and validate the architecture through both simulated (Gazebo) and real-world experiments on the TEO humanoid robot and ABB GoFa collaborative manipulators. Furthermore, we demonstrate the framework's versatility beyond teleoperation by executing an automated Rubik's cube solving task. All source code has been open-sourced to provide a foundation for future developments.**

***Index Terms*—software architecture, underactuation, humanoid robotics, collaborative robotics, teleoperation, middleware, ROS 2, YARP**

I. Introduction

Teleoperation systems enable human operators to perform manipulation tasks by mapping their own motion directly to the robotic platform. Underactuated anthropomorphic hands, compared to other gripper-like tools, benefit from a direct mapping to the user's hands while offering a wider range of grasping capabilities. However, seamlessly integrating a robot arm and such a hand, along with another arm-hand pair for bimanual operation, and a head with support vision system for live inspection, poses a challenge since existing control pipelines adhere to specific hardware or software constraints, thus hindering extensibility and component reuse. To address these limitations, this work presents a modular software framework that treats each component as a building block in a distributed architecture. This approach seeks to boost code portability across platforms, while also providing a basis upon which to build new applications with minimal hardware constraints and software compatibility issues.

II. Background

The present work iterates on previous developments by the authors framed within the RoboticsLab research group. The foundation of a modular teleoperation architecture was first laid out using YARP as the main orchestrating middleware framework. The system was validated on multiple robot platforms (TEO humanoid, ASIBOT and AMOR assistive manipulators) across different realities (real scenario, OpenRAVE simulator) and input peripherals (SpaceNavigator 3D mouse, WiiMote controller, Leap Motion sensor) [1]. A similar preliminary solution was adopted on the TIAGo robot and its Gazebo model using ROS and the 3D mouse [2], later expanded to the dual-arm TIAGo++ robot and a VR-based telepresence scenario using Meta Quest 2/3 integrated capabilities (Figure 1a) [3]. Critically, the Cartesian control component was later refactored to introduce a dual YARP-ROS 2 architecture bridging both networks [4]. This step paved the way to exploring the addition of haptic peripherals such as the Phantom Omni device, and its integration in the control framework of ABB industrial robots [5].

On a parallel line of research, the underactuated Dextra hand evolved from a design initially used to represent Spanish sign language with the humanoid TEO (Figure 1b) [6]. Underactuation was later modeled using neural networks [7].



(a) Teleoperation task with dual-arm TIAGO++ and Meta Quest 2 [3].

(b) TEO signing "L" in Spanish sign language using the Dextra hand [7].

Fig. 1: Relevant previous works by RoboticsLab UC3M.

Other authors have tackled the challenges covered in this work. While early YARP-ROS 1 interoperability relied on complex IDL (Interface Definition Language) conversion techniques [8], modern YARP versions interface directly with ROS 2 via its C++ client library (rclcpp) network wrappers. We adopted this approach in our work. Regarding input devices, study results favor vision-tracking methods over cumbersome wearables (such as gloves), which require greater effort from the user despite their precision [9]. Our architecture remains input-agnostic to accommodate any device based on hardware availability.

In terms of teleoperation frameworks, AnyTeleop (2023) introduced a vision-based generic system platform that outperforms alternatives designed for specific hardware architectures. It supports multiple arms manipulators, dexterous hands, realities (real robots and simulated models) and camera configurations (RGB, RGB-D). The source code was partly published [10]. Fully open-source alternatives followed, such as OpenTeach [11] and BEAVR [12], focusing on the Meta Quest virtual reality (VR) headsets as their core input and visualization device. Inaccuracies of Meta's proprietary hand detection and tracking were cited as the main shortcoming of the former, while the latter work acknowledges the hardware limitations of the LEAP hand regarding its precision, repeatability and bulkiness. Moreover, both projects depend on ROS 1 and Python as their programming interface.

Improving upon that, our work splits the internals of the teleoperation system into smaller components, and relies on ROS 2 for their orchestration. Similarly to the aforementioned works, ours is designed to provide a system that is agnostic of the robot and hand brands, supports any simulators able to understand ROS 2 or YARP communications, and handles bimanual scenarios. In addition, thanks to ROS 2 and YARP, we propose a distributed architecture that is not constrained to a single programming language, which combined with our open-source commitment provides developers with a versatile tool for their research and custom implementations. Lastly, we incorporate the robot's head in our control scheme to enable control of the robot's viewpoint during the task. We seek to extend this architecture to include locomotion and communication features to fully embrace telepresence scenarios [13].

III. Materials and Tools

A. Dextra Underactuated Anthropomorphic Robot Hand

The Dextra robotic hand project was originally submitted to Hackaday [14]. It features a low-cost and open-source hardware and software design based on 3D-printed modular components: five identical interchangeable fingers having two (thumb) or three phalanges each. The actuation relies on five DC motors with a high power-to-mass ratio, each pulling a tendon-like cable to bend each finger, while the retraction is achieved through elastic bands. The thumb features an additional servomotor to perform abduction movements. Since six actuated axes drive fifteen coupled physical joints, the mechanism is underactuated. The control is performed via computer interface or myoelectric (EMG) signals.

The design has been revised towards the integration of Dextra with GoFa, as depicted in Figure 2. Its accompanying software has been rebuilt to boost interoperability with the existing YARP and ROS 2 ecosystems of TEO and GoFa.

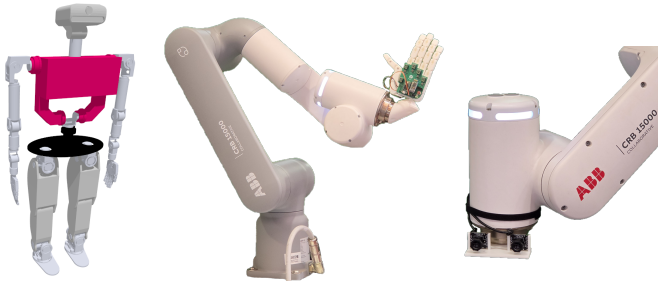


Fig. 2: Palm side of the left and right Dextra hands.

B. Robot Platforms and Tools

Two robotic platforms have been involved in this work. Firstly, the full-size TEO humanoid developed by the RoboticsLab research team features 28 degrees of freedom and multiple sensing capabilities, including an RGBD camera on its pan-tilt controlled head [15]. Figure 3a depicts the simulated TEO model in the Gazebo environment.

On the other hand, up to three units of the ABB CRB 15000-5 "GoFa" collaborative manipulator arm have been used in the teleoperation application, holding a different tool: right Dextra hand (Figure 3b), left Dextra hand, and a 3D-printed adapter with a dual low-cost webcam setup (Figure 3c).



(a) TEO (b) ABB GoFa (c) Dual camera system

Fig. 3: Robot platforms.

Interoperation and communication between software components and systems rely on the YARP and ROS 2 middlewares. YARP serves as the backbone of TEO's low-level control stack, leveraging its plugin-based approach to adopt a distributed architecture of smaller software building blocks ("devices"), either custom or developed by the community, for robot motion control and perception, among other goals [16]. A thin abstraction layer enables exposing existing YARP devices through the ROS 2 network and its richer ecosystem. A ROS 2 controller has been developed for GoFa wrapping the proprietary Externally Guided Motion (EGM) feature, allowing high-frequency (250 Hz) external command references and robot state publishing [17].

IV. Proposed Architecture

The software architecture illustrated in Figure 4 expands on a previous work involving the robot TIAGo by PAL Robotics and a virtual reality (VR) headset in a telepresence context [3]. This work describes a distributed network of reusable components orchestrated by YARP and ROS 2. The core contribution is the integration of multiple robots and environments within these frameworks to partake in a multi-modal teleoperation task. Since each component acts as a building block, including the robots and peripherals, the proposed control scheme naturally generalizes across different contexts. It would be straightforward to use a different robot by providing its YARP and/or ROS 2 interfaces, or to introduce new input devices (e.g. different sensors and joysticks).

The human operator and the robot platforms are the main actors of the task. Regardless of the robot, its capabilities are matched against the ones of the VR headset. For instance, the vision system of the robot (e.g. a single or dual RGB webcam setup) would stream frames to the display of the headset. Similarly, any speaker or microphone installed on the robot can stream to or from the built-in headset's microphone or speaker, respectively.

The application on the headset communicates with an external control PC through a bridge capable of interfacing with the ROS 2 network.[1][2] The VR system grants access to the relevant motion parameters of the user's hand, arm and head, to be translated as Cartesian commands to the robot.

Said commands are decoded on the control PC by the endpoint and exposed over ROS 2 topics to the Cartesian controller. In a generic YARP-driven robot, this architecture proposes an intermediate joint controller layer for interfacing with the real or simulated platform. Regarding ABB robots with EGM capabilities, a separate Cartesian and joint control layer has been developed.

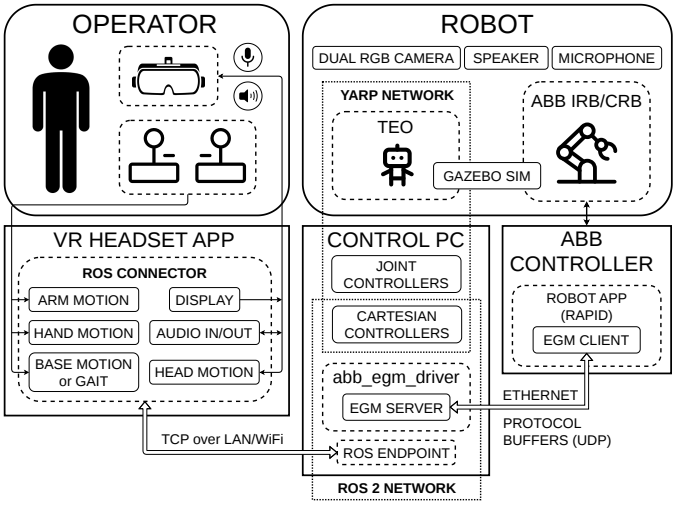


Fig. 4: Diagram of the proposed architecture.

[1]https://github.com/Unity-Technologies/ROS-TCP-Connector
[2]https://github.com/Unity-Technologies/ROS-TCP-Endpoint

V. Implementation

A. Underactuation and Kinematics of the Dextra Fingers

Each Dextra finger is actuated by a compact DC motor that winds a wire (e.g. a fishing line) to bend the two or three phalanges. Consequently, each actuated axis drives two or three physical joints. In previous studies, it was proved that these mappings roughly obey a piecewise linear function [7].

The code implementation accompanying this work features a generic tool to translate between physical joints and actuated axes. Certain assumptions were made to cater to the specific Dextra context, such as 1-to-N non-interleaved mapping, i.e., each physical joint is coupled with exactly one actuated axis, while each actuated axis can be coupled with one or more physical joints. A bijective relation is therefore assumed between both spaces. The logic is shown in Algorithm 1. The θ_j input represents a vector of physical joint values, while θ_m stores the resulting actuated (motor) axis setpoints.

Algorithm 1: Mapping physical joints to actuated axes.

Function `physical_to_actuated(`$\boldsymbol{\theta_j}$`)`**:**
- $\boldsymbol{\theta_m} \leftarrow$ resize($n_actuated_axes$, 0.0)
- **for** $index_m$ in range($n_actuated_axes$) **do**
  - $items \leftarrow actuated_to_physical_map[index_m]$
  - $min_value \leftarrow \infty$
  - **for** $index_j$, $transformation$ in $items$ **do**
    - $pos \leftarrow \boldsymbol{\theta_j}[index_j]$
    - $value \leftarrow transformation.position(pos)$
    - **if** $value < min_value$ **then**
      - $min_value \leftarrow value$
    - **end**
  - **end**
  - $\boldsymbol{\theta_m} \leftarrow min_value$
- **end**
- **return** $\boldsymbol{\theta_m}$

The motion of a three-phalanx Dextra finger is illustrated in Figure 6a. In addition to solving the underactuation problem, inverse kinematics must be performed to map from the user input coordinates in the Cartesian space to the joint space of the fingers. Traditional numeric solvers are avoided to prevent convergence issues; instead, a product of exponentials is expressed (see Equation 1 and Figure 6b) following the purely geometric screw theory approach [18]. Solutions are approximated if they cannot be computed exactly.

$$e^{\hat{\xi_1}\,\theta_1}\, e^{\hat{\xi_2}\theta_2}\, e^{\hat{\xi_3}\theta_3}\, {}^{o}H_p = H(\vec{\theta}) \tag{1}$$

A custom solver was used to apply the correct sequence of Paden-Kahan (PK) or Pardos-Gotor (PG) subproblems [19]:

1) PG4: rotation about ω_2, and then ω_1, applied to "c".
2) PK1: rotation about ω_3 applied to "o".

Although two solutions are available, only one is feasible since the finger cannot bend outwards. The kinematics of the thumb only involve one PG4 step: rotation about ω_2, and then ω_1, applied to "o".

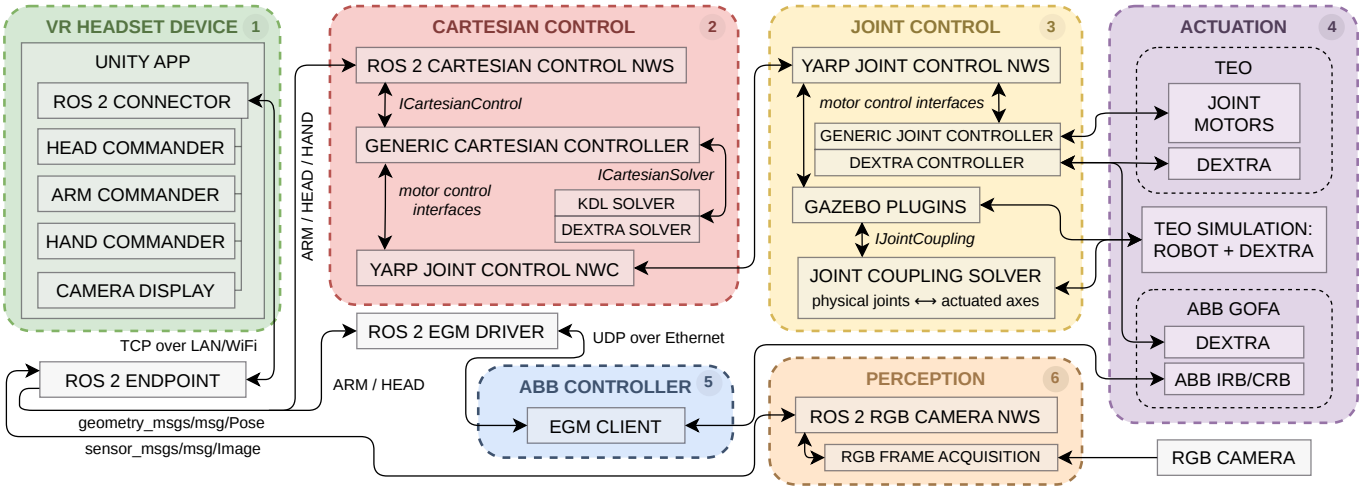
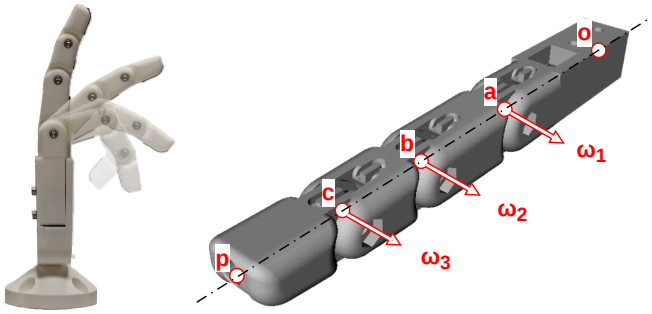


Fig. 5: Component diagram of the VR-based teleoperation application on TEO and GoFa.



(a) Finger motion (b) Screw axes and characteristic points

Fig. 6: Kinematics of a three-phalanx Dextra finger.

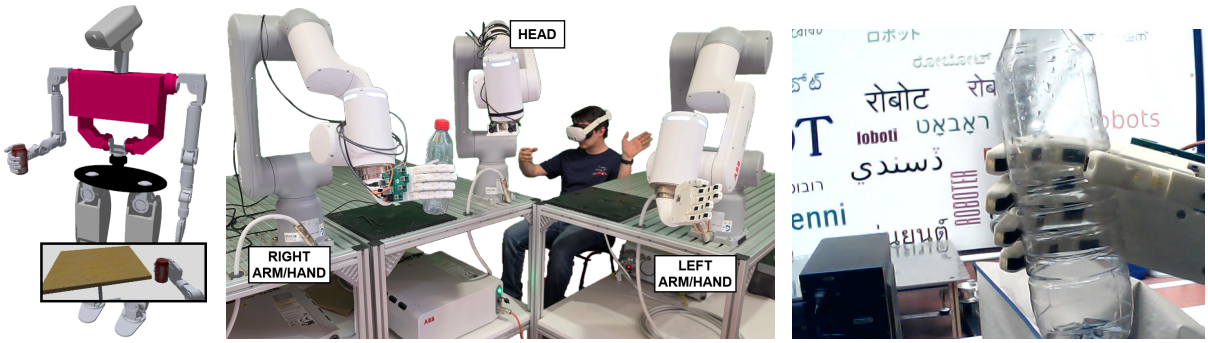
B. Teleoperation of TEO and GoFa using a VR Headset

The architecture described earlier has materialized in a teleoperation application for a Meta Quest 2/3/3S VR headset and the robot platforms TEO and GoFa. Earlier works have paved the way for the necessary YARP-ROS 2 abstraction layers and communication bridges [4]. This highly modular approach empowers component reuse, maintainability and future enhancements.

Figure 5 breaks down the building blocks of the application and their interconnections:

1) The VR application executed on the headset interfaces with the ROS 2 network through the connector layer, which acts as a TCP-based proxy of the usual ROS client libraries, e.g., its API offers topic subscription, publishing, ROS types, etc. The underlying Unity environment is accessed for dynamic capture of the pose and motion of the headset through its integrated inertial sensors, and the user's hands by means of proprietary vision algorithms. These are later translated into motions of the robot's head, and its arms and hands, respectively. Frames grabbed by the robot's cameras are shown in the headset display for real-time visual feedback of the task.
2) ROS 2 messages are routed by the endpoint to subscriber nodes such as the Cartesian controller. YARP uses client-server device pairs to abstract the communications from the actual implementation. In this case, a thin communications layer processes incoming ROS 2 messages (of the "geometry_msgs/Pose" type) through a network wrapper server (NWS) device, and lets them through to the underlying generic Cartesian controller adhering to a custom C++ interface. Kinematic computations are performed by a separate solver device, accessed by the controller through composition. The resulting joint variables are handled by the client side (NWC) of the joint controller's network wrapper.
3) A generic joint controller adhering to standard YARP C++ interfaces processes incoming commands and converts them to motion on the real robot. A specific Dextra controller was added to account for its hardware specifics. For the simulated TEO, Gazebo plugins are interfaced with, where an additional joint coupling layer was added to perform the necessary actuated-to-physical conversions on the virtual Dextra model.
4) The final recipients of the motion commands are any generic robot adhering to TEO's control scheme, or any ABB IRB/CRB robot compatible with EGM.
5) The ABB controller interfaces with a custom ROS 2 node to translate EGM messages into motion commands.
6) Color frames from the robot's RGB camera(s) are exposed over the ROS 2 network through a dedicated NWS device and the "sensor_msgs/Image" ROS message type.

The main advantage of this architecture is modularity: the controller is broken down into its basic building blocks and orchestrated by the middleware that best suits the environment at hand. Even though TEO has been driven by YARP due to historic reasons, the inclusion of a ROS-based robot (ABB GoFa) proves that this scheme can be extrapolated to different robotic systems and peripherals. For instance, the VR headset can be replaced by a hand-tracking Leap Motion device.



(a) Simulated experiment (b) Components and their layout in the real scenario (c) Viewpoint from the head's vision system

Fig. 7: Teleoperation experiment on the simulated model of TEO and three real GoFa robots.

C. Rubik's Cube Solver

An additional application has been developed to prove that the proposed architecture can be extrapolated to other tasks beyond teleoperation. The GoFa robot, a Dextra hand attached to it, and a fixed RGB camera have been combined to automate solving the Rubik's cube through initial visual inspection, then computing the solution, and finally sending a sequence of predefined motion commands to the robot and hand. Communications are orchestrated entirely by ROS 2, while the low-level hardware implementation of the Dextra relies on the same YARP-compliant joint controller as described earlier. The underlying cube solver implements the two-phase algorithm described by Kociemba et al. [20]. It computes solutions within milliseconds, making it suitable for constrained embedded platforms such as the Raspberry Pi 3.

Figure 8 depicts the application logic. The high-level controller interfaces with other components through ROS 2 topics. RGB frames acquired from the camera are used as input. This block also sends messages storing float arrays to the Dextra controller and the EGM driver. The former accepts joint motion commands, whereas the latter receives a sequence of steps resulting from the two-phase solver, subsequently used by a RAPID program running on the robot's controller to command the GoFa involving predefined motions.

The execution flow starts with the visual inspection of the cube. The robot grabs it and holds it in front of the camera while a visual recognition algorithm builds and stores the initial configuration of the 54 facelets in program memory. The cube is rotated accordingly so that all faces are inspected by the camera. The Kociemba's algorithm[3] is run to find a solution, usually in less than 20 steps, and encodes the result in a string describing the rotations to be applied to the cube. This string is sent to the RAPID program to perform the arm movements. Simultaneously, actuation commands are sent to the Dextra controller when necessary to hold or release the cube. Only the upper level of facelets is rotated relative to the remaining levels. In order to properly position the cube, it can be grasped from any side and rotated either 90 or 180 degrees.

[3]https://github.com/hkociemba/RubiksCube-TwophaseSolver

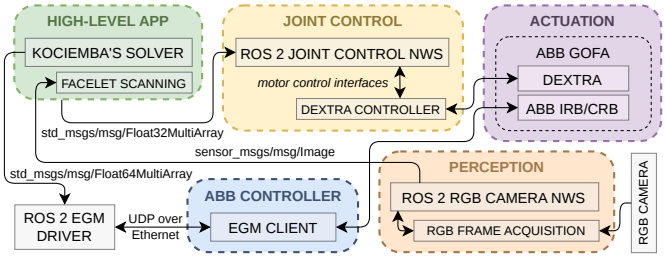


Fig. 8: Component diagram of the Rubik's cube solver app.

VI. Experiments

A. Teleoperation With Simulated TEO and Real GoFa Arms

A first experiment was carried out with the simulated TEO robot in a Gazebo environment and the virtual Dextra hands, as shown in Figure 7a. A monocular RGB camera sensor on the robot's head streams to the VR headset, allowing the operator to inspect the scene from the robot's perspective. The motion of the operator's head, arms and hand is replicated by the robot model. In this scenario, the goal of the operator is to use the robot's right hand to grab a can from a table placed in front of it, while also using the head's degrees of freedom to inspect the surroundings. The proposed implementation has been successfully validated, although suboptimal contacts between the hand and the can hindered stable grasping, resulting in frequent object drops.

A second scenario was prepared using three GoFa robots in a real environment, as shown in Figure 7b. The left and right Dextra hands and a dual-webcam system are mounted on the robots, mimicking the upper torso of a human: left and right arms and hands, and a central head. In a similar fashion to the simulated scenario, the motions of the operator are replicated by the three robots following said layout, and the goal is to grab an object placed in front of them. Thanks to the dual-camera system, each webcam streams frames to each operator's eye separately, therefore the distance between the Dextra and the target objects can be estimated through the acquired depth perception. Figure 7c depicts the viewpoint of one of these camera sensors.

B. Solving the Rubik's Cube With a GoFa Robot

The Rubik's cube solver has been validated on a real setup with a single GoFa robot, as described earlier. A 3D-printed support platform had to be introduced in order to facilitate the grasping of the cube, which is achieved with just the index and thumb fingers. Due to the amount of side switching movements, it takes about 6 minutes for the application to solve the cube once the initial scanning phase is completed. Figure 9 depicts the setup and three stages of the execution.

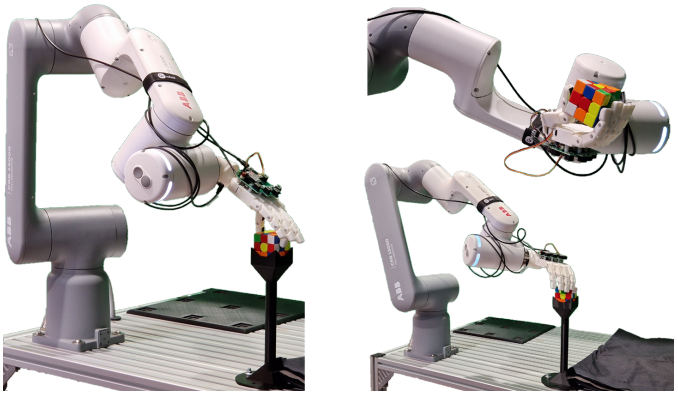


Fig. 9: Solving a Rubik's cube with the GoFa+Dextra setup.

VII. Conclusions

This work presents a modular software architecture that leverages YARP and ROS 2 middleware frameworks in teleoperation scenarios for varied robotic platforms, such as the TEO humanoid and ABB GoFa collaborative arms. Certain components have been specifically developed for this project: a generic joint coupling solver, a ROS 2 driver for EGM, and the high-level control applications. Thanks to YARP and ROS 2, integration with existing components and control schemes has been seamless, thus providing a reusable scheme for future robotic projects. To prove its suitability for other tasks, a Rubik's cube solver application has been also developed, showcasing the manipulability of the anthropomorphic Dextra hands. Experiments have been carried out to validate multiple setups both in simulated and real environments.

Haptic feedback is considered for future enhancements. The source code is publicly available under a free license.[4]

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)); and EU structural funds.

[4]Public account: https://github.com/roboticslab-uc3m; relevant repositories: "dextra" (3D CAD models, PCB files, firmware)", "dextra-yarp-devices" (Dextra-related logic and simulator models), "tools" (generic joint coupling solver), "abb_egm_driver" (EGM-based ROS 2 driver), "dextra-unity-vr" (VR-based teleoperation app), and "gofa-rubik-solver" (Rubik's cube solver app).

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] A. Calzada-García, B. Łukawski, J. G. Victores, and C. Balaguer, "Teleoperation of the robot TIAGo with a 3D mouse controller," Simposio de Robótica, Bioingeniería y Visión por Computador, CEA, 2024. URL: http://hdl.handle.net/10662/21260

[3] B. Łukawski, I. Montesino, E. D. Oña, J. G. Victores, C. Balaguer, and A. Jardón, "Towards the development of telepresence applications with TIAGo and TIAGo++ using a virtual reality headset," Int. Conf. on Autonomous Robot Systems and Competitions (ICARSC), IEEE, 2025. DOI: 10.1109/ICARSC65809.2025.10970173

[4] 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

[5] 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

[6] J. J. Gago, J. G. Victores, and C. Balaguer, "Sign language representation by TEO humanoid robot: End-user interest, comprehension and satisfaction," Electronics, 8(1), 57, 2019. DOI: 10.3390/electronics8010057

[7] J. J. Gago, B. Łukawski, J. G. Victores, and C. Balaguer, "Under-actuation modelling in robotic hands via neural networks for sign language representation with end-user validation," Lecture Notes in Computer Science, Springer Nature, pp. 239–251, 2020. DOI: 10.1007/978-3-030-62365-4_23

[8] M. Aragão, P. Moreno, and A. Bernardino, "Middleware interoperability for robotics: A ROS–YARP framework," Frontiers in Robotics and AI, 3, 2016. DOI: 10.3389/frobt.2016.00064

[9] J. Fu et al., "Teleoperation control of an underactuated bionic hand: Comparison between wearable and vision-tracking-based methods," Robotics 11(3), 61, 2022. DOI: 10.3390/robotics11030061

[10] Y. Qin et al., "AnyTeleop: A general vision-based dexterous robot arm-hand teleoperation system," Robotics: Science and Systems, 2023. DOI: 10.15607/RSS.2023.XIX.015

[11] A. Iyer et al., "Open Teach: A versatile teleoperation system for robotic manipulation," 2024. DOI: 10.48550/arXiv.2403.07870

[12] A. Posadas-Nava, A. Carrasco, and R. Linares, "BEAVR: Bimanual, multi-embodiment, accessible, virtual reality teleoperation system for robots," Int. Conf. on Control and Robotics (ICCR), IEEE, 2025. DOI: 10.1109/ICCR67607.2025.11372114

[13] C. Lenz et al., "NimbRo wins ANA Avatar XPRIZE immersive telepresence competition: Human-centric evaluation and lessons learned," Int. Journal of Social Robotics, Springer Nature, 17(3), 2025, pp. 337–361. DOI: 10.1007/s12369-023-01050-9

[14] A. Villoslada, "Dextra: open-source myoelectric hand prosthesis," Hackaday, 2016. URL: https://hackaday.io/project/9890-dextra.

[15] 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

[16] 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

[17] 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

[18] J. M. Pardos-Gotor, "Screw theory in robotics: An illustrated and practicable introduction to modern mechanics," CRC Press, 2021. DOI: 10.1201/9781003216858

[19] B. Łukawski, I. Montesino, J. G. Victores, A. Jardón, and C. Balaguer, "An inverse kinematics problem solver based on screw theory for manipulator arms," Jornadas de Automática, CEA, 43, 2022. DOI: 10.17979/spudc.9788497498418.0864

[20] T. Rokicki, H. Kociemba, M. Davidson, and J. Dethridge, "The diameter of the Rubik's cube group is twenty," SIAM Review, 56(4), 2014, pp. 645–670. DOI: 10.1137/140973499