CEA Comité Español de Automática
XLVII JORNADAS DE AUTOMÁTICA CÓRDOBA 2026
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

YARP and ROS 2 GenICam middleware for visual servoing applications

Álvaro Santos-García*, Bartek Łukawski, Juan G. Victores, Carlos Balaguer, Alberto Jardón

RoboticsLab, Department of Systems Engineering and Automation, Universidad Carlos III de Madrid, Avda. Universidad, 30, 28911, Leganés, Spain.

To cite this article: Santos-García, Á., Łukawski, B., Victores, J. G., Balaguer, C., Jardón, A. 2026. YARP and ROS 2 GenICam middleware for visual servoing applications.
Jornadas de Automática, 47. https://doi.org/10.17979/ja-cea.2026.47.13836

Resumen

Los robots de servicio y humanoides dependen de sistemas de percepción visual fiables, habitualmente basados en cámaras industriales con protocolos de conexión heterogéneos como USB3 Vision y GigE Vision. Este trabajo presenta un middleware de adquisición para cámaras compatibles con el estándar GenICam, implementado como un dispositivo de YARP en C++ sobre la librería Aravis. El dispositivo expone las interfaces estandarizadas de YARP para la captura de imágenes y el control de parámetros, como la exposición o la ganancia, publicándolas simultáneamente sobre las redes de YARP y ROS 2 mediante los correspondientes adaptadores (*wrappers*). Esta doble exposición permite reutilizar la misma abstracción hardware en dos plataformas con ecosistemas distintos: el robot humanoide TEO, integrado nativamente en YARP, y el manipulador móvil TIAGo, basado en ROS 2. Este va acompañado de una interfaz gráfica de control y una demostración de visual servoing guiada por marcadores ArUco.

Palabras clave: Visual servoing, Tecnologías robóticas, Percepción y sensorización, Visión por computador, Robots humanoides, Arquitecturas software para robótica

Abstract

Service and humanoid robots rely on dependable visual perception systems, commonly built around industrial cameras that expose heterogeneous connectivity protocols such as USB3 Vision and GigE Vision. This work presents an acquisition middleware for cameras compliant with the GenICam standard, implemented as a YARP device written in C++ on top of the Aravis library. The device exports the standardized YARP interfaces for image acquisition and parameter control, such as exposure or gain, publishing them simultaneously over the YARP and ROS 2 networks through the corresponding network wrappers. This dual exposure allows the same hardware abstraction to be reused on two platforms with distinct ecosystems: the TEO humanoid robot, natively integrated in YARP, and the TIAGo mobile manipulator, based on ROS 2. This was accompanied by a graphical control interface and a demonstration of visual servoing guided by ArUco markers.

Keywords: Visual servoing, Robotics technology, Perception and sensing, Computer vision, Humanoid robots, Robot software architectures

1. Introduction

Visual perception is a critical capability for autonomous robots, supporting tasks such as object recognition, navigation, manipulation, and human-robot interaction. This perception is often delegated to industrial cameras, preferred over consumer devices for their robustness, resolution and fine control over acquisition parameters. Their integration is complicated, however, as vendors expose proprietary software development kits and devices implement different physical interfaces and transport protocols, so the resulting glue code is rarely portable between robots. The GenICam standard provides an abstraction layer to discover and configure features of the underlying interface, but it does not solve the problem

*Corresponding author: alvarosang2003@gmail.com

at the system level, since middlewares such as YARP (Metta et al., 2006) and ROS 2 (Quigley et al., 2009) still require device drivers to bridge cameras into their data-flow models. YARP is widely adopted for its real-time services, whereas ROS 2 dominates the broader robotics community.

This work presents an acquisition middleware targeting both ecosystems, built around a YARP device named AravisGigE that wraps the open-source Aravis library to interface GenICam cameras over USB3 Vision and GigE Vision. It exports the standard YARP frame grabber interfaces and is published simultaneously over the YARP and ROS 2 networks, so a single hardware abstraction feeds two robots with very different stacks; the TEO humanoid, consumed natively through YARP, and the TIAGo mobile manipulator, through ROS 2. The contributions are a generic GenICam device for YARP supporting dual exposure through network wrappers, an extension to the YARP Python bindings enabling feature-range queries, and a graphical control interface with a visual servoing demonstration.

Finally, this document is organized as follows. Section 2 reviews the related work; Section 3 introduces the hardware and software used; Section 4 details the proposed system and its implementation; Section 5 presents the experimental validation on both platforms; and Section 6 gathers the conclusions and future work.

2. Background

The trend is towards standardization in industrial cameras, with the GigE Vision and USB3 Vision transport standards and the GenICam generic interface for configuration (Automated Imaging Association (AIA), 2026); on top of these, open implementations such as Aravis provide a transport and control layer for GigE Vision and USB3 Vision cameras, and general-purpose libraries such as OpenCV expose GenICam backends for image processing. Within the ROS ecosystem, drivers such as the GenICam ROS driver and OpenCV integrations relying on Aravis or PvAPI backends provide access to GenICam-compliant cameras, but tend to focus on image acquisition and offer only partial run-time control over camera parameters, with dynamic reconfiguration usually requiring command-line tools or edits to configuration files. Within the YARP ecosystem, the iCub humanoid platform follows a similar pattern through its yarpdev module, which publishes images from a variety of cameras into YARP ports but does not provide a unified interface for configuring industrial cameras from different vendors and protocols, and visualization tools such as yarpview or RViz cover the monitoring side without addressing dynamic parameter reconfiguration.

YARP follows a distributed architecture based on modular components called devices, which encapsulate hardware functionalities behind standardized C++ interfaces and can be reached either through a direct in-process connection or through a client and wrapper pair that exposes the device over the network (Metta et al., 2006). The platform also provides a name service, support for multiple transport carriers and a resource finder that locates configuration files at run time, all of which favour deployments distributed across several machines. For image acquisition, YARP defines the IFrameGrabberImage and IFrameGrabberImageRaw interfaces, and for parameter control, the IFrameGrabberControls interface, so that heterogeneous cameras are treated uniformly and interchangeably.

ROS 2, in turn, structures distributed applications into independent processes called nodes that exchange typed messages over topics and services, with camera streams conventionally transported as sensor_msgs/Image accompanied by sensor_msgs/CameraInfo. Bridging the YARP and ROS 2 ecosystems enables the reuse of existing YARP device interfaces alongside modern ROS 2 components, an approach previously explored within our group for generic teleoperation controllers and screw-theory inverse kinematics (Łukawski et al., 2023, 2022). The present work applies the same philosophy to the sensing side, exposing a single GenICam acquisition device over both networks.

3. Tools and materials

3.1. Vision sensors

The primary sensor used for development and testing is a FLIR (formerly Point Grey) Flea3 FL3-U3-88S2C-C USB 3.0 camera with a progressive scan CMOS sensor of 8.8 MP (FLIR Integrated Imaging Solutions Inc., 2012), compatible with USB3 Vision and the GenICam standard, which offers detailed control of acquisition parameters such as exposure and gain and a high resolution suitable for medium- and long-range vision tasks. To exercise the GigE Vision branch of the device, a second GenICam-compliant color camera with a Gigabit Ethernet interface is used (The Imaging Source DFK Z12G445). Both sensors are shown in Figure 1, the USB3 Vision camera in Figure 1(a) and the GigE Vision camera in Figure 1(b).

(a) FLEA3 camera (b) GigE Vision camera

Figure 1: Camera sensors.

3.2. TEO humanoid robot

TEO, standing for Task Environment Operator, is a full-sized humanoid robot platform developed by the RoboticsLab group at the Department of Systems Engineering and Automation of Universidad Carlos III de Madrid (Pérez Martínez et al., 2010). The platform traces back to the RH-0 robot, initiated in 2002 and upgraded to RH-1 in 2005, before evolving into its current version, RH-2, which introduced improvements in energy efficiency, control precision and sensor integration flexibility. TEO distributes 28 degrees of freedom across its limbs, torso and neck, with 6 DoF per arm, 6 DoF per leg, 2 DoF in the torso and 2 DoF in the neck, and has served as a research platform for bipedal locomotion and human-robot interaction.

Its sensorization includes an intelligent head equipped with a Kinect RGB-D camera alongside industrial sensors, an inertial measurement unit and four force-torque sensors, historically used to generate 3D environment models for domestic manipulation tasks such as folding or ironing clothes.

Its control stack is built natively on YARP, with Cartesian and joint level controllers exposed as YARP devices, including ICartesianControl, IPositionControl and IControlMode interfaces backed by a KDL-based inverse kinematics solver, that command both arms and the head. Module parameters, such as port names and device paths, are resolved at run time through the YARP ResourceFinder from dedicated configuration files. An overview of the robot is shown in Figure 2(a) and a detailed view of its head in Figure 2(b), together composing Figure 2.

(a) TEO humanoid robot (b) Detail of TEO's head and cameras

Figure 2: UC3M RoboticsLab's TEO full-body humanoid robot.

3.3. TIAGo dual-arm mobile manipulator

TIAGo is a dual-arm mobile manipulator developed by PAL Robotics and widely used in service-robotics research. It combines an omnidirectional mobile base, a lifting torso, a pan-tilt head and two 7-DoF arms, and is physically available at the RoboticsLab group of Universidad Carlos III de Madrid. Its software ecosystem is fully built on ROS, which makes it the natural platform to validate the ROS 2 acquisition path of this work. In this setup, the GenICam camera is treated as an additional head-mounted sensor whose image stream is consumed by the robot's ROS 2 perception pipeline, demonstrating that the same low-level driver can feed a ROS-native platform without modifications to the core implementation. The robot is shown in Figure 3(a) and a detail of its head-mounted cameras in Figure 3(b), together composing Figure 3.

(a) TIAGo (b) Detail of TIAGo's head-mounted cameras

Figure 3: PAL Robotics' TIAGo dual-arm mobile manipulator robot.

4. Architecture

The proposed system separates hardware abstraction, network exposure and application logic, as shown in Figure 4. At the lowest level, a YARP device named AravisGigE interfaces the physical camera through the Aravis library and exports the standard YARP frame grabber interfaces for image acquisition (IFrameGrabberImage and IFrameGrabberImageRaw) and parameter control (IFrameGrabberControls). On startup, the device discovers compatible cameras and configures a continuous acquisition stream with the hardware trigger disabled. This continuous, free-running mode simplifies single-camera acquisition but does not address temporal synchronization across multiple cameras, which would require coordinating hardware triggers and is left for future work. After this, frames are retrieved through a timeout-based buffer pop; this timeout, set to 200 ms, was chosen to tolerate the packet loss and latency variability typical of GigE Vision transport without stalling the acquisition thread. The two acquisition interfaces are routed differently: IFrameGrabberImageRaw returns the sensor data as a grayscale image, while IFrameGrabberImage delivers an RGB image, performing a pixel-format conversion with OpenCV (for example: Bayer to BGR) only when the format requested by the client differs from the one produced by the sensor, and passing the buffer through untouched otherwise. To support development, the device also offers a fake acquisition mode, activated by a launch flag, that synthesizes a moving grayscale gradient at the requested resolution, so downstream modules and the graphical interface can be validated independently of the sensor availability.

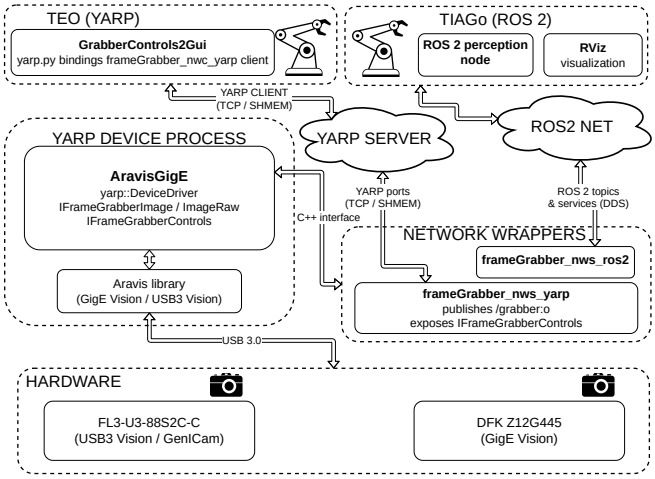

Figure 4: Proposed system architecture.

Feature control is handled generically through the GenICam node tree, tracking integer and float features in separate maps so that values are converted safely at run time and any automatic regulation mode is disabled before a manual write. Initialization parameters are handled via the YARP ResourceFinder, which parses standard `.ini` files for camera discovery and network configuration.

The device also includes an optional interactive terminal that exposes a read, evaluate and print loop for run-time calibration and diagnostics without interfering with the acquisition flow.

During development, a limitation was found in the YARP Python bindings, where the overload of getFeature returning the valid minimum and maximum range of a feature was not exposed for frame grabbers; a contribution was submitted to the official YARP repository to expose it, enabling feature-range queries from high-level interfaces.[1]

Thanks to the ROS 2 wrapper for YARP frame grabber devices, the image stream is published on the ROS 2 side as sensor_msgs/Image accompanied by sensor_msgs/CameraInfo, while the control interface is mapped onto a set of services that mirror the frame grabber control methods, as summarized in Table 1. This lets the same hardware abstraction feed both TEO through YARP and TIAGo through ROS 2.

Table 1: Frame grabber interfaces mapped to ROS 2.

topic / service	message / service type	role
image	sensor_msgs/Image	publisher
camera_info	sensor_msgs/CameraInfo	
has_feature	HasFeature	service
get_feature	GetFeature	
set_feature	SetFeature	
get_feature_range	GetFeatureRange	
set_mode	SetMode	

For interactive use, a control application named GrabberControls2Gui, developed in Python with PyQt5 and the YARP Python bindings, connects to the device through a YARP network wrapper client. It follows a model, view and controller separation paradigm, with the visual layout designed in Qt Designer. It provides real-time visualization of the image stream and intuitive parameter control through sliders and spin boxes, automatically exposing only the features supported by the connected camera and reflecting their valid ranges. Image frames are received on a buffered image port and mapped to a Qt pixmap via ctypes, since the official frame grabber image interface lacked usable Python bindings at the time of development.

The above described interface is shown in Figure 5.

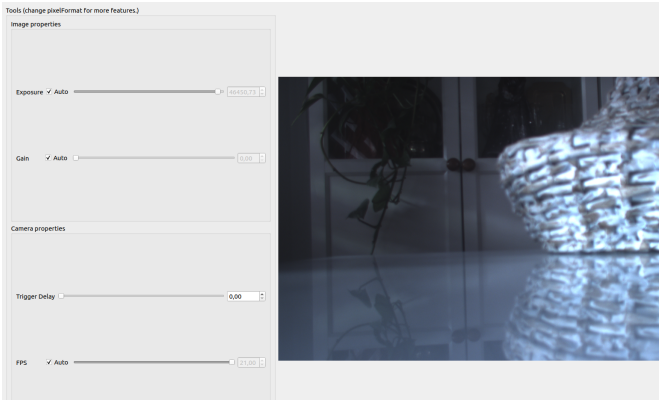

Figure 5: Graphical control interface, developed in Python.

5. Experiments

Validation was carried out on both platforms, exercising the two acquisition paths (Figure 6 and Figure 7). For TEO, a simulated environment based on Gazebo Classic is used together with the real industrial camera for the acquisition tests; the YARP server and the Gazebo world containing the TEO model are launched first, followed by the AravisGigE device, the ArUco and head detection modules, and the Cartesian controllers for the head and the right arm, all orchestrated through a yarpmanager application file for reproducibility. Before connecting the real camera, a fake acquisition mode verifies the correct visualization of a synthesised moving grayscale gradient pattern and the handling of the data flow; a laptop webcam is also used as a provisional substitute during early debugging of the detection pipeline, prior to integrating the definitive industrial camera.

The physical camera, a FLIR Flea3 FL3-U3-88S2C-C equipped with a 6 mm HR f/1.2 objective, is then connected and the device is verified to transform the stream to RGB when configured in color mode while the camera delivers a grayscale pixel format. Camera parameters are modified using both the interactive terminal and the graphical interface, confirming that the two stay synchronized. The right arm is first set to an extended reference pose through an RPC call to the simulated controller, and the visual servoing demonstration is then launched: the camera detects a human head and, once a specific ArUco marker is identified using the native fiducial detector of OpenCV 4.7 (Romero-Ramirez et al., 2018), the head reorients to track the target while the spatial pose error, filtered by a deadband to avoid jitter near the goal, is mapped into a Cartesian velocity command (KDL::Twist) that drives the 6-DoF right arm through the Orocos Kinematics and Dynamics Library in incremental steps, decelerating as it approaches the marker.

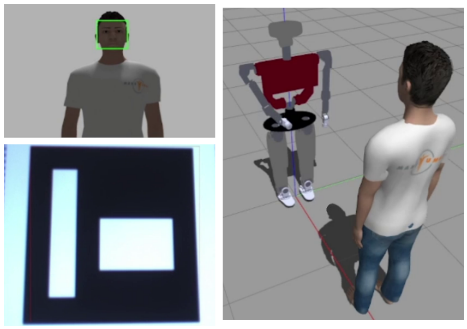

Figure 6: Experiment with TEO.

For TIAGo, the same device is exposed through the ROS 2 network wrapper, publishing the camera stream as sensor_msgs/Image with the corresponding sensor_msgs/CameraInfo, and the image topic is visualized in RViz and consumed by a ROS 2 detection node analogous to the one used on TEO. To exercise the GigE Vision branch of the driver, a secondary color camera with a Gigabit Ethernet interface (The Imaging Source DFK Z12G445) is used.

[1] https://github.com/robotology/yarp/pull/3236

This confirms that the same hardware abstraction is reused without modifying the driver and that exposure and gain can be adjusted from the ROS 2 side through the mapped services, with the path validated in simulation using the TIAGo model and mirroring the procedure followed with TEO.

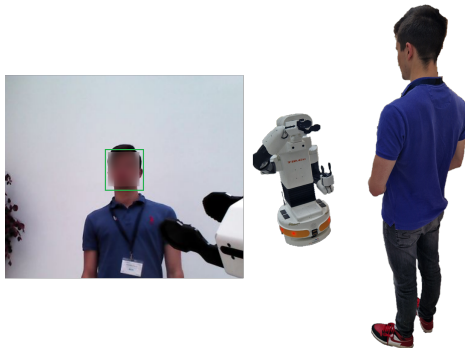

Figure 7: Experiment with TIAGo.

6. Conclusions

This work has presented an acquisition middleware for GenICam-compliant industrial cameras, implemented as a YARP device on top of the Aravis library and exposed simultaneously over the YARP and ROS 2 networks, so that the same hardware abstraction is reused on the TEO humanoid through YARP and on the TIAGo mobile manipulator through ROS 2. A graphical control interface, dynamic pixel-format conversion and a visual servoing demonstration guided by ArUco markers have been validated, together with a contribution to the YARP Python bindings. The main limitations concern the run-time reconfiguration of the pixel format, which currently requires a device restart, the still partial coverage of the frame grabber bindings in high-level languages, and the lack of support for hardware-triggered synchronization across multiple cameras.

A quantitative evaluation of latency, frame rate and CPU overhead when streaming simultaneously over YARP and ROS 2, together with a direct comparison against existing ROS 2 GenICam drivers, was not performed in this work and is identified as a priority for future validation.

Future work will also focus on resolving the pixel-format reconfiguration, extending compatibility to additional camera protocols, adding remote procedure calls to the interactive terminal, and integration with ROS 2 (Łukawski et al., 2025). The source code is publicly available online.[2]

[2] https://github.com/roboticslab-uc3m/aravis-yarp-devices

Acknowledgments

This research has been financed by “iRoboCity2030-CM, Robótica Inteligente para Ciudades Sostenibles” (TEC-2024/TEC-62, 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.

References

Automated Imaging Association (AIA), 2026. GenICam standard. https://www.automate.org/vision/vision-standards/vision-standards-additional-machine-vision-standards.

FLIR Integrated Imaging Solutions Inc., 2012. Flea3 USB3 Cameras Datasheet (FL3-U3-88S2C-C). https://altami.ru/assets/files/brochures/datasheet_flea3_usb3.pdf.

Łukawski, B., Montesino Valle, I., Victores, J.G., Jardón Huete, A., Balaguer, C., 2022. An inverse kinematics problem solver based on screw theory for manipulator arms, in: XLIII Jornadas de Automática, CEA. doi:doi.org/10.17979/spudc.9788497498418.0864.

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

Łukawski, B., Victores, J.G., Balaguer, C., 2023. A generic controller for teleoperation on robotic manipulators using low-cost devices, in: XLIV Jornadas de Automática, CEA. doi:10.17979/spudc.9788497498609.785.

Metta, G., Fitzpatrick, P., Natale, L., 2006. YARP: Yet another robot platform. International Journal of Advanced Robotic Systems 3, 43–48. doi:10.5772/5761.

Pérez Martínez, C., Pierro, P., Martinez, S., Pabon, L., Arbulú, M., Balaguer, C., 2010. RH-2: an upgraded full-size humanoid platform, in: Mobile Robotics: Solutions and Challenges. World Scientific, pp. 471–478. doi:10.1142/9789814291279_0058.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., 2009. ROS: an open-source robot operating system, in: ICRA Workshop on Open Source Software, p. 5.

Romero-Ramirez, F.J., Muñoz-Salinas, R., Medina-Carnicer, R., 2018. Speeded up detection of squared fiducial markers. Image and Vision Computing 76, 38–47. doi:10.1016/j.imavis.2018.05.004.