MujocoROS2Control: Seamless MuJoCo Integration with ROS 2 for Robot Simulation and Control

Adrian Danzglock [1], **Vamsi Krishna Origanti** [1], **and Frank Kirchner** [1,2]

1 Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI), Bremen, Germany **2** Universität Bremen, Bremen, Germany

DOI: 10.21105/joss.09140

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @ishaanamahajan
- @mazrk7

Submitted: 25 September 2025
Published: 10 December 2025

Summary

`ROS 2` is an open-source framework that provides standardized communication, tools, and libraries for building robot applications. It uses a Data Distribution Service (DDS) backend for reliable, real-time communication and supports features such as zero-copy message passing when nodes run on the same system. ROS 2 is supported by a broad ecosystem of community and industry-maintained packages, ranging from sensor drivers to visualization tools and control modules.

`MuJoCo` (Multi-Joint dynamics with Contact) is a high-performance physics engine designed for accurate simulation of articulated bodies, contacts, and constraints. Widely used in robotics and machine learning research, it emphasizes both simulation speed and physical fidelity. MuJoCo is implemented in C/C++ and also offers a JAX-based version optimized for GPU-accelerated reinforcement-learning workflows.

The `MujocoROS2Control` hardware interface enables seamless integration between MuJoCo (Todorov et al., 2012), a high-performance physics engine, and ROS 2 (Open Robotics, 2023), widely adopted middleware for robotic systems. This interface provides an efficient solution for simulating and controlling robots using MuJoCo's physics capabilities within the ROS 2 ecosystem.

To support ROS-based workflows, we developed a dedicated URDF-to-MJCF conversion script. This tool translates URDF models into MJCF (MuJoCo XML format), preserving kinematic and dynamic properties and allowing custom MuJoCo-specific parameters such as sensors, actuators, and collision definitions to be specified directly in the URDF. This conversion ensures compatibility and adaptability for simulation.

`MujocoROS2Control` bridges the gap between ROS 2 and MuJoCo, offering a streamlined workflow for simulation, controller testing, and reinforcement learning.

Statement of Need

ROS 2 is widely used across robotics, but its associated simulator—Gazebo/Ignition—is generally not as fast or as convenient for dynamic simulation as MuJoCo. MuJoCo, in turn, is widely used in robotics-focused machine learning. Establishing a MuJoCo interface for ROS 2 therefore enables developers to use the same simulator both for training and for validating policies before deployment on real systems, or to test ROS 2 components without a physical system, helping unify robotics development pipelines.

Developing and validating control algorithms for robotic systems often requires extensive testing, which on physical hardware can be expensive, time-consuming, and subject to wear. Accurate simulation environments are essential for safe and scalable development.

MuJoCo (Multi-Joint dynamics with Contact) is a fast and accurate physics engine designed for robotics, control, biomechanics, and reinforcement learning. It provides precise multi-body dynamics with advanced numerical integration, high-speed simulation for real-time control and machine learning, sophisticated contact modeling and soft constraint handling, flexible actuator models and comprehensive sensor support.

MuJoCo is ideal for reliable torque feedback due to its high-fidelity torque-controlled joints and smooth-contact soft constraint solver, which enables precise compliance behavior (Zhang et al., 2025).

Gazebo, on the other hand, typically relies on hard-constraint engines like ODE or Bullet, which struggle with compliant torque-based tasks unless carefully tuned. These engines often require smaller timesteps than MuJoCo to maintain stability (Gazebo Simulation Team, 2025).

Drake supports torque control, but due to its symbolic and rigid-body emphasis, tuning compliance can be computationally intensive (Tedrake & the Drake Development Team, 2019).

Table 1: Comparison of usability of different Simulators for usage with Compliant controllers

Controller Type	**MuJoCo**	**Gazebo/Ignition**	**Drake**
Torque-based impedance	High-fidelity, fast, stable	Accurate but tuning required	Accurate, symbolic, slower than mujoco
Admittance with wrench input	Stable, smooth compliant	Plugin-lag, tuning required	precise, heavier computationally
Time-step flexibility	Larger stable steps (e.g. 2-5ms)	Smaller steps required (1ms)	Variable but slower dynamics

Table 2: Comparison of actual ROS 2 simulator wrappers

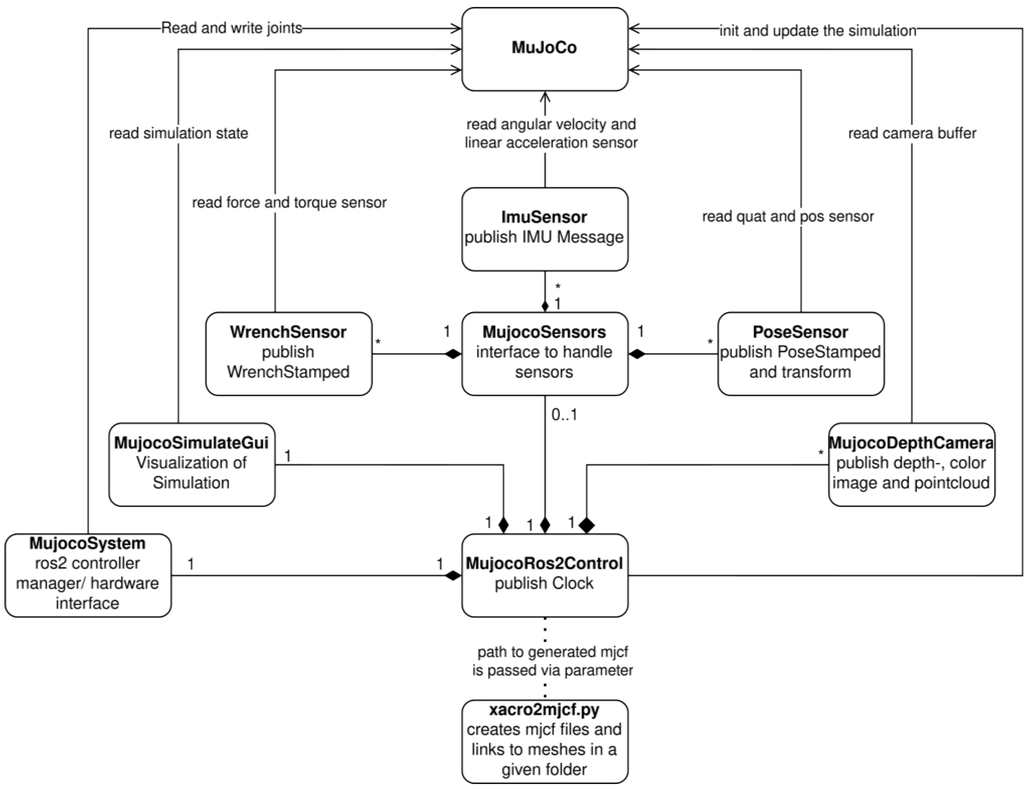
Feature	**Mujoco ROS2 Control**	**mujoco ros2 control (MoveIt, 2025)**	**gz ros2 control (ROS-Controls Team, 2025)**	**drake-ros (Robot Locomotion, 2025)**	**mujoco ros2 (Woolfrey, 2025)**	**mujoco ros (Peter David Fagan, 2025)**
Simulation Engine	MuJoCo	MuJoCo	Gazebo	Drake	MuJoCo	MuJoCo
Sensor Support	Yes, IMU, Pose, Wrench, RGBD	Yes, IMU, Wrench, RGBD	Yes, Supports various sensors via Gazebo plugins	Yes, but aditional code is required	No	Must be implemented in the env
URDF Support	Yes, direct loading from URDF via urdf to mjcf script in launchfile	No, Planned	Yes, Uses URDF/SDF for robot models	Yes, Supports URDF and custom formats	No	No

Feature	**Mujoco ROS2 Control**	**mujoco ros2 control (MoveIt, 2025)**	**gz ros2 control (ROS-Controls Team, 2025)**	**drake-ros (Robot Locomotion, 2025)**	**mujoco ros2 (Woolfrey, 2025)**	**mujoco ros (Peter David Fagan, 2025)**
Control System	ros2_control	ros2_control	ros2_control	Experimental API for ROS2	Topic based	Must be implemented in the env
Control Methods	PID, Mujoco Actuators, Torque	PID, Torque, not integrated position and velocity control	PID, Effort, Position, Velocity	PID, Optimization-based-control	Mujoco Actuators	Must be implemented in the env
Mimic Joints	Yes	Yes	Yes, sometimes difficult to setup (Stack Exchange User, 2025)	Yes	No	Must be implemented in the env

Despite these capabilities, MuJoCo lacks native support for ROS 2, limiting its adoption in modern robotic development pipelines. `MujocoROS2Control` addresses this gap, enabling users to simulate ROS 2-compatible robots in MuJoCo with minimal overhead.

Implementation

`MujocoROS2Control` integrates MuJoCo with the `ros2_control` framework. A key component is the URDF-to-MJCF converter, which maintains fixed joints which MuJoCo typically collapses, allows sensor and actuator tags within URDFs, and generates MJCF files compatible with MuJoCo's expectations.



MuJoCo
Read and write joints
init and update the simulation
read simulation state
read angular velocity and linear acceleration sensor
read camera buffer
read force and torque sensor
read quat and pos sensor
ImuSensor
publish IMU Message
WrenchSensor
publish WrenchStamped
MujocoSensors
interface to handle sensors
PoseSensor
publish PoseStamped and transform
MujocoSimulateGui
Visualization of Simulation
MujocoDepthCamera
publish depth-, color image and pointcloud
MujocoSystem
ros2 controller manager/ hardware interface
MujocoRos2Control
publish Clock
path to generated mjcf is passed via parameter
xacro2mjcf.py
creates mjcf files and links to meshes in a given folder

Figure 1: Overview of the MujocoRos2Control Structure

The interface supports direct torque control, PID-based position/velocity/acceleration control and MuJoCo's native actuator models.

Joint states and simulation time are published for synchronization with the ROS 2 system time (`/clock`). Sensors defined in the URDF are exposed as individual ROS nodes using `realtime_tools` (ROS-Controls Team, 2024) to maintain real-time performance.

Use Cases

This package provides a fast and accurate dynamics simulation environment that can serve as a drop-in alternative to Gazebo/Ignition. Robots can be integrated by supplying a URDF model, which is automatically converted to MJCF. The framework also supports extensibility through a sensor interface that allows users to implement custom sensing modules.

Typical application domains include:

- contact-rich manipulation tasks (e.g., interactions with grippers and multiple high-resolution rigid bodies),
- testing controllers in position, velocity, or torque mode,
- evaluating controllers that rely on wrench feedback (e.g., admittance control),
- simulating underactuated systems (such as robots with free-floating bases), and
- visual-perception tasks (although this was not the primary focus).

Usage in projects

MujocoRos2Control was utilized for testing and validating various torque and admittance controllers within the scope of the HARTU project (*HARTU Project*, n.d.). The software also played a key role in conducting experiments for the publication "Look-Ahead Optimization for

Managing Nullspace in Cartesian Impedance Control of Dual-Arm Robots" (Origanti et al., 2025).

Our framework has been successfully employed to test components such as force-torque sensor gravity compensation, torque-based Cartesian controllers, and Dynamic Movement Primitives (DMP)-based skill reproduction (Fabisch, 2024).

Examples

Franka FR3 with IndustRealKit Gears

This example integrates the Franka FR3 robot (FRANKA ROBOTICS, 2024) with high-resolution gear models from the IndustRealKit (Tang et al., 2023). MuJoCo actuators are used to generate joint torques, although the implemented PID control and torque control are also supported. For the high-resolution collision modeling, we use CoaCD (Wei et al., 2022) to create multiple convex hulls from complex mesh geometry. The URDF-to-MJCF converter automatically replaces original mesh files in the mjcf, when the converted files are in the same directory.

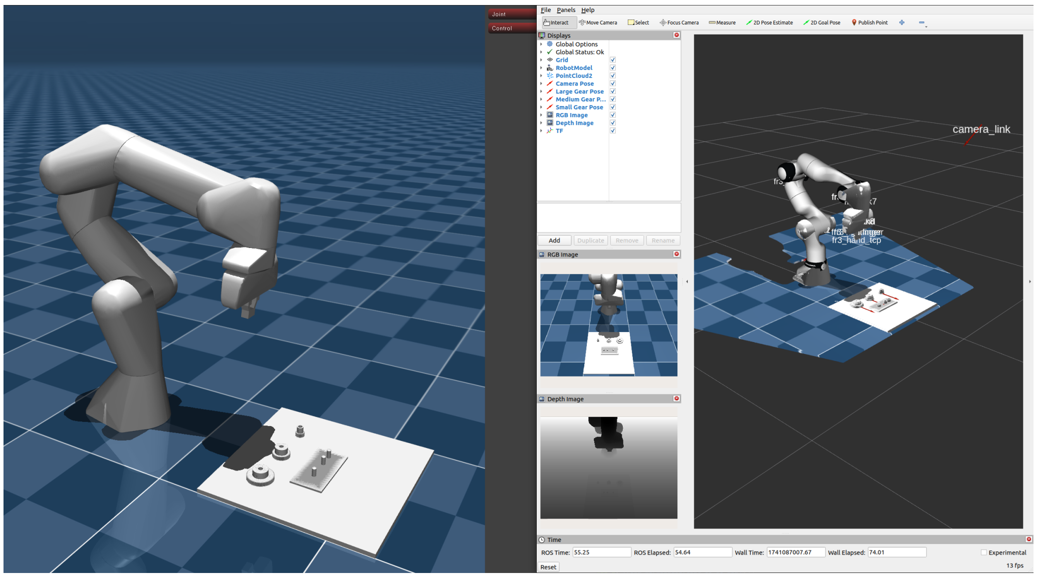


Figure 2: Franka FR3 controlled with ROS 2 Joint Trajectory Controller

Unitree H1

This example uses the loads the Bipedal robot Unitree H1 (Unitree Robotics, 2025) with a floating base and tf2 transformations from world to pelvis. All joints are controlled via ros2_control, with mujoco actuators for position and velocity control (PD Control).

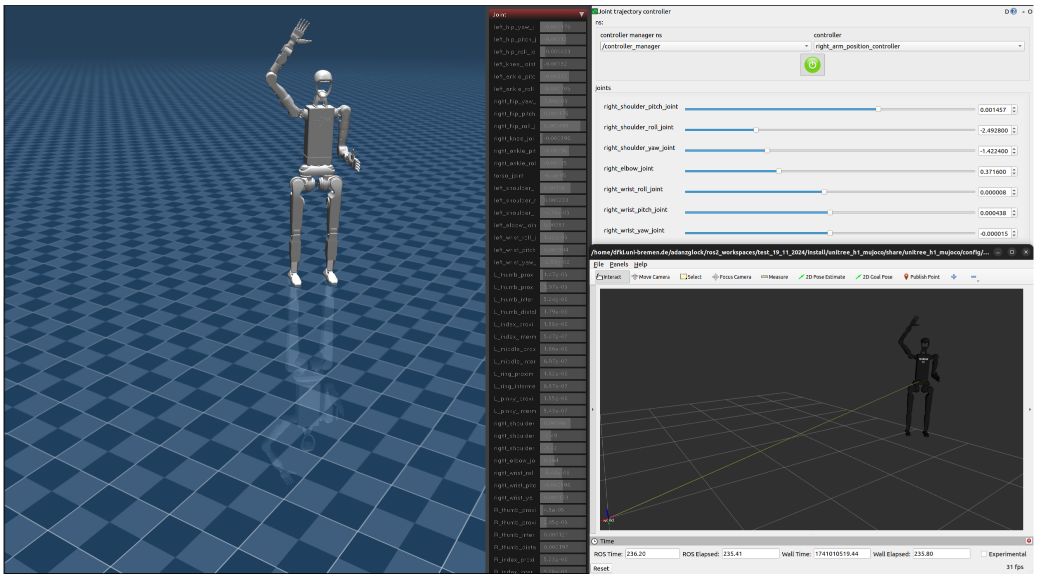


Figure 3: Unitree H1 controlled with ROS 2 control and tf2 transformation, and IMU

IMRK System

In this example, we simulate the iMRK system developed at DFKI Bremen, consisting of two KUKA LBR iiwa 14 robots (Mronga, 2025). A ROS 2 Cartesian impedance controller (migrated version of (Mayr & Salt-Ducaju, 2024)) is used for each arm. MuJoCo actuators manage the Robotiq 2F grippers, and force-torque sensors are simulated at both end-effectors. Because the robot description for the IMRK system is not public available, we can't provide the files for this example.

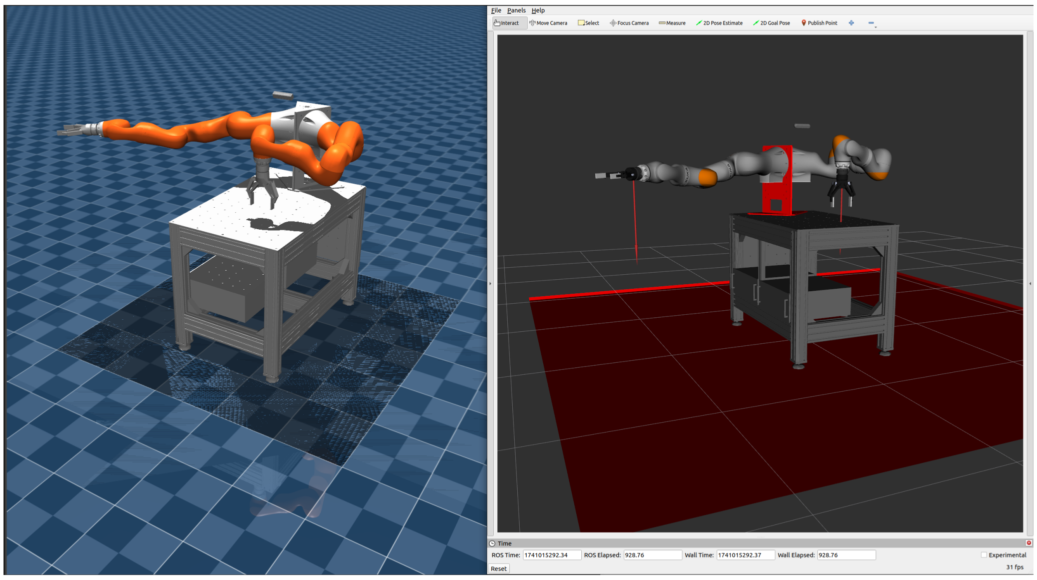


Figure 4: IMRK with Robotiq 2F Gripper and Robotiq FT300 Sensor

Conclusion

`MujocoROS2Control` enables high-fidelity robotic simulation by integrating MuJoCo with ROS 2. Through its robust conversion utilities, sensor bridging, and actuator support, it provides a reliable framework for control development, validation, and machine learning research.

Its lightweight design and fast simulation performance make it well-suited for force-based control, trajectory optimization, and reinforcement learning applications. Future enhancements may include support for additional sensor types, multi-robot coordination, and real-time closed-loop learning.

Acknowledgements

This library was initiated and developed at the Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), Bremen, Germany, as part of the HARTU Project. This project received funding from the European Union's Horizon Europe research and innovation program under grant agreement No. 101092100.

References

Fabisch, A. (2024). Movement_primitives: Imitation learning of Cartesian motion with movement primitives. *Journal of Open Source Software*, *9*(97), 6695. https://doi.org/10.21105/joss.06695

FRANKA ROBOTICS. (2024). *Franka_description: Official models of Franka Robotics GmbH robots*. https://github.com/frankaemika/franka_description.

Gazebo Simulation Team. (2025). *Gazebo physics parameters*. https://classic.gazebosim.org/tutorials?tut=physics_params&cat=physics

HARTU project. (n.d.). https://www.hartu-project.eu/.

Mayr, M., & Salt-Ducaju, J. M. (2024). A C++ implementation of a Cartesian impedance controller for robotic manipulators. *Journal of Open Source Software*, *9*(93), 5194. https://doi.org/10.21105/joss.05194

MoveIt. (2025). *MuJoCo ROS2 control*. https://github.com/moveit/mujoco_ros2_control

Mronga, D. (2025). *iMRK dual-arm robot*. https://robotik.dfki-bremen.de/en/research/robot-systems/imrk/.

Open Robotics. (2023). *ROS 2: Robot operating system*. https://github.com/ros2.

Origanti, V. K., Danzglock, A., & Kirchner, F. (2025). Look ahead optimization for managing nullspace in Cartesian impedance control of dual-arm robots. *2025 IEEE/SICE International Symposium on System Integration (SII)*, 990–997. https://doi.org/10.1109/SII59315.2025.10871056

Peter David Fagan. (2025). *A ROS 2 package for launching MuJoCo simulation instances that interface with ROS 2*. https://github.com/peterdavidfagan/mujoco_ros.

Robot Locomotion. (2025). *Drake ROS integration*. https://github.com/RobotLocomotion/drake-ros

ROS-Controls Team. (2024). *Realtime_tools: Contains a set of tools that can be used from a hard realtime thread, without breaking the realtime behavior*. https://github.com/ros-controls/realtime_tools.

ROS-Controls Team. (2025). *Gz_ros2_control: Connect the latest version of Gazebo with ros2_control*. https://github.com/ros-controls/gz_ros2_control.

Stack Exchange User. (2025). *Gazebo ROS2 control mimic joints not working as expected*. https://robotics.stackexchange.com/questions/110622

Tang, B., Lin, M. A., Akinola, I., Handa, A., Sukhatme, G. S., Ramos, F., Fox, D., & Narang, Y. (2023). *IndustReal: Transferring contact-rich assembly tasks from simulation to reality*. https://doi.org/10.15607/RSS.2023.XIX.039

Tedrake, R., & the Drake Development Team. (2019). *Drake: Model-based design and verification for robotics*. https://drake.mit.edu

Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. https://doi.org/10.1109/IROS.2012.6386109

Unitree Robotics. (2025). *Unitree_ros*. https://raw.githubusercontent.com/unitreerobotics/unitree_ros.

Wei, X., Liu, M., Ling, Z., & Su, H. (2022). Approximate convex decomposition for 3D meshes with collision-aware concavity and tree search. *ACM Trans. Graph.*, *41*(4). https://doi.org/10.1145/3528223.3530103

Woolfrey, J. (2025). *Allows for communication between MuJoCo and ROS2*. https://github.com/Woolfrey/mujoco_ros2.

Zhang, J. Z., Howell, T. A., Yi, Z., Pan, C., Shi, G., Qu, G., Erez, T., Tassa, Y., & Manchester, Z. (2025). *Whole-body model-predictive control of legged robots with MuJoCo*. https://arxiv.org/abs/2503.04613