CRISP - Compliant ROS2 Controllers for Learning-Based Manipulation Policies and Teleoperation

Daniel San José Pro, Oliver Hausdörfer, Ralf Römer, Maximilian Dösch, Martin Schuck and Angela P. Schoellig

***Abstract*— Learning-based controllers, such as diffusion policies and foundation models output low-frequency and discontinuous target states. Tracking such references smoothly requires a low-level controller to handle such high-level commands and convert them to joint torques, ideally in a compliant way. To this end, we present CRISP, a lightweight C++ implementation of compliant Cartesian and joint-space controllers for the ROS2 control standard designed for seamless integration with learning-based policies and teleoperation. Through our CRISP_PY Python and CRISP_GYM Gymnasium interfaces, we provide a unified pipeline to use the CRISP controllers for data collection through teleoperation, policy training, and subsequent policy deployment. The controllers are compatible with any manipulator exposing a joint effort interface and have been validated on numerous platforms including Franka Robotics FR3 and Universal Robot UR5. Designed for rapid integration, flexible deployment, and real-time performance, our implementation provides a unified pipeline for data collection and policy execution, lowering the barrier to applying learning-based methods on ROS2-compatible manipulators. Detailed documentation is available at the project website.**

***Index Terms*— Compliance and Impedance Control, Teleoperation, Software-Hardware Integration for Robot Systems, Foundation Models, Imitation Learning.**

I. Introduction

In recent years, learning-based policies for robotic manipulation have become increasingly prevalent. Such policies command discontinuous, low-frequent changes in the robot state as action often at around 5 to 10 Hz [1]–[5]. This paradigm requires low-level controllers to respond to command changes during deployment rather than following precomputed smooth trajectories as used in traditional control pipelines for manipulation [6], [7]. At the same time, many challenging manipulation tasks involve interaction with the environment and contact-rich behaviors, necessitating the use of compliant controllers.

While ROS and ROS2 have generally simplified robotics software development, current ROS2 tooling for such requirements often introduces significant overhead through extensive frameworks. At the same time, several robot manufacturers including Franka Robotics, Kinova, Duatic, and ROBOTIS, together with the broader ROS2 community, have contributed high-quality, lightweight `ros2_control` [8] interfaces that support direct torque control. This convergence creates an opportunity to implement modular, low-level controllers that integrate directly with the ROS2 control interface, are compatible with any manipulator providing a joint torque interface, and are well-suited to the requirements of high-level learning-based policies without the overhead of extensive frameworks.

Main Contributions. To this end, we present CRISP, an implementation of compliant Cartesian and joint-space controllers for robotic manipulators compatible with ROS2 and `ros2_control` for use with teleoperation and learning-based policies. CRISP_PY and CRISP_GYM provide lightweight Python and the Gymnasium interfaces to use the CRISP controllers in a unified pipeline for data collection and policy deployment.

II. Related Work

Traditionally, executing complex robot manipulation tasks relies on a path planning and perception pipeline. Established frameworks such as MoveIt [9] provide comprehensive planning capabilities; however, they often depend on extensive software stacks that require users to have in-depth knowledge for effective integration into their systems. In contrast, recent learning-based approaches focus on producing small chunks or low-frequency target poses rather than full trajectories, which reduces system complexity and necessitates a fundamentally different architectural framework.

Several existing efforts address discontinuous Cartesian control within ROS. Table I compares our implementation with other available options. The Cartesian controllers developed by the FZI Research Center for Information Technology (FZI CC) [10] support position and velocity control but do not provide torque-based control. Other controllers, such as the Cartesian Impedance Controller (CIC) [11], `compliant_controllers` [12], or `serl_franka_controllers` [13], implement Cartesian impedance control but are only available for ROS1 or are specific to a single manipulator type. Additional user-developed tools have emerged [14]–[16], but these are typically robot-specific or lack `ros2_control` integration. LeRobot [17] is one emerging framework that improves accessibility for learning-based manipulation, but does not yet support a wide range of manipulators, low-level controllers, and lacks integration with ROS2.

In contrast to previous works, we present robot-agnostic controllers that integrate seamlessly with the ROS2 standard.

The authors are with the Technical University of Munich (TUM), Germany; the TUM School of Computation, Information and Technology, Department of Computer Engineering, Learning Systems and Robotics Lab and the TUM Munich Institute of Robotics and Machine Intelligence (MIRMI). Email: {daniel.sanjose.pro; oliver.hausdoerfer; maximilian.doesch; martin.schuck; angela.schoellig}@tum.de.

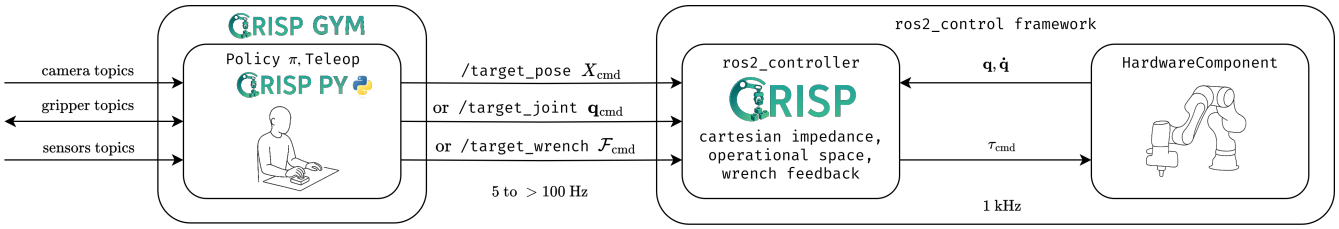

Fig. 1. CRISP, CRISP_PY, and CRISP_GYM architecture overview and integration with robots using ROS2. The user can publish target commands at arbitrary frequencies via ROS2 topics, and the robot will track the most recent target. For instance, high-level learning-based policies such as vision-language-action (VLA) models use the `target_pose` topic. We run the CRISP controllers on a real-time enabled Linux workstation at 1 kHz, while CRISP_GYM with the VLA model runs on a separate workstation in the same network, communicating via ROS2 topics. CRISP_GYM further easily integrates other hardware components, such as robotic grippers, cameras, and additional sensors via ROS2 topics.

	FZI CC	CIC	Ours
ROS1	x	x	
ROS2	x		x
Torque/effort interface		x	x
Robot agnostic	x	x	x
Highly Configurable	(x)	x	x
Cartesian Impedance		x	x
Nullspace Biasing		x	x
Force control		x	x
Joint limits barrier			x
Friction Compensation			x
Operation Space Control			x
Built with `pinocchio`[1]			x
Gymnasium Wrapper			x

TABLE I

COMPARISON TO THE CLOSEST AVAILABLE CONTROLLERS

Through additional interfaces, we allow both data collection through teleoperation and deployment of learning-based policies in one pipeline.

III. IMPLEMENTATION

To ensure robot agnosticism, we implement compliant controllers in C++ for the `ros2_control` interface, capable of running at 1 kHz on a joint-torque interface. These controllers are designed to be responsive and robust to the discontinuous command updates typically generated by high-level learning-based policies. As shown in Figure 1, users can publish target commands at any frequency using, for instance, CRISP_PY and CRISP_GYM, and the controller will continuously track the most recent command.

The controllers implement joint limit barriers and include friction compensation for smooth teleoperation. All required rigid-body computations, as well as parsing of the robot structure from the standard Universal Robot Description File (URDF), are handled by `pinocchio` [18]. The `pinocchio` library efficiently implements rigid-body dynamics, is distributed with ROS2 binaries, and supports modern automatic differentiation in addition to other features required for advanced control algorithms.

[1]A ROS native, well-tested kinematics and rigid-body dynamics library based on Lie groups.

To enable configurability and ease of integration into other software stacks, all controller parameters are dynamically configurable via standard ROS2 parameter services. Examples of configurable parameters include stiffness, damping, nullspace stiffness, torque rate limits, and error limits. Several default configurations are provided in our repositories.

A. Task Torque

The task torque $\boldsymbol{\tau}_{\text{task}}$ refers to the desired torques to be applied to solve the task of following a target pose $(\mathbf{x}_{\text{target}}, \mathbf{R}_{\text{target}}) \in \mathbb{R}^3 \times SO(3)$ which is represented with respect to the base frame. It attempts to minimize the error $\mathbf{e} = (\mathbf{e}_{\text{pos}}, \mathbf{e}_{\text{rot}}) \in \mathbb{R}^6$ with respect to the current pose $(\mathbf{x}_{\text{current}}(\mathbf{q}), \mathbf{R}_{\text{current}}(\mathbf{q}))$ computed using forward kinematics. The error can computed with respect to the base as

$$\mathbf{e}_{\text{pos}} = \mathbf{x}_{\text{target}} - \mathbf{x}_{\text{current}}(\mathbf{q}) \tag{1}$$

$$\mathbf{e}_{\text{rot}} = \text{Log}(\mathbf{R}_{\text{target}}\mathbf{R}_{\text{current}}^{\top}(\mathbf{q})) \tag{2}$$

or with respect to the end effector-frame as

$$\mathbf{e}_{\text{pos}} = \mathbf{R}_{\text{current}}^{\top}(\mathbf{x}_{\text{target}} - \mathbf{x}_{\text{current}}(\mathbf{q})), \tag{3}$$

$$\mathbf{e}_{\text{rot}} = \text{Log}(\mathbf{R}_{\text{current}}^{\top}(\mathbf{q})\mathbf{R}_{\text{target}}), \tag{4}$$

where Log is the logarithmic map used to map a rotation difference to the tangent space of $SO(3)$. This can be seen as an angular velocity that turns the end-effector rotation towards the target rotation [19]. Defining the error in different frames allows the user to define anisotropic stiffness and damping behavior of the controller with respect to the end-effector or base frame.

1) Cartesian Impedance (CI) Controller: This controller implements a virtual spring between the end-effector and the reference pose and is well suited for contact-rich, compliant tasks, as it adapts well to contacts [20]:

$$\boldsymbol{\tau}_{\text{task}} = \mathbf{J}^{\top}\left(\mathbf{K}_{\text{p}}\mathbf{e} - \mathbf{K}_{\text{d}}\mathbf{J}\dot{\mathbf{q}}\right), \tag{5}$$

where $\mathbf{J}$ denotes the geometric Jacobian with respect to the base or end-effector frame computed from the provided URDF and $\mathbf{K}_{\text{p}}$, $\mathbf{K}_{\text{d}}$ are the stiffness and damping matrices respectively chosen by the user.

2) Operational Space Controller (OSC): Compared to the CI controller, OSC accounts for the dynamics, in particular the inertia, in the control law [21]. The OSC tends to be more precise but less compliant in the presence of contact. The task torque is given by

$$\boldsymbol{\tau}_{\text{task}} = \mathbf{J}^\top \boldsymbol{\Lambda}(\mathbf{q}, \dot{\mathbf{q}}) \left(\mathbf{K}_{\text{p}}\mathbf{e} - \mathbf{K}_{\text{d}}\mathbf{J}\dot{\mathbf{q}}\right), \tag{6}$$

where $\boldsymbol{\Lambda}(\mathbf{q}, \dot{\mathbf{q}}) = (\mathbf{J}^{\dagger\top}\mathbf{M}^{-1}\mathbf{J}^{\dagger})^{-1}$ denotes the task-space mass matrix computed using the mass matrix $\mathbf{M}$. The mass matrix is determined with the provided URDF using `pinocchio`. OSC does not compensate for the Coriolis and gravity terms in the task space. Instead, these are dynamically decoupled and directly compensated in joint-space, since it makes computations easier and the performance is similar to the original OSC [22].

B. Null Space Control

For redundant manipulators, we provide the option to control in the null space of the task, which is useful to limit unnecessary motions, avoid singularities, and stabilize the joints to the resting position. This can be achieved by projecting a secondary control law $\boldsymbol{\tau}_{\text{secondary}}$ with a null space projector $\mathbf{N} \in \mathbb{R}^{n\times n}$ to a subspace that does not interfere with the primary task [23]. The secondary task in our control law is a joint impedance controller computed as

$$\boldsymbol{\tau}_{\text{secondary}} = \mathbf{K}_{\text{p,s}}(\mathbf{q}_{\text{target}} - \mathbf{q}) + \mathbf{K}_{\text{d,s}}(\dot{\mathbf{q}}_{\text{target}} - \dot{\mathbf{q}}). \tag{7}$$

We provide different projectors to compute the null space torque $\boldsymbol{\tau}_{\text{ns}} = \mathbf{N}\boldsymbol{\tau}_{\text{secondary}}$: (1) a static null space projector $\mathbf{N} = \mathbf{I} - \mathbf{J}^\top(\mathbf{J}^\dagger)^\top$, (2) a dynamic null space projector $\mathbf{N} = \mathbf{I} - \mathbf{J}^\top\bar{\mathbf{J}}^\top$, where $\bar{\mathbf{J}} = \mathbf{M}^{-1}\mathbf{J}^\top(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^\top)^{-1}$ is the generalized inverse Jacobian [21], and (3) an identity null space projector $\mathbf{N} = \mathbf{I}$ that can be used to directly control the manipulator with joint commands if the task stiffness $\mathbf{K}_{\text{p}}$ and damping $\mathbf{K}_{\text{d}}$ have been set to 0. The dynamic null space projector leads to better primary task tracking compared to the static null space projector, provided that a sufficiently accurate model (URDF) of the robot is available [24].

C. Joint Barrier

In order to avoid reaching the joint limits of the robot and prevent damage, we add joint barriers that apply an increasing torque as the joint gets close to the limits by ϵ:

$$\tau_{i,\text{joint}} = \begin{cases} -K_{i,\text{joint}}(q_{i,\text{max}} - q_i) & \text{if } q_i > \theta_{i,\text{max}} - \epsilon, \\ -K_{i,\text{joint}}(q_{i,\text{min}} - q_i) & \text{if } q_i < \theta_{i,\text{min}} + \epsilon, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

D. Gravity and Coriolis Compensation

We efficiently compute the gravity compensation and coriolis term using `pinocchio`, which uses the Recursive Newton Euler Algorithm (RNEA) to solve the Inverse Dynamics problem $\tau = RNEA(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \boldsymbol{\mathcal{F}}_{\text{ext}})$ to determine $\boldsymbol{\tau}_{\text{gravity}} = \mathbf{g}(\mathbf{q}) = RNEA(\mathbf{q}, \mathbf{0}, \mathbf{0}, \mathbf{0})$ and $\boldsymbol{\tau}_{\text{coriolis}} = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$. For the user, it is therefore important to provide the inertial parameters in the URDF file if these are not included by default. In case a manipulator already provides gravity compensation, it can be removed from the control law.

E. Friction Compensation

We provide a term to compensate for friction effects that are unmodeled terms in the equations of motion for rigid bodies. The friction can be modeled as a combination of static Coulomb friction, viscous friction, and a friction offset. We provide a simplified model as used in [25] to estimate the Franka Emika Robot friction term:

$$\boldsymbol{\tau}_{\text{friction}} = \boldsymbol{\varphi}_1 \left(\sigma(\boldsymbol{\varphi}_2(\dot{\mathbf{q}} + \boldsymbol{\varphi}_3)) - \sigma(\boldsymbol{\varphi}_2\boldsymbol{\varphi}_3)\right) \tag{9}$$

with $\sigma(u) = \frac{1}{1+\exp(-u)}$. This model of friction neglects viscous terms and uses a sigmoidal function instead of a sign function to avoid sudden discontinuities around zero velocities.

While we provide friction parameters for the Franka Robotics FR3 [25], these parameters must be determined for each robot by the user or the term omitted entirely. In practice, this term is particularly useful for teleoperation, as it makes the operator feel less unnatural resistance when moving the robot.

F. Target Wrench

Using our implementations, it is possible to command a wrench at the end-effector to apply external forces and torques. The term is computed as $\boldsymbol{\tau}_{\text{wrench}} = \mathbf{J}^\top\boldsymbol{\mathcal{F}}_{\text{target}}$, where $\boldsymbol{\mathcal{F}} \in \mathbb{R}^6$ denotes the desired wrench. Wrench control is useful for tasks that demand careful interaction with the environment, require specific torques to be applied such as in screw driving [7], compensate for the weight of end-effector extensions, or simply drive the robot with a wrench instead of Cartesian poses.

G. Final Control Law

Our final control law is a sum of all the previous terms:

$$\begin{aligned} \boldsymbol{\tau}_{\text{cmd}} = \boldsymbol{\tau}_{\text{task}} + \boldsymbol{\tau}_{\text{ns}} + \boldsymbol{\tau}_{\text{joint}} + \boldsymbol{\tau}_{\text{gravity}} \\ + \boldsymbol{\tau}_{\text{coriolis}} + \boldsymbol{\tau}_{\text{friction}} + \boldsymbol{\tau}_{\text{wrench}} \end{aligned} \tag{10}$$

In our implementation, each term can be switched on and off depending on the task, giving full flexibility to the user.

H. Additional Features

We provide additional safety features including joint torque and rate limits, exponential filtering of targets to avoid fast movements, and tracking error clipping for precise yet safe control when targets are distant [13]. For teleoperation, we implement force-torque feedback controllers where follower forces are reflected to the leader as $\boldsymbol{\tau}_{\text{fb}} = -k_{\text{p,fb}}\mathbf{J}^\top\mathcal{F}_{\text{follower}} - k_{\text{d,fb}}\dot{\mathbf{q}}_{\text{leader}}$.

IV. Evaluation

We tested the on hardware using the Franka Robotics FR3 and in simulation with the Kuka IIWA14 and Kinova Gen3, as shown in Figure 3. Videos of these experiments are available on the project website. In this section, we present results for three representative tasks performed with our controllers on a real Franka Robotics FR3. The exact tracking performance depends on the parameters selected by the user for the specific task as well as the manipulator used.

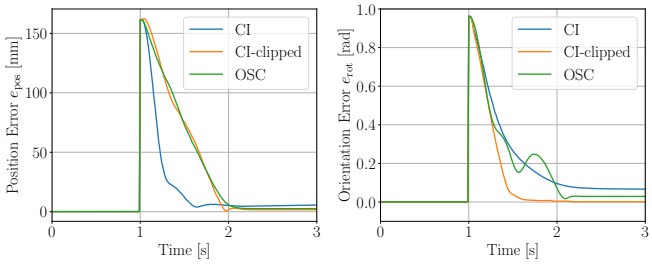

Fig. 2. **Robot-agnostic Cartesian control [I]:** Error evolution for tracking a discontinuous target pose using a Franka Emika FR3 on hardware with CRISP. We set a new target pose at $t = 1$ s. For reproducibility, the controller parameters used are available on our website. Note that exact tracking performance depends on controller parameterization, which should be chosen based on the specific task. **Left:** Position error evolution with steady-state errors of 5.54 mm, 4.73 mm, and 0.81 mm for OSC, CI, and CI-clipped, respectively. **Right:** Rotational error evolution with steady-state errors of 0.0998 rad, 0.0532 rad, and 0.0029 rad for OSC, CI, and CI-clipped, respectively.

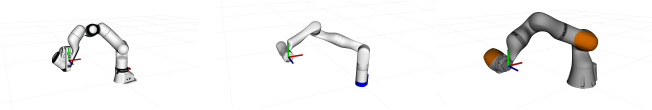

Fig. 3. **Robot agnostic Cartesian control [II]:** We demonstrate our controllers on a Franka FR3 on hardware, and on multiple manipulators in simulation. **Left:** Franka Robotics FR3. **Middle:** Kinova Gen3. **Right**: Kuka IIWA 14.

First, we evaluate the tracking capabilities by issuing a randomly generated target pose using CRISP_PY within a safe motion range and monitoring the evolution of the tracking error over time, as shown in Figure 2. This experiment illustrates the tracking performance for a fixed target pose under different parameter configurations available to the user.

Second, we demonstrate our teleoperation setup for a Lego stacking task in Figure 4. During teleoperation, the end effector poses from the leader robot are streamed to the follower at roughly 30 Hz, where they are tracked using the CRISP CI controller. The torque-based controllers allow for direct force feedback to the human operator as well as compliant behavior during task execution.

Third, we evaluate our controllers deployed using the CRISP_GYM interface with two imitation learning policies, namely Diffusion Policy [3] and SmolVLA [26], running at different inference frequencies on a GPU. The policies are trained using the data collected in the previously described task and using the same CRISP_GYM interface. The results of deployment target command tracking during deployment are shown in Figure 5.

Our experiments demonstrate that the proposed controllers are suitable to track both poses commanded through a human operator via teleoperation and policy deployment.

V. Conclusion

In this work, we present CRISP - an implementation of controllers for robotic manipulators designed for the deployment of learning-based policies that require tracking of small action chunks or low-frequency, discontinuous target joints or poses. The robot-agnostic implementation in ROS2 enables broad applicability across platforms, while its lightweight software design facilitates easy integration in new software stacks. Our framework, including the CRISP_PY and CRISP_GYM interfaces, can be used for both data collection and policy deployment across different manipulators, thereby accelerating research in learning-based robotics.

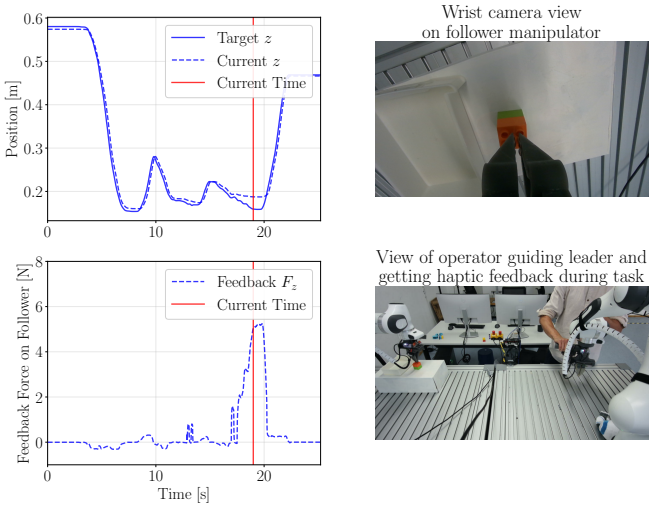

Fig. 4. **Unified data collection and policy deployment [I]:** Recording a block-stacking task with Lego using teleoperation and force-torque feedback for the operator. **Upper left:** Plot of the follower's target and end-effector positions. **Lower left:** Feedback wrenches applied at the leader manipulator. **Upper right:** Image of the recording on the follower manipulator at the current time. **Lower right:** Image of the operator guiding the leader robot to record data on the follower at the current time.

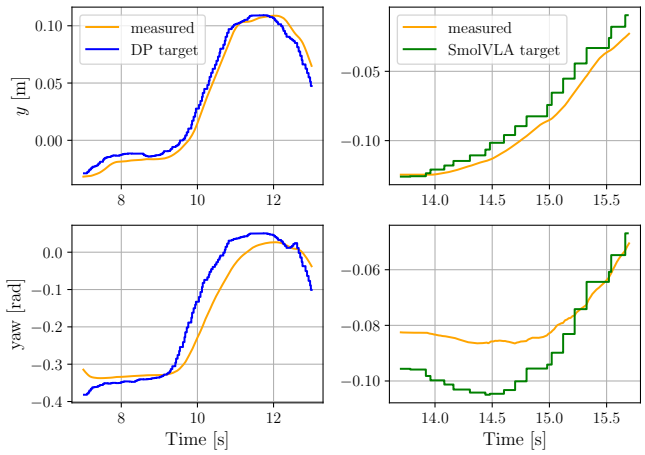

Fig. 5. **Unified data collection and policy deployment [II]:** Using the same recorded data in the CRISP_GYM interface, different learned policies can be successfully deployed in hardware. **Left:** Diffusion Policy [3] operating at approximately 30 Hz. **Right:** SmolVLA [26] operating at approximately 10 Hz. Both policies provide end-effector target pose updates that are tracked by our CRISP controllers. Note the different timescales shown in the plots. For policy deployment, we used the CI controllers with CRISP_GYM.

References

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh,

R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," in *Proceedings of the 7th Conference on Robot Learning*, vol. 229, 2023, pp. 2165–2183.

[2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," in *Proceedings of Robotics: Science and Systems (RSS XIX)*, 2023.

[3] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *The International Journal of Robotics Research*, 2024.

[4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, "π_0: A Vision-Language-Action Flow Model for General Robot Control," 2024.

[5] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, "OpenVLA: An Open-Source Vision-Language-Action Model," in *Proceedings of the 8th Conference on Robot Learning*, vol. 270, 2025, pp. 2679–2713.

[6] M. Brady, J. M. Hollerbach, T. Johnson, T. Lozano-Perez, and M. T. Mason, *Robot Motion: Planning and Control*, 1982.

[7] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. New York: Cambridge University Press, 2017.

[8] "ROS2 control," https://control.ros.org/rolling/index.html, accessed: 2024-05-03.

[9] D. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study," *Journal of Software Engineering for Robotics*, vol. 5, no. 1, pp. 3–16, 2014.

[10] S. Scherzinger, A. Roennau, and R. Dillmann, "Forward Dynamics Compliance Control (FDCC): A new approach to cartesian compliance for robotic manipulators," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4568–4575.

[11] M. Mayr and J. M. Salt-Ducaju, "A C++ Implementation of a Cartesian Impedance Controller for Robotic Manipulators," *Journal of Open Source Software*, vol. 9, no. 93, p. 5194, 2024.

[12] A. L. Mitchell, T. Flatscher, and I. Posner, "Task and Joint Space Dual-Arm Compliant Control," 2025.

[13] "serl_franka_controllers," https://github.com/rail-berkeley/serl_franka_controllers, accessed: 2025-05-05.

[14] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, "VIOLA: Imitation Learning for Vision-Based Manipulation with Object Proposal Priors," *6th Annual Conference on Robot Learning (CoRL)*, 2022.

[15] J. Elsner, "Taming the Panda with Python: A powerful duo for seamless robotics programming and integration," *SoftwareX*, vol. 24, p. 101532, 2023.

[16] T. Schneider, "franky," https://github.com/TimSchneider42/franky, accessed: 2025-08-04.

[17] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, S. Palma, P. Kooijmans, M. Aractingi, M. Shukor, D. Aubakirova, M. Russi, F. Capuano, C. Pascale, J. Choghari, J. Moss, and T. Wolf, "LeRobot: State-of-the-art Machine Learning for Real-World Robotics in Pytorch," https://github.com/huggingface/lerobot, 2024.

[18] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[19] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," 2021.

[20] N. Hogan, "Impedance Control: An Approach to Manipulation," in *1984 American Control Conference*, 1984, pp. 304–313.

[21] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[22] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational Space Control: A Theoretical and Empirical Comparison," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.

[23] A. Dietrich, C. Ott, and A. Albu-Schäffer, "An overview of null space projections for redundant, torque-controlled robots," *The International Journal of Robotics Research*, vol. 34, no. 11, pp. 1385–1400, 2015.

[24] A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger, "Cartesian impedance control of redundant robots: Recent results with the DLR-light-weight-arms," in *IEEE International Conference on Robotics and Automation*, vol. 3. IEEE, 2003, pp. 3704–3709.

[25] C. Gaz, M. Cognetti, A. Oliva, P. Robuffo Giordano, and A. De Luca, "Dynamic Identification of the Franka Emika Panda Robot With Retrieval of Feasible Parameters Using Penalty-Based Optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4147–4154, 2019.

[26] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti *et al.*, "SmolVLA: A vision-language-action model for affordable and efficient robotics," *arXiv preprint arXiv:2506.01844*, 2025.