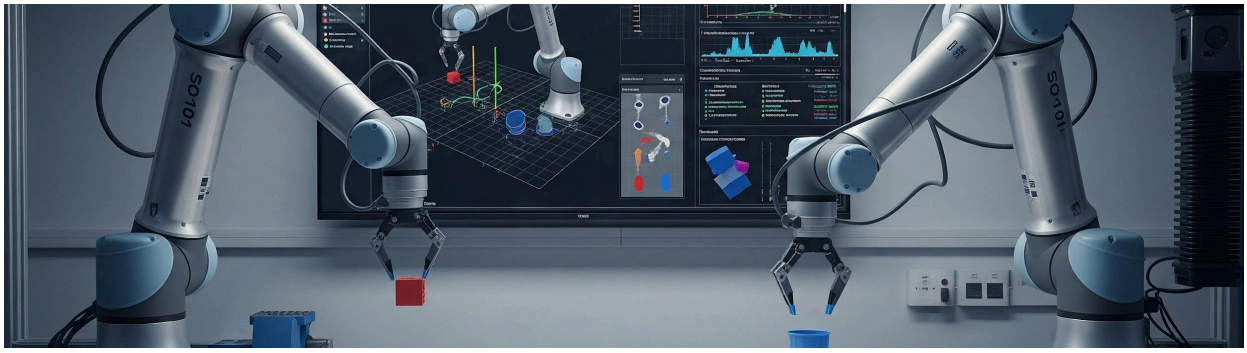

Masterclass Final Project Proposal: Vision-Based Imitation Learning with Dual SO-101 Arms in ROS2

Autonomous Continuous Pick-and-Place with Dual SO-101 Robot Arms Using Vision-Based Imitation Learning

Student: Dmitri Manajev
Date: 21 January 2026

This project combines modern imitation learning with practical ROS2 robotics to create an impressive, real-world autonomous manipulation demo using affordable hardware.

1. Project Description

In this masterclass final project, a complete vision-based imitation learning pipeline is developed using dual SO-101 robot arms (leader-follower setup) fully integrated in ROS2. A human operator teleoperates the leader arm while the follower mirrors movements in real-time, enabling efficient collection of high-quality demonstration data from multi-camera observations (wrist + overhead). An Action Chunking Transformer (ACT) policy is trained on this dataset to enable the follower arm to autonomously perform pick-and-place tasks using only camera images and proprioception—no explicit pose estimation or traditional planning required.

The system progresses from basic teleoperation to closed-loop autonomous execution: first achieving reliable pick-and-place into a fixed box with randomized object positions, then extending to a challenging continuous task where placed objects return to new random locations (via a slotted box), allowing the policy to repeat the manipulation indefinitely with robustness to variation.

The resulting system achieves fully autonomous, vision-only manipulation on low-cost hardware, comparable to state-of-the-art research demonstrations.

2. Key Components and Tasks Achieved

- Real-time dual-arm teleoperation and synchronized data recording (joint trajectories, end-effector poses, multi-camera images).
- Dataset creation and validation with open-loop replay on physical hardware.
- Custom ACT policy implementation in PyTorch, integrated as a ROS2 node for real-time inference.
- Evaluation on simple and advanced pick-and-place scenarios with physical objects.
- Full visualization and debugging in RViz, including URDF models, TFs, and live camera feeds.

3. Technologies and Skills Practiced

- **Main Technologies:** ROS2 Jazzy (control, visualization, bags), PyTorch (ACT implementation), OpenCV (multi-camera processing), Imitation Learning, Dual-arm teleoperation, C++ (optional for performance-critical or custom controllers).
- **Skills:** System integration in ROS2, data pipelines for robot learning, training vision-based policies, real-world deployment and iteration, robot manipulation with low-cost hardware, potential low-level controller optimization in C++ if needed.

4. Expected Outcomes

The final system demonstrates impressive autonomous manipulation: >70% success on randomized pick-and-place and 10+ consecutive cycles in the continuous task, all running on physical SO-101 arms. The project includes polished video demonstrations (teleoperation, training, autonomous tasks), RViz screenshots, success metrics, and failure analysis—resulting in a complete masterclass-style showcase of end-to-end vision-based robot learning.

5. Project Details

- **Level of Complexity:** High
- **Estimated Time:** 100–150 hours
- **Requirements:** Dual physical SO-101 robot arms with USB cameras; primary development and evaluation on real hardware.

6. Task list

1. **Hardware Setup and Dual-Arm Control**
    - Assemble leader and follower SO-101 arms, integrate drivers, and establish basic ROS2 control for both arms independently and in mirrored mode.
    - ***Expected*:** Reliable joint control and real-time leader-follower teleoperation.
2. **ROS2 Visualization in RViz**
    - Publish joint states, TFs, and multi-camera feeds (wrist + overhead) with proper URDF models for both arms.
    - ***Expected*:** Full dual-arm workspace visible in RViz with synchronized robot models and live camera views.
3. **Teleoperation and Demonstration Collection**
    - Implement a smooth teleoperation interface and record synchronized demonstration episodes capturing joint trajectories, end-effector poses, and multi-camera images. Use ROS2 bags or compatible/reusable formats (potentially drawing from established solutions like LeRobot's structure for efficiency).
    - ***Expected*:** Dataset of 50–100 high-quality episodes with diverse reaching, grasping, and manipulation motions, easily viewable and ready for training.
4. **Episode Replay and Validation**
    - Develop replay functionality to execute recorded trajectories on the physical follower arm.
    - ***Expected*:** Accurate open-loop reproduction of collected demonstrations.
5. **ACT Policy Implementation and Training**
    - Implement Action Chunking Transformer (ACT) policy in PyTorch, adapt data pipeline, and train on collected dataset.
    - ***Expected*:** Trained policy capable of predicting joint actions from camera observations and proprioception.
6. **Closed-Loop Evaluation on Simple Pick-and-Place Task**
    - Deploy policy as ROS2 node for real-time inference and evaluate on pick-and-place into a fixed box with randomized object positions.
    - ***Expected*:** Autonomous success rate >60–70% over multiple trials, with video demonstration.
7. **Advanced Continuous Pick-and-Place Task**
    - Extend task to a setup where placed objects return to random positions (e.g., via slotted box), enabling long-horizon repetition. Retrain/fine-tune policy as needed.

- ***Expected*:** Policy performs 10+ consecutive cycles autonomously, demonstrating robustness to randomization.

8. **Final Demonstration and Documentation**
    - Prepare polished videos (teleop, training process, both tasks), RViz screenshots, success metrics, and failure analysis.
    - ***Expected*:** Complete masterclass-style project presentation with impressive autonomous manipulation demos.

7. Resources

The following resources will be utilized for this project:

- **Hardware**:
    - Two physical SO-101 robot arms (leader and follower configuration)
    - Two USB cameras (one wrist-mounted on the follower arm, one overhead/third-view)
    - Objects for manipulation (e.g., small blocks or toys) and a custom box with slot for the continuous task
- **Software**:
    - ROS2 Jazzy (primary control and visualization framework)
    - Python 3.10+
    - PyTorch for ACT policy implementation and training
    - OpenCV for camera processing
    - Optional: Reusable components from open-source libraries (e.g., LeRobot dataset structure or data loading utilities) to accelerate development
    - Visualization tools: RViz, Foxglove Studio, or Rerun.io for dataset inspection
    - Optional: MuJoCo for potential simulation-based debugging or policy pre-training (if suitable models become available)
- **Development Environment**:
    - Ubuntu 24.04 (recommended stable version)
    - GPU recommended for faster training (local machine or cloud instance)

8. References

- ACT paper: Zhao et al., "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware" (arXiv 2023)
- LeRobot documentation (optional reference for data formats): https://huggingface.co/docs/lerobot
- ROS2 documentation and community SO-101 packages