From Scratch to Task: Building a SO-ARM100 Robot Arm for Custom Grasping with LeRobot

Spring 2025 GPT robotics Project

袁能远
202364820241

魏鹏宇
202364820791

Abstract

This paper presents the problem of building embodied intelligent Lerobot SO-ARM100 robotic arms and complete custom grasping tasks, explores the relevant methods, and introduces the key contributions of this work. The goal of this project is to complete the production, training, and evaluation of a dataset for scotch grasping, and the results demonstrate [highlight the main findings]. Our approach builds upon the instruction 'using SO-ARM100 in lerobot' [1] provided by seeed studio, as well as lerobot framework, and extends it by implementing solutions for potential calibration and remote control errors in the robotic arm. Additionally, a dataset for custom material gripping was constructed and trained using ACT (Adaptive Control for Manipulation).

1 Introduction

The rise of robotics and imitation learning has made it increasingly important to develop reliable grasping systems for low-cost robotic arms. While previous works have attempted to address this issue through reinforcement learning frameworks, these approaches are often limited by high sample complexity and hardware deployment challenges. In this paper, we focus on building an adaptive grasping pipeline for the SO-ARM100 robotic arm using the Lerobot framework, aiming to achieve custom material gripping tasks. Our approach builds upon Seeed Studio's integration guidelines[1] and extends them by implementing solutions for potential calibration and remote control errors. Key contributions include: (1) A systematic workflow for hardware debugging and policy deployment.(2) Construction of a custom dataset for material grasping trained with ACT.(3) Experimental validation revealing critical insights on policy robustness.

2 Related Work

2.1 Lerobot with SO-ARM100

As an open-source lightweight robotic platform, Lerobot integrates the SO-ARM100 manipulator with ROS2 middleware to provide unified control interfaces. Its core architecture comprises three layers: Hardware abstraction layer translating joint commands via CAN bus, motion planning layer leveraging MoveIt for collision-free trajectory generation, and task-oriented API layer enabling high-level operations like grasping. The SO-ARM100's 5-DOF design with servo driven makes it suitable

for tabletop manipulation tasks, while Lerobot's simulation-gym toolkit facilitates policy transfer from Gazebo to physical hardware.

2.2 ACT Policy for Grasping

ACT (Action Chunking with Transformer) is an end-to-end imitation learning framework that decomposes continuous manipulation into executable action chunks. Its transformer-based architecture processes observation sequences $o_{1:t}$ to predict action windows $a_{t:t+k}$, effectively balancing long-horizon reasoning and fine-grained control. Compared to traditional RL methods like DDPG, ACT reduces sample complexity in grasping tasks. We adopt ACT due to its compatibility with Lerobot's demonstration recorder module, allowing direct policy training from human-guided teleoperation trajectories.

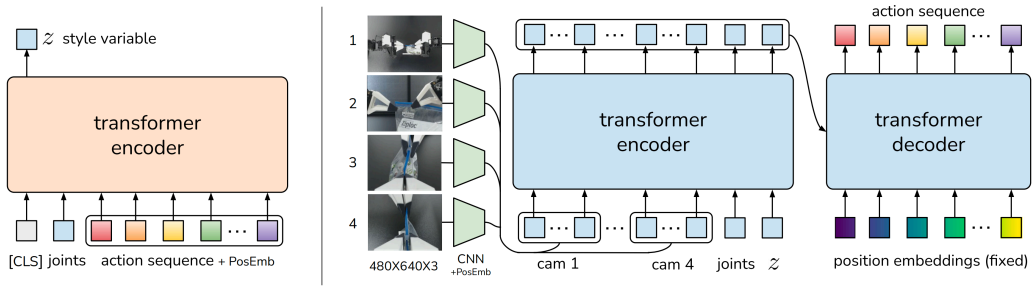

figure 1: Schematic of the ACT Policy Architecture[2]

3 Method

3.1 ACT-Lerobot Integration Framework

Our methodology fundamentally integrates ACT's end-to-end learning framework with Lerobot's three-tiered architecture to enable robust grasping on the SO-ARM100 platform. This synthesis leverages Lerobot's hardware abstraction layer to convert ACT's predicted action chunks $a_{t:t+k}$ into precise CAN bus commands executable by the servo-driven joints, while utilizing MoveIt's motion planning capabilities - as implemented in Lerobot's middleware layer - to transform these actions into collision-free trajectories. Crucially, we exploit the native compatibility between ACT and Lerobot's demonstration recorder module, which permits direct policy training from human-guided teleoperation trajectories recorded through Lerobot's task-oriented API. This integrated approach systematically addresses the calibration anomalies and remote control discrepancies documented in 4.1, while preserving seamless interoperability with Lerobot's simulation-gym toolkit for efficient policy transfer between simulated and physical environments.

3.2 Implementation Process

1. Hardware Setup and Stability

Following the hardware integration steps outlined by Seeed Studio[1], we completed the physical assembly and initial connection of the SO-ARM100 arm. Key to proceeding further was ensuring basic operational stability, which involved resolving critical errors encountered during initial testing.

2. Dataset Construction and Training

As the methodology outlined in the Abstract, we constructed a specialized dataset for custom material gripping tasks using Lerobot's demonstration recorder capabilities. This involved recording human-guided teleoperation trajectories under varying conditions, with particular attention to the experimental scenarios described in 4.2: the controlled setting with standardized lighting and backdrop. ACT policies were subsequently trained on this dataset following the evaluation framework established in 4.2, employing the transformer-based architecture to process observation sequences $a_{1:t}$ and predict action windows $a_{t:t+k}$ while accounting for the hardware-specific constraints identified during debugging.

4 Experiments

4.1 Hardware Debugging workflow

The hardware debugging workflow for the robotic arm system comprises several critical stages: servo configuration, arm assembly, calibration, remote control testing, camera integration. Throughout this process, systematic error handling proved essential for successful implementation. During servo configuration, strict adherence to the ID assignment sequence (1-6) is required to ensure proper communication between servos and the control system. In teleoperation testing, an "OutRangeError: Wrong motor position range detected for gripper" may arise if the leader and follower arm ports are mismatched in the configuration file. Before recalibration, port assignments should be verified. Subsequently, using "FT Servo Debug (a servo diagnostic utility)" to diagnose servo connectivity and positional accuracy assists to avoid unnecessary reinitialization. Additionally, physically repositioning the arm to the servo's default starting position can resolve initialization failures. An observed anomaly occurs when a servo remains unresponsive during calibration yet functions correctly in teleoperation. This issue can often be mitigated by manually resetting the joint position of the affected arm segment before initiating teleoperation, ensuring proper servo engagement.

4.2 Grasping Task Evaluation

We systematically evaluated grasping performance through three experimental iterations. The initial attempt in a dormitory environment with cluttered background and variable lighting resulted in complete failure to initiate grasping, as background noise in RGB-D observations caused ACT's attention mechanism to focus on non-task-relevant features, confirming the necessity of controlled environments for imitation learning. The second attempt under controlled conditions with white backdrop and consistent illumination successfully accomplished the grasp phase but failed completely in placement, with the root cause being policy overfitting to suboptimal trajectory segments due to excessive micro-adjustments during human demonstrations, which induced oscillatory placement behavior. The final attempt using an augmented dataset with increased demonstration volume and fixed camera pose showed significant test performance degradation, with critical findings revealing extreme policy sensitivity to positional deviations and failure to generalize to minor environmental variations.

In the ACT framework, the Transformer-based attention mechanism inherently processes all visual features in the observation input equally. In the cluttered dormitory environment, dynamic backgrounds such as moving shadows and changing lighting caused the RGB-D observations o_t to contain significant noise features. These features were irrelevant to the grasping task, yet received attention weights $\alpha_{ij} \propto \exp(q_i^T k_j)$ comparable to the actual grasping targets during computation.

This attention dispersion led the policy network to waste computational resources on background features.

The three experiments collectively revealed ACT's pathological dependence on absolute positioning. The root cause lies in: ACT learns a direct mapping from observation space to action space through Transformer$(o_{1:t}) \rightarrow a_{t:t+k}$, rather than general grasping logic. During data collection, the operator's multiple micro-adjustments for precise positioning caused the policy to misinterpret "repeated adjustments" as necessary action patterns. More critically, when slight pose differences existed between testing and training environments, such as camera angle offset, ACT's output action sequence $a_{t:t+k}$ exhibited cumulative errors due to its lack of explicit coordinate system invariance design.

5 Conclusion

This project demonstrates that deploying imitation learning on SO-ARM100 hardware necessitates systematic error handling—including strict servo ID sequencing (1-6) and physical reset protocols —to resolve initialization failures, while 4.2 experiments reveal ACT's pathological dependence on absolute positioning, causing performance degradation in cluttered scenes and failure to generalize with augmented datasets. Building on these findings, future work will integrate domain adaptation with randomized backgrounds for robustness, develop force-torque feedback mechanisms to address oscillatory placement, and implement high-precision encoders to mitigate servo jitter. Crucially, we deliver: (1) a debugging toolkit solving calibration/remote control errors, (2) a custom ACT-trained material grasping dataset evaluated across scenarios, and (3) an open-source framework compatible with Lerobot's simulation-gym toolkit.

References

[1] Seeed Studio. How to use the SO-ARM100 and SO-ARM101 robotic arm in Lerobot, n.d. Seeed Studio Wiki. Accessed: 2025-06-10.

[2] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Appendix A:

Command of scripts for all steps in the project:

```
#Setting Port Permissions
sudo chmod 666 /dev/ttyACM0
sudo chmod 666 /dev/ttyACM1
#Servo Configuration
python lerobot/scripts/configure_motor.py \
  --port /dev/ttyACM0 \
  --brand feetech \
  --model sts3215 \
  --baudrate 1000000 \
  --ID 1
```

```
#Calibration
python lerobot/scripts/control_robot.py \
  --robot.type=so100 \
  --robot.cameras='{}' \
  --control.type=calibrate \
  --control.arms='["main_follower"]'
python lerobot/scripts/control_robot.py \
  --robot.type=so100 \
  --robot.cameras='{}' \
  --control.type=calibrate \
  --control.arms='["main_leader"]'

#Adding Cameras
python lerobot/common/robot_devices/cameras/opencv.py \
    --images-dir outputs/images_from_opencv_cameras

#Teleporation
python lerobot/scripts/control_robot.py \
  --robot.type=so100 \
  --control.type=teleoperate \

#Dataset Recording
huggingface-cli login --token {your_token} --add-to-git-credential
HF_USER=$(huggingface-cli whoami | head -n 1)
echo $HF_USER
python lerobot/scripts/control_robot.py \
  --robot.type=so100 \
  --control.type=record \
  --control.fps=30 \
  --control.single_task="put a scotch tape in the circle." \
  --control.repo_id=${HF_USER}/{dataset_name} \
  --control.tags='["so100","tutorial"]' \
  --control.warmup_time_s=5 \
  --control.episode_time_s=30 \
  --control.reset_time_s=30 \
  --control.num_episodes=10 \
  --control.push_to_hub=true \
  --control.resume=true

#Training
python lerobot/scripts/train.py \
  --dataset.repo_id=${HF_USER}/{dataset_name} \
  --policy.type=act \
  --output_dir=outputs/train/act_so100_test \
  --job_name=act_so100_test \
```

```
  --device=cuda \
  --wandb.enable=true

#Evaluation
python lerobot/scripts/control_robot.py \
  --robot.type=so100 \
  --control.type=record \
  --control.fps=30 \
  --control.single_task="Grasp a tape and put it in the drawing circle." \
  --control.repo_id=${HF_USER}/eval_act_so100_test \
  --control.tags='["tutorial"]' \
  --control.warmup_time_s=5 \
  --control.episode_time_s=30 \
  --control.reset_time_s=30 \
  --control.num_episodes=10 \
  --control.push_to_hub=true \
  --control.policy.path=outputs/train/act_so100_test/checkpoints/last/pretrained_model
```