Group Project

In this project, we will use the RoboTwin 2.0 platform to train a diffusion policy on Galbot for the `beat_block_hammer` task. The task consists of two steps:

1. Use the right gripper to grasp the hammer.
2. Move the hammer to the block and strike the block with the hammer.

A basic implementation of the data generation, data processing, policy training, and simulation evaluation pipeline has been provided in this repository.

We will use **sim-and-real co-training** in this project. A large amount of diverse simulation data can be collected in the simulation environment, while a smaller amount of real-world teleoperation data with more limited coverage will be provided. The real-world data will be released on **June 8, 2026**.

The trained policy will be evaluated in the following settings:

1. Simulation evaluation
2. Real-world in-distribution evaluation
3. Real-world out-of-distribution (but simulation in-distribution) evaluation

You are allowed to modify the simulation data generation method and the sim-and-real co-training strategy. However, you are **not allowed** to modify the diffusion policy model architecture or use any other type of model.

Environment

You can set up the environment by following the instructions in the RoboTwin 2.0 official documentation. The main steps are:

```
conda create -n RoboTwin python=3.10 -y
conda activate RoboTwin
bash script/_install.sh
bash script/_download_assets.sh
```

Additionally, you need to download and unzip the Galbot assets, then place them under `assets/embodiments`.

Part I: Simulation Evaluation (25%)

Simulation evaluation can help verify the correctness of the training and evaluation pipeline, and provide an initial estimate of the policy's capability.

In this part, you may modify the data generation method and the training method, but you may not modify the model architecture. We will evaluate the success rate of your model in the simulation environment. If the success rate is greater than or equal to **70%**, you will receive full credit for this part.

Even if your success rate does not reach 70%, you can still receive most of the points for this part if your implementation is basically correct.

The commands for each step are listed below.

```
ROOT=$(pwd)

# 1. Collect 50 clean right-arm demonstrations
rm -rf data/beat_block_hammer/galbot_demo_clean
python script/collect_galbot_beat_block_hammer_dataset.py \
    --clean-episodes 50 \
    --skip-randomized \
    --save-path data \
    --overwrite \
    --skip-render-test \
    --force-arm-tag right

# 2. Process the data into a single-head-camera, right-arm-only 8D zarr dataset
rm -rf policy/DP/data/beat_block_hammer-galbot_demo_clean-8d-50.zarr
python policy/DP/process_data.py beat_block_hammer galbot_demo_clean 50 \
    --load-dir data/beat_block_hammer/galbot_demo_clean \
    --save-dir policy/DP/data/beat_block_hammer-galbot_demo_clean-8d-50.zarr \
    --right-arm-only

# 3. Train for 600 epochs
cd $ROOT/policy/DP
python train.py --config-name=robot_dp_8.yaml \
    task.name=beat_block_hammer \
    task.dataset.zarr_path=$ROOT/policy/DP/data/beat_block_hammer-
galbot_demo_clean-8d-50.zarr \
    training.num_epochs=600 \
    training.seed=0 \
    training.device=cuda:0 \
    dataloader.batch_size=48 \
    val_dataloader.batch_size=48 \
    head_camera_type=D435 \
    expert_data_num=50 \
    setting=galbot_demo_clean \
    exp_name=galbot_clean50_head_8d \
    logging.mode=offline
```

```
# 4. Evaluate the trained policy
cd $ROOT
CKPT=$ROOT/policy/DP/checkpoints/beat_block_hammer-galbot_demo_clean-8d-50-
galbot_demo_clean-galbot_clean50_head_8d-0/600.ckpt
python script/eval_policy.py \
    --config policy/DP/deploy_policy.yml \
    --overrides \
    --policy_name DP \
    --task_name beat_block_hammer \
    --task_config galbot_demo_clean \
    --ckpt_setting galbot_demo_clean \
    --seed 0 \
    --instruction_type unseen \
    --expert_data_num 50 \
    --checkpoint_num 600 \
    --ckpt_file $CKPT \
    --action_dim 8 \
    --eval_video_log True \
    --eval_test_num 50 \
    --eval_step_lim 200 \
    --force_arm_tag right \
    --force_block_arm_tag right
```

To generate more diverse data, you may increase the randomness of the initial states in simulation, such as the initial poses of the hammer and the block, the initial pose and qpos of the robot, the size of the table, and other important parameters. You may also use the interfaces already provided by RoboTwin to modify the textures of the table and background and the lighting.

Before the real-world data are released, we do not recommend making substantial changes to the trajectory planning method.

Part II: Real-World In-Distribution Evaluation (50%)

This part will be released on **June 8, 2026**.

Part III: Real-World Out-of-Distribution Evaluation (25%)

This part will be released on **June 8, 2026**.

Submission

For the simulation evaluation, please submit the checkpoint of your trained policy on `course.pku.edu.cn`.

Details about the real-world evaluation will be released on **June 8, 2026**.