Physical AIハンズオン

環境

	OS	GPU
ノートPC	Ubuntu24.04 / Windows11	RTX 4060 laptop
計算サーバ	Ubuntu24.04	RTX 5090

1. LeRobotのセットアップ

```
git clone <https://github.com/huggingface/lerobot.git>
cd lerobot
# dataset v2.1のcommitまで戻す
git checkout d602e8169cbad9e93a4a3b3ee1dd8b332af7ebf8
uv sync
uv add opencv-python scipy "wandb<=0.21.0"
uv pip install -e ".[feetech]"
# attention map可視化用
git clone <https://github.com/villekuosmanen/physical-AI-interpretability.git>
```

2. SO-100のセットアップ

参考：
https://github.com/huggingface/lerobot/blob/main/docs/source/so100.mdx

- ポートの検出

フォロワー or リーダー側のUSBケーブルを抜いてEnterを押すと，ポートが表示される

```
uv run lerobot-find-port
```

これ以降，

フォロワー（AIが動かす方）が `/dev/ttyACM1`

リーダー（人間が動かす方）が `/dev/ttyACM0`

として進める.

- ポートに権限を付与

```
sudo chmod 666 /dev/ttyACM0
sudo chmod 666 /dev/ttyACM1
```

- キャリブレーション

こちらの動画のとおりにキャリブレーションする

SO-101

フォロワー

```
uv run lerobot-calibrate \\
        --robot.type=so100_follower \\
    --robot.port=/dev/ttyACM1 \\
    --robot.id=follower
```

リーダー

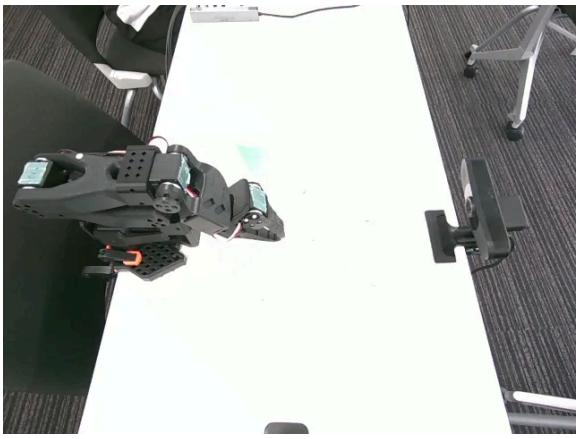
```
uv run lerobot-calibrate \\
        --teleop.type=so100_leader \\
        --teleop.port=/dev/ttyACM0 \\
        --teleop.id=leader
```

- カメラセットアップ

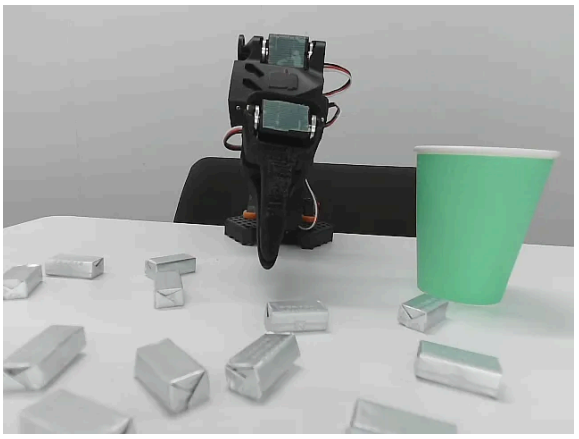
Webカメラ（フロント側）とRealsense（サイド側）のパスを調べる

```
uv run lerobot-find-cameras opencv
```

`outputs/captured_images/`に出力された画像を見て、カメラのパスを特定する



Realsense（サイド側）



Webカメラ（フロント側）

上のような画像になるようにカメラの向きを整える

ロボットのベースの右端が、机の右端から20cm

フロントカメラはロボットの正面

サイドカメラは机の足の横

コップはロボットの右、サイドカメラの正面に置く

コップは養生テープで机に固定する

これ以降，フロントが`/dev/video8`、サイドが`/dev/video6`として進める

- テレオペレーション

SO-101を使っている場合はtypeをso101_follower or so101_leaderとする．

portやidも適宜自分の環境のものに合わせる．

```
uv run lerobot-teleoperate \\
    --robot.cameras="{ front: {type: opencv, index_or_path: /dev/video8, width: 640, height: 480, fps: 30}, side: {type: opencv, index_or_path: /dev/video6, width: 640, height: 480, fps: 30}}" \\
    --robot.type=so100_follower \\
    --robot.port=/dev/ttyACM1 \\
    --robot.id=follower \\
    --teleop.type=so100_leader \\
    --teleop.port=/dev/ttyACM0 \\
    --teleop.id=leader \\
    --display_data=false
```

正常に動作していればOK

3. データ収集

`DATASET_NAME`は既存のものと重複しないように適宜変更する．

それぞれの引数を自分の環境に合わせて修正すること．

```
export DATASET_NAME=record0
uv run lerobot-record \\
    --robot.cameras="{ front: {type: opencv, index_or_path: /dev/video8,
width: 640, height: 480, fps: 30}, side: {type: opencv, index_or_path: /dev/v
ideo6, width: 640, height: 480, fps: 30}}" \\
    --robot.type=so100_follower \\
    --robot.port=/dev/ttyACM1 \\
    --robot.id=follower \\
    --teleop.type=so100_leader \\
    --teleop.port=/dev/ttyACM0 \\
    --teleop.id=leader \\
    --display_data=false \\
    --dataset.repo_id=local/${DATASET_NAME} \\
    --dataset.root=datasets/${DATASET_NAME} \\
    --dataset.num_episodes=60 \\
    --dataset.push_to_hub=false \\
    --dataset.episode_time_s=12 \\
    --dataset.reset_time_s=8 \\
    --dataset.video_encoding_batch_size=20 \\
    --dataset.single_task="Pick up puccho"
```

途中から再開する際は`--resume=true`を付ける

20エピソードごとにエンコードが入るので，そのタイミングで交代する

Tips

- ゆっくり操作する
- ぷっちょの位置はランダムに置くことを心掛ける
- 掴むときはグリッパーを大きく開く
- 掴むときはエンドエフェクタをしっかり回転させる

- ロボットを初期位置に戻すのはリセットタイムになってから．
    - それまではロボットが何らかの動きをとるようにする

4. データマージ

USBにデータを集めてWindowsに移す

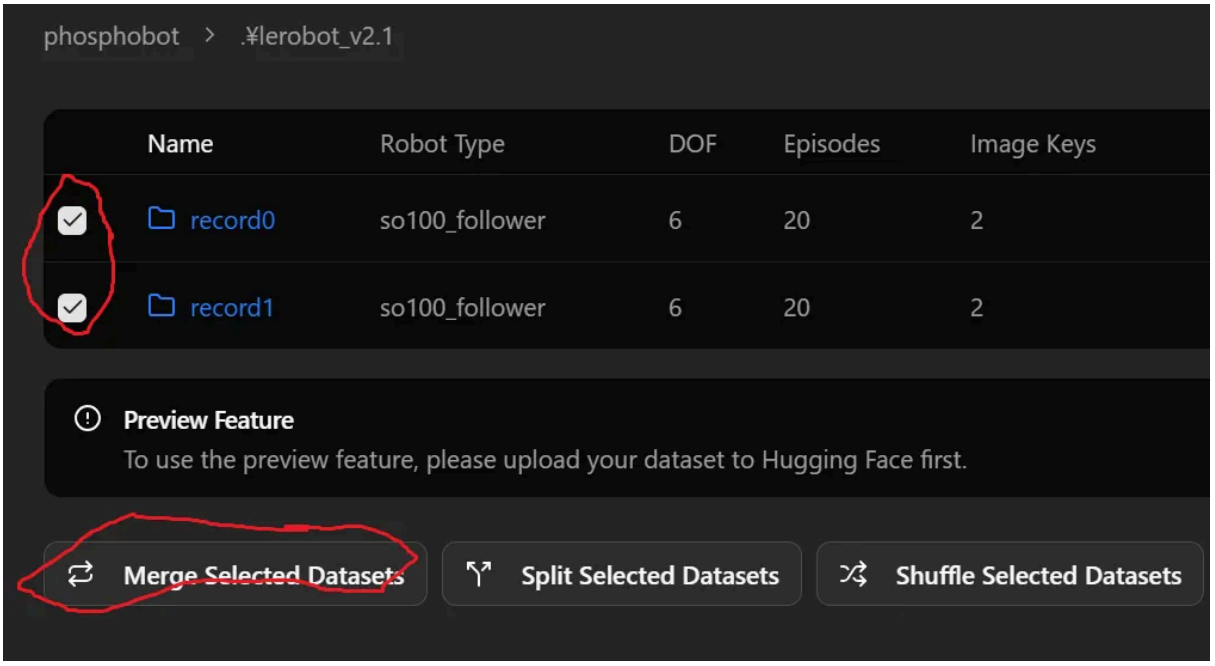
データを`C:\\Users\\<user_name>\\phosphobot\\recordings\\lerobot_v2.1`に貼り付け

以下のコマンドを実行

```
phosphobot run
```

http://localhost/にアクセス

`Browse your Datasets` > `lerobot_v2.1` の順に開く



上の画像のような画面になるので，マージしたいデータセットを選択して`Merge Selected Datasets`を選択．

マージ後の名前を聞かれるので，`merged_record`という名前にして，VSCode経由でサーバに送る．

5. 学習

- サーバ側のセットアップ

```
uv remove torch torchcodec torchvision
uv pip install torch torchvision --index-url <https://download.pytorch.org/whl/cu129>
```

```
export DATASET_NAME=merged_record
uv run lerobot-train \\
  --dataset.repo_id=local/${DATASET_NAME} \\
  --dataset.root=datasets/${DATASET_NAME} \\
  --policy.type=act \\
  --policy.push_to_hub=false \\
  --policy.device=cuda \\
  --wandb.enable=true \\
  --wandb.disable_artifact=true \\
  --output_dir=outputs/train/act_${DATASET_NAME} \\
  --job_name=act_${DATASET_NAME} \\
  --batch_size=16 \\
  --steps=30000
```

https://wandb.ai/hirekatsu0523/lerobot?nw=nwuserhirekatsu0523

こちらのページにアクセスして，ログを表示

RTX 5090なら3万ステップで45分くらい

6. 実機評価

学習後の重み（`pretrained_model`フォルダ）をダウンロードしてUSBに入れて配布．

ノートPC側のlerobotディレクトリ内のoutput下に配置

- `pretrained_model/config.json`の`n_action_steps`を40にする

```
  "chunk_size": 100,
  "n_action_steps": 40, # 100
  "vision_backbone": "resnet18",
```

ちなみに，`n_action_steps`を1，`temporal_ensemble_coeff`を0.01などにすると，temporal_ensemble機能を有効化できる（が，今回のタスクではあまりうまくいかない）．

- 動作確認

`DATASET_NAME`が既存のものと被らないようにする.

名前は必ず`eval_xxx`という形式にする.

```
export DATASET_NAME=eval_act0
uv run lerobot-record \\
    --robot.cameras="{ front: {type: opencv, index_or_path: /dev/video8,
width: 640, height: 480, fps: 30}, side: {type: opencv, index_or_path: /dev/v
ideo6, width: 640, height: 480, fps: 30}}" \\
    --robot.type=so100_follower \\
    --robot.port=/dev/ttyACM1 \\
    --robot.id=follower \\
    --display_data=false \\
    --dataset.repo_id=local/${DATASET_NAME} \\
    --dataset.root=datasets/${DATASET_NAME} \\
    --dataset.num_episodes=1 \\
    --dataset.push_to_hub=false \\
    --dataset.episode_time_s=60 \\
    --dataset.single_task="Pick up puccho" \\
    --policy.path=outputs/pretrained_model
```

- attention mapの可視化

`DATASET_NAME`には評価したいデータセット名を入れる.

```
source .venv/bin/activate
export HF_LEROBOT_HOME=~/SourceCode/lerobot/datasets
export DATASET_NAME=eval_act0
cd physical-AI-interpretability
python -m examples.visualise_original_data_attention \\
    --dataset-repo-id=${DATASET_NAME} \\
    --policy-path=../outputs/pretrained_model
```

`physical-AI-interpretability/output/analysis_output`に動画が出力される