VIP GE:AI

GE Research

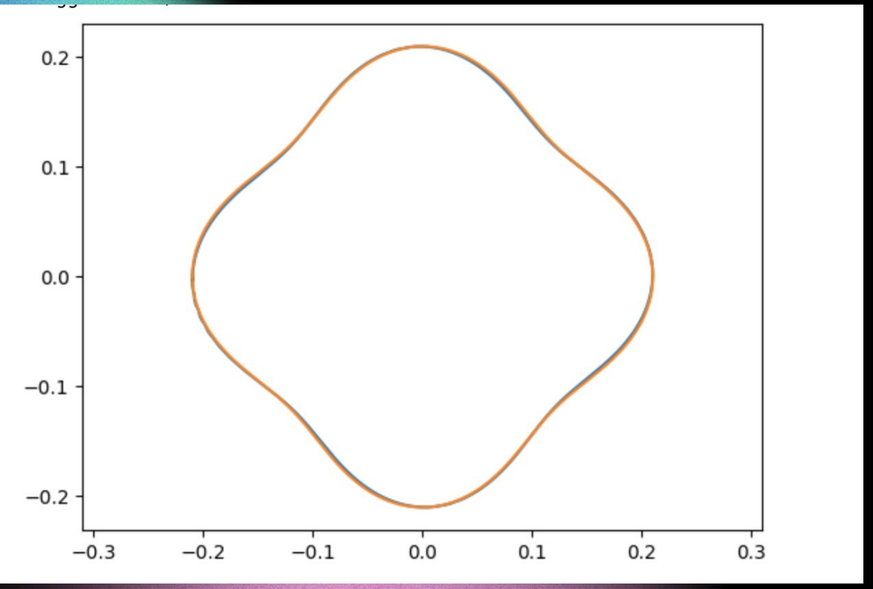
Recap: MuJoCo + OpenAI Gymnasium

OpenAI: Simulate Movements
MuJoCo: Multi-joint robotic arm to test simulations

Gymnasium

Video Demonstration

Utilizing RoboHive & Gymnasium for Inverse Modular Kinematics




```
MSE_x: 3.0712894331631748e-06
MSE_y: 3.0922587876781538e-06
MSE: 6.1635482208413286e-06
```

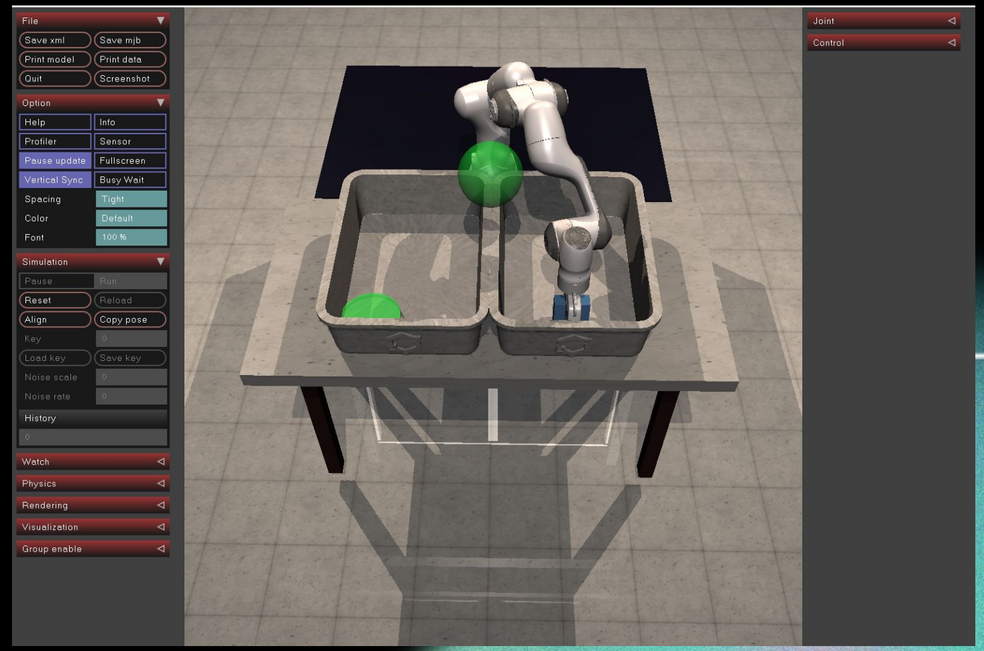
```python
ik_result3 = qpos_from_site_pose(
    physics=sim,
    site_name="end_effector",
    target_pos=box_pos + np.array([0, 0, .5]),  # change!
    target_quat=np.array([0, 1, 0, 0]),  # change!
    inplace=False,
    regularization_strength=1.0,
)

ik_result4 = qpos_from_site_pose(
    physics=sim,
    site_name="end_effector",
    target_pos=target_pos + np.array([0, 0, .5]),  # change!
    target_quat=target_quat,  # change!
    inplace=False,
    regularization_strength=1.0,
)

ik_result5 = qpos_from_site_pose(
    physics=sim,
    site_name="end_effector",
    target_pos=target_pos,  # change!
    target_quat=target_quat,  # change!
    inplace=False,
    regularization_strength=1.0,
)

ik_result6 = qpos_from_site_pose(
    physics=sim,
    site_name="end_effector",
    target_pos=target_pos + np.array([0, 0, .5]),  # change!
    target_quat=target_quat,  # change!
    inplace=False,
    regularization_strength=1.0,
)
```

```python
waypoints3 = generate_joint_space_min_jerk(
    start=ik_result2.qpos[:ARM_nJnt],
    goal=ik_result2.qpos[:ARM_nJnt],
    time_to_go=horizon,
    dt=sim.model.opt.timestep,
)

waypoints4 = generate_joint_space_min_jerk(
    start=ik_result2.qpos[:ARM_nJnt],
    goal=ik_result3.qpos[:ARM_nJnt],
    time_to_go=horizon,
    dt=sim.model.opt.timestep,
)

waypoints5 = generate_joint_space_min_jerk(
    start=ik_result3.qpos[:ARM_nJnt],
    goal=ik_result4.qpos[:ARM_nJnt],
    time_to_go=horizon,
    dt=sim.model.opt.timestep,
)

waypoints6 = generate_joint_space_min_jerk(
    start=ik_result4.qpos[:ARM_nJnt],
    goal=ik_result5.qpos[:ARM_nJnt],
    time_to_go=horizon,
    dt=sim.model.opt.timestep,
)
```

Issues we ran into

Robohive installation (needed to change vim.rc, install additional libraries, and change directories not provided in instructions)

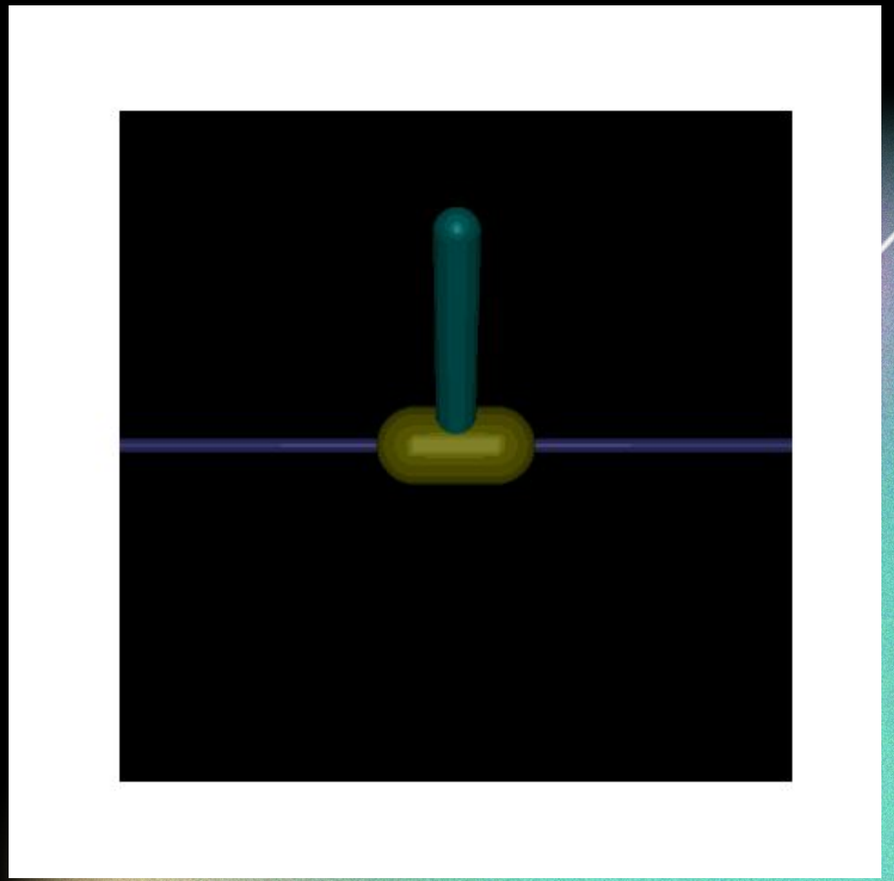
We kept knocking over block



Further Enhancements to Robot

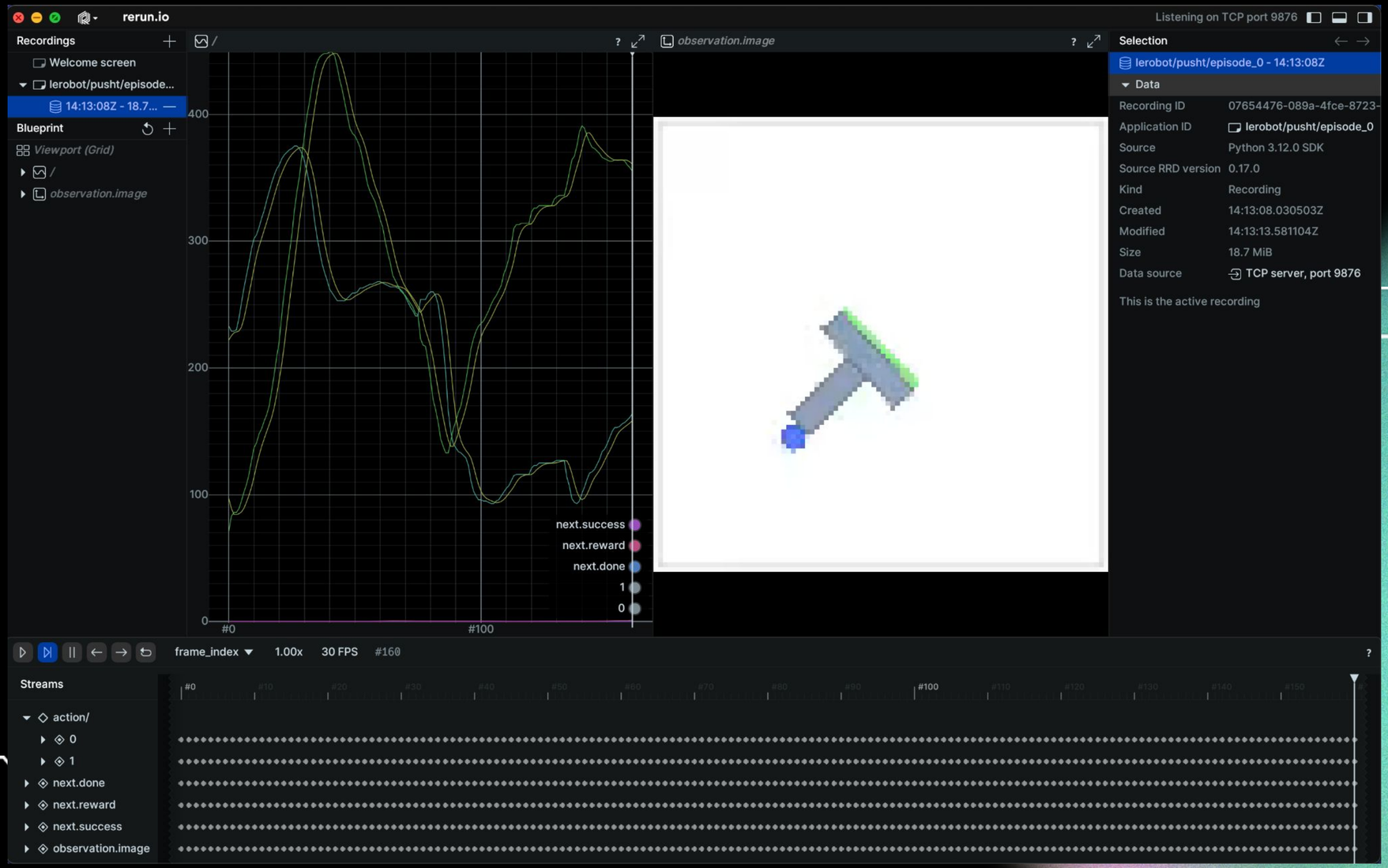
In order to avoid collisions of the box with either boundary or the wall in between, we have implemented RL models like LeRobot to mitigate this risk.

This has been possible through applying a motion planning algorithm.



```python
from torch.profiler import profile, record_function, ProfilerActivity

def trace_handler(prof):
    prof.export_chrome_trace(f"tmp/trace_schedule_{prof.step_num}.json")

with profile(
    activities=[ProfilerActivity.CPU, ProfilerActivity.CUDA],
    schedule=torch.profiler.schedule(
        wait=2,
        warmup=2,
        active=3,
    ),
    on_trace_ready=trace_handler
) as prof:
    with record_function("eval_policy"):
        for i in range(num_episodes):
            prof.step()
```

HuggingFace LeRobot Rerun Viewer

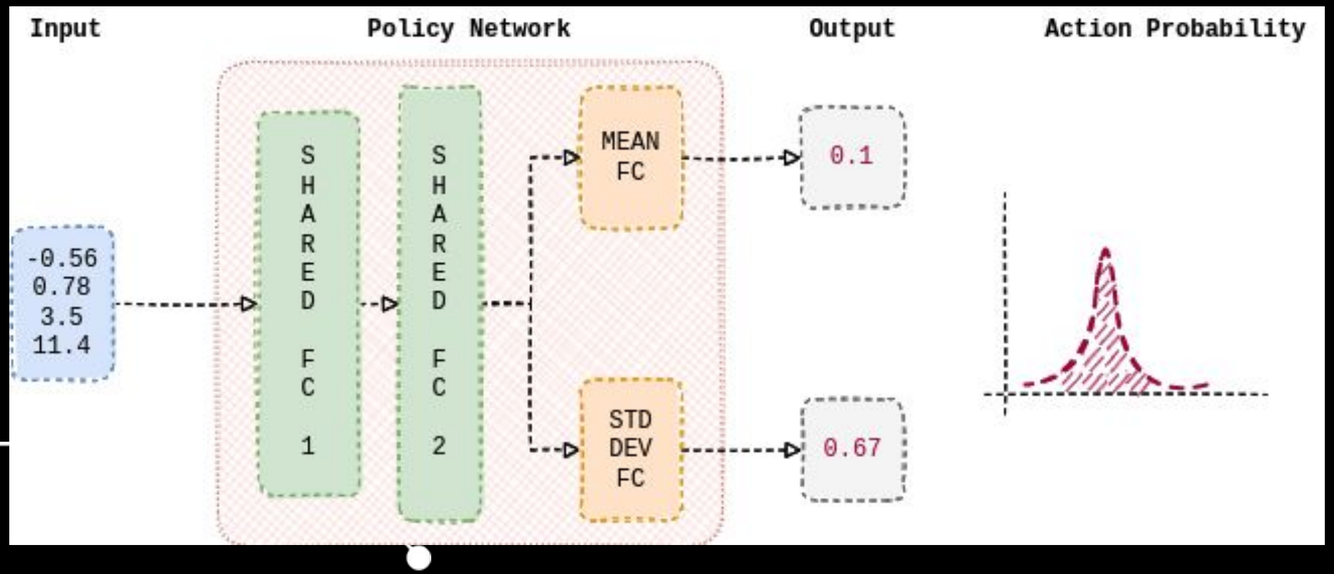


Video Demonstration of ML model

Improvements for high efficiency

Applying another Collision Detection algorithm utilizing REINFORCE for Mujoco

```python
def forward(self, x: torch.Tensor) -> tuple[torch.Tensor, torch.Tensor]:
    """Conditioned on the observation, returns the mean and standard deviation
     of a normal distribution from which an action is sampled from.

    Args:
        x: Observation from the environment

    Returns:
        action_means: predicted mean of the normal distribution
        action_stddevs: predicted standard deviation of the normal distribution
    """
    shared_features = self.shared_net(x.float())

    action_means = self.policy_mean_net(shared_features)
    action_stddevs = torch.log(
        1 + torch.exp(self.policy_stddev_net(shared_features))
    )

    return action_means, action_stddevs
```



Thank you!