Physic HOI Project

Code of Conduct

• If you use any open-source code, it is important to provide proper attribution and give credit to the

original authors. Proper citation and acknowledgment of sources are crucial in research, so be sure to

follow best practices and give credit where credit is due.

1 Basic Knowledge

1.1 Human-Object Interaction (HOI)

Human-Object Interaction (HOI) studies the interaction patterns between humans and objects, encompassing subtasks such as interaction detection, recognition, pose estimation, semantic understanding, action generation, and action control.

Methodologically, HOI research employs both large-scale vision-and-language data-driven frameworks and policy-learning approaches based on human demonstrations or instructions, such as imitation learning and reinforcement learning.

In terms of simulation platforms, MuJoCo, IsaacGym, IsaacSim, and Genesis provide critical support for collecting high-quality interaction examples, training interaction policies, and validating sim-to-real transfer. Unlike traditional object interaction tasks that focus on robotic arms, frontier research now explores fine-grained, whole-body or fingertip-level HOI on simulated humanoid robots (e.g., G1, H1) to investigate rich interaction scenarios and multimodal feedback.

1.2 Simulator

A simulator is a software tool that, powered by a physics engine, emulates the interaction between a robot and its environment, providing a virtual platform for algorithm development and testing.

With a simulator, developers can validate control algorithms, generate training data, and troubleshoot issues without relying on physical robot hardware, thereby reducing development costs and risks. Common robot simulators include MuJoCo, IsaacGym, IsaacSim, and Genesis, each supporting different physics and rendering engines to suit various application needs.

In this project, we use IsaacSim as our development simulator; for more details, please refer to NVIDIA’s official Isaac Lab and IsaacSim tutorials.

1.3 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a policy-gradient–based reinforcement learning algorithm proposed by OpenAI, designed to improve sample efficiency while maintaining training stability. PPO introduces a clipped surrogate objective to constrain policy updates, preventing excessively large parameter changes and ensuring the new policy stays close to the old one. At each optimization step, PPO maximizes the following objective:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}\big[\min\big(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\big],$$

Where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the new and old policies, $\widehat{A_t}$ is the estimated advantage, and ϵ controls the clipping range.PPO typically combines this clipped surrogate loss with a value-function loss, collects multiple trajectories in parallel environments, and performs several epochs of minibatch updates per rollout to balance training efficiency and convergence stability. It has become a benchmark algorithm for both continuous-control and discrete-action tasks. In embodied AI, PPO is among the most widely adopted methods. In this project, we will use PPO as our base algorithm.

2 Problem

2.1 Background

In this project, we will divide the learning of HOI into three levels: Basic, Intermediate, and Advanced, corresponding to three different skill sets:

Basic: The ability to get familiar with the relevant tools of a field, reproduce simple examples from related project code, and fine-tune parameters.

Intermediate: The ability to take a project and, based on related work, fine-tune and add code to meet specific requirements.

Advanced: The ability to build a complete pipeline and accomplish the given task independently.

The specific tasks are as follows:

2.2 Basic Task:Install and Get Started with Your Simulator

2.2.1 Install and Launch Your Simulator

Learn to install IsaacSim and IsaacLab, and run the startup program to open a

blank interface. You can refer to the Welcome to Isaac Lab! — Isaac Lab Documentation for guidance.

2.2.2 Use Isaac Lab Framework for Simple Locomotion

Learn to use the IsaacLab framework and try implementing locomotion for the Humanoid,H1, and G1 robots. You can refer to"source/isaaclab_tasks/isaaclab_tasks/direct/humanoid/humanoid_env.py" for implementation. The reason we choose the DirectEnv framework for this tutorial is that it serves as the most basic template, and the logic of many subsequent projects will follow a similar approach.

2.3 Intermediate Task:Implement HOI Tasks with RL

2.3.1 Clone and Run ProtoMotions

Visit https://github.com/NVlabs/ProtoMotions and run the checkpoint to ensure it works properly.

2.3.2 Add Objects to the Scene and Train Using PPO

Learn to add objects to the scene and then implement the HOI tasks based on ProtoMotions using PPO from the HOI benchmark objects.

2.4 Implement HOI Tasks Using AMP, MaskedMimic, and Others

2.4.1 Build Your Own Motion and Object Dataset

Collect, process, and use the AMASS dataset, and perform data preprocessing: unify frame rates, align the world coordinate system, and remove abnormal frames.

2.4.2 Implement Your HOI Tasks Based on ProtoMotions

Refer to algorithms from papers like MaskedMimic, AMP, and others, and use your dataset to build your own pipeline to implement your HOI tasks.

3 Reference

1. IsaacSim Tutorial: https://isaac-sim.github.io/IsaacLab/main/index.html
2. Isaaclab Tutorial: https://docs.isaacsim.omniverse.nvidia.com/4.5.0/index.html
3. Protomotions Code: https://github.com/NVlabs/ProtoMotions
4. AMASS: https://amass.is.tue.mpg.de/
5. [ACM SIGGRAPH Asia 2024]MaskedMimic: Unified Physics-Based Character Control Through Masked Motion Inpainting
6. [Transactions on Graphics (Proc. ACM SIGGRAPH 2021)]AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control
7. Proximal Policy Optimization Algorithms