ES158 Lab: Upkie Pendulum Environment

Your Name

November 23, 2025

1 Environment Description

The Upkie pendulum environment is a Gymnasium wrapper that transforms the Upkie humanoid robot into a wheeled inverted pendulum system. This simplification enables focused study of balance control while abstracting away the complexity of full humanoid dynamics.

1.1 System Dynamics

The environment constrains the robot's legs to remain straight, effectively reducing the system to a single rigid body (the base) balanced on two wheels. The robot behaves as a wheeled inverted pendulum, where the control objective is to maintain an upright posture by commanding wheel velocities.

1.2 Observation Space

The observation space is a 4-dimensional continuous vector:

$$o = \begin{bmatrix} \theta \\ p \\ \dot{\theta} \\ \dot{p} \end{bmatrix} \tag{1}$$

where:

- θ is the pitch angle of the base with respect to the world vertical, in radians. Positive values indicate forward lean.
- p is the position of the average wheel contact point, in meters.
- $\dot{\theta}$ is the body angular velocity of the base frame along its lateral axis, in radians per second.

- $\dot{p}$ is the velocity of the average wheel contact point, in meters per second.

The observation space is not normalized, and full spine observations remain available in the `info` dictionary returned by `reset()` and `step()`.

1.3 Action Space

The action space is a 1-dimensional continuous vector:

$$a = \left[\dot{p}^*\right] \in \mathbb{R} \tag{2}$$

where $\dot{p}^*$ is the commanded ground velocity in m/s. The action space is bounded by:

$$a \in [-a_{\max}, +a_{\max}] \tag{3}$$

where $a_{\max}$ is the maximum ground velocity (default: 1.0 m/s). This action is internally converted to wheel velocity commands using:

$$\omega_{\text{wheel}} = \frac{\dot{p}^*}{r_{\text{wheel}}} \tag{4}$$

where ω_{wheel} is the wheel angular velocity (rad/s) and r_{wheel} is the wheel radius from the robot configuration. A practical range for ground velocities is $[-1, 1]$ m/s, though the environment allows for higher values by adjusting $a_{\max}$.

1.4 Reward Function

The reward function is designed to encourage stable balancing behavior. It penalizes deviations from the ideal balanced state ($\theta = 0$, $\dot{\theta} = 0$, $\dot{p} = 0$) and large control actions. The reward at each timestep is computed as:

$$r_t = 1.0 - \left(w_\theta \theta_t^2 + w_{\dot{\theta}} \dot{\theta}_t^2 + w_{\dot{p}} \dot{p}_t^2 + w_a |a_t|^2 \right) \tag{5}$$

where:

- θ_t is the pitch angle at timestep t (rad)
- $\dot{\theta}_t$ is the angular velocity at timestep t (rad/s)
- $\dot{p}_t$ is the linear velocity at timestep t (m/s)
- a_t is the action (commanded ground velocity) at timestep t (m/s)

- $|a_t|$ denotes the absolute value of the action

The reward weights are:

$$w_\theta = 0.5 \tag{6}$$
$$w_{\dot{\theta}} = 0.1 \tag{7}$$
$$w_{\dot{p}} = 0.1 \tag{8}$$
$$w_a = 0.05 \tag{9}$$

The reward is then clamped to ensure it remains non-negative:

$$r_t = \max(0.0, r_t) \tag{10}$$

When the robot is perfectly balanced with $\theta_t = 0$, $\dot{\theta}_t = 0$, $\dot{p}_t = 0$, and $a_t = 0$, the reward achieves its maximum value of $r_t = 1.0$. Deviations from this ideal state result in reduced rewards, with quadratic penalties ensuring that larger deviations are penalized more heavily. The action penalty term $w_a|a_t|^2$ encourages smooth control by penalizing large control inputs.

1.5 Termination Conditions

The environment terminates an episode (sets terminated = True) under the following conditions, checked at each timestep t:

1. **Fall detection**: The episode terminates if the absolute pitch angle exceeds the fall threshold:

   $$|\theta_t| > \theta_{\text{fall}} \tag{11}$$

   where $\theta_{\text{fall}} = 1.0$ rad by default. This condition indicates the robot has fallen over and cannot recover.

2. **Excessive position drift**: The episode terminates if the robot drifts too far from its initial position:

   $$|p_t - p_0| > p_{\max} \tag{12}$$

   where p_t is the current position, p_0 is the initial position recorded at reset, and $p_{\max} = 5.0$ m. This prevents the robot from wandering indefinitely and encourages position maintenance while balancing.

3. **Excessive angular velocity**: The episode terminates if the angular velocity exceeds a safety threshold:

$$|\dot{\theta}_t| > \dot{\theta}_{\max} \tag{13}$$

where $\dot{\theta}_{\max} = 10.0$ rad/s. This indicates the robot is spinning too rapidly to recover and prevents learning from unstable trajectories.

4. **Excessive linear velocity**: The episode terminates if the linear velocity exceeds a safety threshold:

$$|\dot{p}_t| > \dot{p}_{\max} \tag{14}$$

where $\dot{p}_{\max} = 2.0$ m/s. This indicates the robot is moving too fast to maintain control and prevents unsafe high-speed behavior.

Additionally, episodes are truncated (sets truncated = True) after a maximum number of steps:

$$t \geq t_{\max} \tag{15}$$

where $t_{\max} = 300$ steps. This time limit ensures reasonable episode lengths during training and prevents episodes from running indefinitely.

The termination conditions are checked in the following order: fall detection is checked first, followed by the additional termination conditions (position drift, angular velocity, linear velocity). If any condition is met, the episode terminates immediately. The time limit truncation is checked independently and can occur even if no termination condition is met.

1.6 Implementation Details

The environment is implemented as a Gymnasium wrapper around the base `UpkieEnv` class. The wrapper:

- Maintains leg joints in a neutral configuration using low-pass filtering
- Converts ground velocity commands to wheel velocity commands
- Extracts the simplified 4D observation from the full spine observations
- Computes rewards and checks termination conditions at each step

The environment integrates with the Upkie spine backend, which handles all physics simulation. This design allows the same environment code to work with both simulation and real robot hardware.

2 Design Rationale

The reward function design balances several objectives:

- **Stability**: The quadratic penalties on θ, $\dot{\theta}$, and $\dot{p}$ encourage the robot to maintain an upright, stationary pose.
- **Smooth control**: The action penalty encourages the policy to use smooth, moderate control inputs rather than aggressive jerky motions.
- **Learning signal**: The reward structure provides a clear gradient toward the desired behavior, with maximum reward at the ideal state.

The termination conditions are designed to:

- **Prevent unsafe states**: Early termination when the robot falls or moves dangerously fast prevents learning from unstable trajectories.
- **Encourage focused learning**: By terminating episodes that drift too far, the policy learns to maintain position while balancing.
- **Enable efficient training**: Reasonable episode lengths (300 steps) allow for efficient exploration and learning.

This environment design enables reinforcement learning algorithms to learn effective balancing policies for the wheeled inverted pendulum system.