Progress Report: Deep Reinforcement Learning for Humanoid Locomotion from an Image-Defined Initial Pose

Anirudh Arrepu *CS23B008* | Aryan Chauhan *CS23B009* | Bairu Srivally *CS23B010*

1. Pose Extraction using MediaPipe

Initially, we attempted to extract a 25-body keypoint skeleton using OpenPose. However, after several dependency issues and incompatibility errors, the OpenPose API could not be successfully integrated. To overcome this, we switched to **MediaPipe**, which provides a 33-body keypoint skeleton.

The 33-point MediaPipe skeleton is mapped to the 25-point format required for the humanoid initialization. Several approaches were attempted to utilize OpenPose, but each failed due to one or another dependency conflict. Hence, for this iteration, we decided to proceed with MediaPipe.

Since MediaPipe inherently extracts the skeleton of the person occupying the largest area in the image, no additional person-selection pipeline was required. This simplified the initial preprocessing stage.

2. Angle Computation

Angles are calculated across all the joints as follows:

```
angles = {
    "neck_tilt": safe_angle(MID_HIP, NECK, NOSE),
    "r_shoulder": safe_angle(NECK, R_SHOULDER, R_ELBOW),
    "r_elbow": safe_angle(R_SHOULDER, R_ELBOW, R_WRIST),
    "l_shoulder": safe_angle(NECK, L_SHOULDER, L_ELBOW),
    "l_elbow": safe_angle(L_SHOULDER, L_ELBOW, L_WRIST),
    "r_hip": safe_angle(MID_HIP, R_HIP, R_KNEE),
    "r_knee": safe_angle(R_HIP, R_KNEE, R_ANKLE),
    "l_hip": safe_angle(MID_HIP, L_HIP, L_KNEE),
    "l_knee": safe_angle(L_HIP, L_KNEE, L_ANKLE),
    "r_ankle": safe_angle(R_KNEE, R_ANKLE, 22),
    "l_ankle": safe_angle(L_KNEE, L_ANKLE, 19),
}
```

Angles are computed using the `arctan2` function applied to the 25 keypoints, ensuring stability and normalization in $[-\pi, \pi]$.

$$\theta = \arctan 2(p_3^y - p_2^y, p_3^x - p_2^x) - \arctan 2(p_1^y - p_2^y, p_1^x - p_2^x)$$

3. Custom Environment in PyBullet

A custom environment has been developed that abstracts the Gymnasium environment. This environment initializes the humanoid using a URDF file. For the current phase, the URDF file has

been created only for the **lower body joints** to simplify training and focus on initial locomotion learning. This simplification allows the agent to learn bipedal motion before extending to full-body dynamics.

The `step()` function generates the simulation in PyBullet and abstracts the Gym environment's step function. The initial pose is partially integrated- only the lower body angles are derived from the extracted pose, while the upper body joints are initialized to zero. Later stages will include the full-body initialization.

4. DQN Agent and Learning Framework

The agent utilizes a **Deep Q-Network (DQN)** framework for learning a walking policy. The DQN replaces the Q-table with a deep neural network that approximates the Q-function:

$$Q(s, a; \theta) \approx \text{expected return of taking action } a \text{ in state } s$$

where θ represents the network parameters.

Action Space

$$\mathcal{A} = [-1, 1]^{n_{\text{joints}}}$$

The action space is continuous, representing torque applied to each joint. These torques are discretized into bins for DQN compatibility.

5. Reward Function Engineering

The reward function is defined as:

$$R_t = w_{\text{vel}} \cdot r_{\text{vel}} + w_{\text{live}} \cdot r_{\text{live}} - w_{\text{energy}} \cdot r_{\text{energy}} + w_{\text{survival}} \cdot r_{\text{survival}}$$

where:

- r_{vel}: reward for forward velocity (x-direction of center of mass)
- r_{live}: small positive reward for staying upright (height above threshold)
- r_{energy}: penalty for sum of squared torques (energy efficiency)
- r_{survival}: exponential reward based on step count to encourage longer survival

A large penalty is applied if the humanoid falls below a threshold height, which also terminates the episode. This design ensures that the agent learns to move with stability rather than abruptly applying high torques to generate movement. The survival term counterbalances the energy penalty to prefer sustained locomotion over brief, low-energy stillness.

6. Training Loop and Observations

To reduce the complexity of the agent's action space, the selected torque values are scaled by a factor of 40 (obtained by trial and error) to modify the steps taken by the agent. Currently, the humanoid performs small scooching motions- indicating progress toward stable walking. To encourage larger, more natural strides, we can extend the action space from $[-1, 1]$ to $[-2, 2]$ and reduce the energy penalty by half, which increases movement amplitude while maintaining stability.

The training loop performs training for several initial poses such that the agent learns stable walking from any given initial lower body configuration.

```
for pose in poses:
    for ep in num_episodes:
```

```
while not done:
    update DQN
```

The training loop uses the computed reward to update the DQN. The agent's ϵ-greedy policy ensures exploration during early episodes and converges toward exploitation as learning stabilizes.

7. Python Library/ Package

This pipeline has been converted to a python package. A `setup.py` file is included to allow installation using `pip install -e .`

8. Future Work

- Complete the full-body URDF and enable full 25-joint initialization rather than lower body only.
- Add gait analysis metrics (stride length, symmetry, energy cost) [6]
- Switch to openpose if mapping full body to urdf fails from mediapipe's configurtaion.

References

[1] C. Lugaresi et al., *MediaPipe: A Framework for Building Perception Pipelines*, arXiv:1906.08172, 2019.

[2] Z. Cao et al., *OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields*, IEEE TPAMI, 2019.

[3] V. Mnih et al., *Human-Level Control through Deep Reinforcement Learning*, Nature, 518:529–533, 2015.

[4] E. Coumans and Y. Bai, *PyBullet, a Python module for physics simulation for games, robotics and machine learning*, GitHub, 2016–2024.

[5] Y. Tassa et al., *DeepMind Control Suite*, arXiv:1801.00690, 2018.

[6] X. B. Peng et al., *DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills*, ACM TOG 37(4), 2018.

[7] N. Heess et al., *Emergence of Locomotion Behaviours in Rich Environments*, arXiv:1707.02286, 2017.

[8] OpenAI et al., *Learning Dexterous In-Hand Manipulation*, arXiv:1808.00177, 2018.