Controllable Humanoid and Ant using Reinforcement Learning with Diverse Rewards

Benet Oriol Sabat Haniyeh Ehsani Oskouie
Avalon Vinella Mohsen Fayyaz Chae Yeon Seo
Computer Science Department, University of California, Los Angeles
{benet, haniyeh, avinella, mohsenfayyaz, chaeyeon}@cs.ucla.edu

Abstract

In this project, we explore the application of reinforcement learning (RL) to train controllable humanoid and ant agents that can switch between different movement modes and styles. Using the Gymnasium simulation environment and the Soft Actor-Critic algorithm (SAC), we developed models capable of executing a variety of locomotive tasks with diverse rewards. The trained agents can adapt to real-time user inputs, offering a flexible and robust simulation for autonomous locomotion.

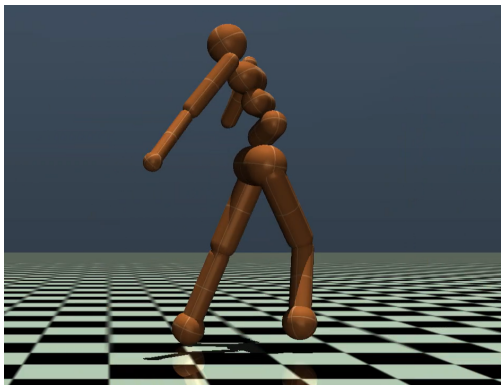


Figure 1: Controllable humanoid walking towards a specified direction.

1 Introduction

Autonomous locomotion is a pertinent topic in artificial life, as the ability for an agent to independently move is key to allowing it to interact with its environment. We explore the potential of reinforcement learning (RL) to train agents to perform diverse locomotive tasks. In particular, we develop two controllable models, an ant and a humanoid, that are able to switch between different modes and styles of movement at the user's real-time command. We leverage the Gymnasium simulation environment (Towers et al., 2023) in concert with the Soft Actor-Critic (SAC) (Haarnoja et al., 2018) learning algorithm to train these autonomous agents for a variety of locomotive tasks through the careful formulation of task-specific reward functions. These task-specific models are then combined into one cohesive, user-driven simulation that smoothly transitions between gaits through the inclusion of an additional high-level controller that manages gait selection and transitions.

2 Simulation

The simulation of autonomous locomotion in this project leverages the Gymnasium Mujoco environments, specifically focusing on the Ant and Humanoid models. Gymnasium Mujoco provides a robust and flexible platform for simulating and training RL agents, offering a variety of predefined environments that are widely used in the RL community.

2.1 Ant

The Ant environment in Gymnasium Mujoco is designed to simulate a quadrupedal agent. The agent consists of a body with four legs, each equipped with two controllable joints, resulting in an eight-dimensional action space. This configuration allows the ant to move in a highly dynamic and flexible manner.

Action Space in the Ant environment is defined as a Box(-1.0, 1.0, (8,), float32). Each action represents the torques applied at the hinge joints, allowing the agent to control the movement of the robot's legs. This high-dimensional action space requires the agent to learn complex control strategies to coordinate the movements of all four legs effectively, enabling the ant to navigate its environment efficiently.

Observation Space provides a detailed snapshot of the ant's state, essential for effective reinforcement learning. It includes the positions and velocities of all joints, as well as the position and velocity of the agent's body. Additionally, it may include

information about contact points with the ground, aiding the agent in understanding its interaction with the environment. Specifically, the observation space consists of 13 elements representing the positions of the robot's body parts (excluding the torso's x- and y-coordinates by default), 14 elements capturing the velocities of these parts, and 78 elements describing the external forces and torques acting on the body parts. This results in a total of 105 elements in the default observation space, which expands to 107 elements if configured to include the torso's x- and y-coordinates. Regardless of this configuration, these coordinates are always available in the info dictionary. This comprehensive set of observations allows the agent to understand and control the robot's complex movements effectively.

2.2 Humanoid

The Humanoid environment in Gymnasium Mujoco is designed to simulate a 3D bipedal robot that mimics human locomotion. The humanoid robot consists of a torso (abdomen), legs with three segments each (thigh, shin, foot), and arms with two segments each (upper arm, forearm). Additionally, tendons connect the hips to the knees, enhancing the realism of the simulation.

Action space in the Humanoid environment is defined as a Box(-0.4, 0.4, (17,), float32). Each action corresponds to the torques applied at the robot's hinge joints, allowing precise control over the movements of its limbs. This high-dimensional action space requires the agent to learn complex control strategies to coordinate its limbs effectively. By applying appropriate torques, the agent can generate various walking patterns and adapt to different scenarios, making the Humanoid environment a challenging and rich testbed for reinforcement learning algorithms.

Observation space in the Humanoid environment is extensive, consisting of 348 elements by default. These elements include 22 position values (*qpos*) representing the positions of the robot's body parts, 23 velocity values (*qvel*) indicating the derivatives of these positions, and 130 elements (*cinert*) related to the mass and inertia of the body parts relative to the center of mass. Additionally, there are 78 elements (*cvel*) for the center of mass-based velocities, 17 elements (*qfrc_actuator*) for the constraint forces generated at each joint, and another 78 elements (*cfrc_ext*) for external forces acting on the body parts. If configured to include the x- and y-coordinates of the torso, the observation space expands to 350 elements. This comprehensive observation space provides the agent with detailed information about the robot's state, enabling it to understand and control its complex movements effectively. The inclusion of these detailed observations ensures that the agent can learn to maintain balance and achieve efficient locomotion in a highly dynamic environment.

3 Learning Algorithms

3.1 Reward Functions

Both Ant and Humanoid environments have a similar baseline reward.

$$Reward = v_x + R_{healty} + R_{control} \quad (1)$$

where v_x is the velocity in the x direction, R_{healty} refers to the model falling or not falling down (binary), $R_{control}$ is a regularization term that incentivizes low force values in the joints.

We address three limitations in this reward value:

- v_x makes the model go to the right and not in any other direction.
- v_x makes the model go as fast as possible in that direction, as opposed to a reasonable or controllable speed.
- R_{healty} can make the model optimize for staying still, minimizing the possibility of falling down.

4 Experiments and Results

4.1 Controlling ant direction

In order to allow inference-time control on the direction of the model, we make two modifications. First, we add 2 components to the input of the policy model, the vector $\boldsymbol{v}_{target} = (\cos\theta, \sin\theta),\ \theta \in [0, 2\pi)$, which represents the user-given direction of the model. This allows the policy to decide actions as a function of some user control, as opposed to being always in the same direction. Second, we modify the reward function such that.

$$Reward = \boldsymbol{v} \cdot \boldsymbol{v}_{target} + R_{healty} + R_{control} \quad (2)$$

This incentivized the model to maximize the dot product between the model's velocity given by the displacement of the center of mass and the target velocity given by the user. Finally, we modify the training procedure such that the target velocity is

randomized by sampling from a direction angle distribution at each training step.

In our experiments, we use different angle distributions to progressively go from more simple to more complex cases. Formally, we uniformly sample θ from the set formed by $\{i\frac{2\pi}{N}\}, i \in \{0, ..., N-1\}$. In other words, we divide a circle into N directions and sample them at random during training. If the model is properly trained for N direction, this allows to control the direction of the ant over these N directions during inference. We train an ant model for each different value of N. We use $N \in \{1, 2, 4, 8\}$, where $N = 1$ would be equivalent to the baseline. We also implement the equivalent of $N \to \infty$ by sampling $\theta \sim \mathcal{U}(0, 2\pi)$, where $\mathcal{U}(\cdot)$ is the uniform distribution.

Initial results show that the ant stalls at a specific place without moving, we suspect this has to do with R_{healty} which encourages the ant to stall still and not fall. We can see we can adapt as follows:

$$Reward = \boldsymbol{v} \cdot \boldsymbol{v}_{target} + 0.5\, R_{healty} + R_{control} \quad (3)$$

By weighting that reward term to have less impact, the model does not stall anymore while keeping its balance and is more incentivized to explore more trajectories.

With this implementation detail, we are able to train all models with different distributions of θ. We show how our ant model trained for "any direction" can indeed take any direction during inference time. Moreover, while it has been trained to follow the same direction at each trajectory, it shows the capability to change directions within the same inference trajectory, allowing for arbitrary real-time control of the model's direction.

4.2 Ant at Varied Velocities

To control the velocity of the ant on the fly, we introduce target velocity terms for both x and y directions. The ant aims to match these target velocities in its movement. We define the forward reward function as follows:

$$\begin{aligned} &Reward_{forward} \\ &= -\{(\boldsymbol{v}_x - \boldsymbol{v}_{x\,target})^2 + (\boldsymbol{v}_y - \boldsymbol{v}_{y\,target})^2\} \end{aligned} \quad (4)$$

The reward increases when the ant's x velocity is close to the x target velocity and its y velocity is close to the y target velocity. We test this reward function in various scenarios. When the x target velocity is set to 1 and the y target velocity to 0, the ant maintains an x velocity of 1 and a y velocity of 0. However, when the x target velocity ranges from 1 to 5 and the y target velocity is 0, neither velocity converges as well as in the first scenario, resulting in the ant remaining in place. Additionally, when we combine this reward with varied direction rewards, the ant similarly moves in the same place.

4.3 Humanoid in Varied Directions

To control the direction of the humanoid while walking, we introduce a two-dimensional target direction vector as an additional observation in the environment. This vector is included along with the default observations as input to the neural network. The target direction vector signifies the intended direction of movement. Additionally, we incorporate the cosine similarity of the humanoid's velocity vector with this target direction as an extra reward for the RL algorithm. The cosine similarity between two vectors A and B, where the angle between them is θ, is defined as follows:

$$\begin{aligned} \cos(\theta) &= \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \\ &= \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \end{aligned} \quad (5)$$

This cosine similarity value ranges from -1 (when the vectors are opposite) to 1 (when they align perfectly). A higher similarity indicates better alignment between the velocity and target direction, leading to higher rewards that the RL algorithm seeks to maximize. In practice, to simplify learning given the complexity of the 3D humanoid's observation and action space, we limit the target direction to four main directions $([1, 0], [-1, 0], [0, 1], [0, -1])$. The result of this part is available in **'humanoid_directions.mp4'**, which demonstrates how the humanoid's direction is controlled during walking using the described method.

4.4 Humanoid at Varied Velocities

In addition to directional control, we introduce new reward terms to encourage the humanoid to stop walking or to walk at a higher velocity. For stopping the humanoid, we define a reward term as follows: if we denote the 2D velocity vector of the humanoid as V, we subtract the squared norm of

V from the reward:

$$Reward_{Still} \coloneqq Reward - \|V\|^2$$

This formulation increases the reward when the model keeps the humanoid still. Conversely, to incentivize faster walking, we use the following reward term:

$$Reward_{Fast} \coloneqq Reward + \|V\|^2$$

Here, the reward increases with the square of the velocity norm, encouraging faster movement.

During training, as the model learns to walk at higher speeds, we take various steps from the training checkpoints to observe the humanoid's behavior at different velocities. The results of these experiments, showcasing the humanoid moving at different speeds, are presented in **'humanoid_speed.mp4'**.

4.5 Crouching Humanoid

In addition to controlling the velocity of the humanoid model, we also explore stylistic modifications to the learned gait. We experiment with the task of crouching locomotion, in which the model must lower its center of gravity while moving forward. Unlike the previous trials, the main goal of this task is to achieve a specific visual style of locomotion. The reward function must be finely tailored towards the desired form; while the model could learn to move effectively by meeting some loose guidelines of a "crouch" by remaining close to the ground, it might not quite match what we would expect a crouch to look like. Thus, our reward function for this task is much more restrictive and includes more penalties than the default reward.

For simplicity, we only train this model to move in four directions with no speed specification; our emphasis is to achieve the crouching form with limited control.

Based on our trials, we include the following rewards and penalties:

- \+ Forward velocity (as in Section 4.3)
- – Pelvis above height threshold
- – Bending torso too far forward or backward
- – Leaning torso to the sides

We additionally lower the "healthy" z range to [0.4, 2.0] to accommodate the crouching style. These reward considerations were incorporated to address form issues that we observed in our experiments. Without the bending penalties, the model would either completely lean backwards and walk on its knees or fold forward. The leaning penalty prevents the model from throwing its torso in the goal direction to increase its velocity rather than locomoting. We also chose to determine height based on the pelvis, rather than the torso, to encourage the model to bend its knees to lower itself instead of bending over.

We additionally observed that the model was more fragile to switching directions than the standing humanoid model. We thus altered the training episode slightly such that midway through, the target direction would be resampled; previously, the model only reset its direction at the beginning of episodes. This allowed the model to "practice" changing directions and remain upright at test time changes.

Our final model, presented in **'humanoid_crouching.mp4'**, resembles more of a lunge than a crouch; however, it is able to move, albeit slowly, with one knee bend and torso at a realistic angle. Given more training time or adjusted weights, the model could surely learn to bend both legs, as it currently requires the straight leg to keep its balance. This would also likely enable it to move at a higher velocity. This model also exhibits robust behavior in switching direction; we can visually see it adjusts its balance as the goal direction is reassigned, and it does not fall over.

5 Controllable Merged Models

Our final executable allows the user to control the speed, velocity, and style (standing versus crouching). At the user's command, the active model is switched to one trained for the specified speed and style; in this manner, we are able to train separate models with less complex tasks, while allowing the user breadth of control options. Since all of the models are trained in multiple directions, the direction can always be dictated. For the majority of standing models, the trajectories are similar enough that no explicit transitions are needed. Surprisingly, our model is able to transition from crouching to some of the upright walking speeds without any additional training or control. However, the other direction is much less successful, as the model immediately tries to achieve a crouching form by bending its hip and lifting its foot off the

```
action_list = {
    0: "hinge in the y-coordinate of the abdomen",
    1: "hinge in the z-coordinate of the abdomen",
    2: "hinge in the x-coordinate of the abdomen",
    3: "rotor between torso/abdomen and the right hip (x-coordinate)",
    4: "rotor between torso/abdomen and the right hip (z-coordinate)",
    5: "rotor between torso/abdomen and the right hip (y-coordinate)",
    6: "rotor between the right hip/thigh and the right shin",
    7: "rotor between torso/abdomen and the left hip (x-coordinate)",
    8: "rotor between torso/abdomen and the left hip (z-coordinate)",
    9: "rotor between torso/abdomen and the left hip (y-coordinate)",
    10: "rotor between the left hip/thigh and the left shin",
    11: "rotor between the torso and right upper arm (coordinate -1)",
    12: "rotor between the torso and right upper arm (coordinate -2)",
    13: "rotor between the right upper arm and right lower arm",
    14: "rotor between the torso and left upper arm (coordinate -1)",
    15: "rotor between the torso and left upper arm (coordinate -2)",
    16: "rotor between the left upper arm and left lower arm"
}
```

Figure 2: List of actions representing the torques applied at each hinge joint.

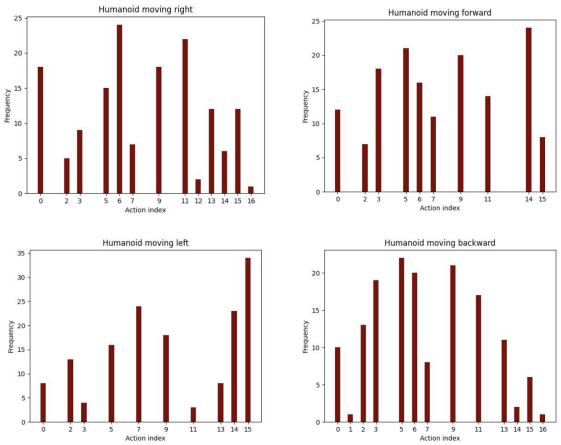


Figure 3: Frequency of the most influential actions.

ground. Similarly, the model struggles to stop from its "crazy walk" mode, as its momentum throws it off balance. These gaits, which are the most dissimilar to the "standard" walking form, would require additional control to smoothly transition between modes.

6 Analysing the Torques Applied at the Hinge Joints

Given the complicated behavior of the human model compared to the ant model, a comprehensive analysis of its behavior is necessary to ensure the accurate selection of actions at each step of the simulation. In this section, we provide a detailed analysis of human actions using activation figures.

6.1 Choice of actions over time

One good analysis is observing the choice of actions over time. The list of actions for the humanoid is shown in Figure 2. At each step, as the human model receives features from the observation space, it decides on a list of actions. Within the selected actions, there exists one particular action that induces the most substantial rotation. This particular action, which holds the highest influence in terms of rotation, is referred to as the "most influential action." In Table 1, we provide an illustration of the torques that result in the most significant rotations at each step for 40 steps of simulation. As shown, action 6 which corresponds to the rotor between the right hip/thigh and the right shin is the most frequently repeated action with the largest rotation when moving to the right. This observation holds true for other movement directions as well, suggesting a logical pattern. It is important to note that the beginning of these 40 steps occurs after the initial 1000 steps of testing the model in the simulation. It is crucial to undertake this approach because during the initial stages, the model may not demonstrate stable behavior.

6.2 Frequency of the most influential actions

Figure 3 illustrates the frequency of the most influential actions across 150 steps for various goals, including moving right, left, forward, and backward. Remarkably, actions involving the utilization of feet joints (actions 3 to 10) consistently emerge as the most repetitive actions across all goals. An

Table 1: The torques that caused the largest rotations in 40 steps of simulation.

Goal	Actions
Moving right	0, 13, 6, 7, 6, 6, 6, 6, 5, 6, 9, 6, 5, 9, 13, 2, 3, 3, 2, 3, 7, 3, 0, 0, 9, 9, 3, 7, 5, 6, 2, 6, 15, 9, 11, 2, 2, 9, 7, 7, 9
Moving left	14, 15, 7, 14, 5, 9, 14, 11, 3, 3, 9, 11, 2, 5, 9, 0, 9, 5, 7, 0, 14, 3, 5, 15, 3, 9, 7, 5, 15, 2, 15, 3, 9, 7, 13, 11, 9, 5, 3, 15, 15
Moving forward	2, 5, 5, 2, 11, 6, 6, 9, 5, 7, 9, 11, 11, 11, 0, 2, 6, 0, 9, 11, 9, 11, 2, 7, 6, 6, 2, 6, 14, 14, 0, 2, 9, 9, 6, 6, 0, 2, 7, 5, 0
Moving backward	0, 15, 2, 0, 2, 3, 11, 9, 0, 2, 9, 7, 9, 3, 11, 3, 4, 4, 11, 6, 15, 3, 0, 2, 3, 3, 0, 7, 14, 5, 9, 6, 6, 3, 6, 5, 5, 6, 9, 6, 15

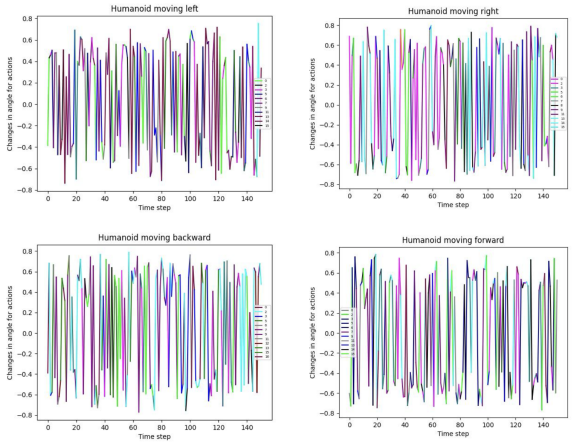
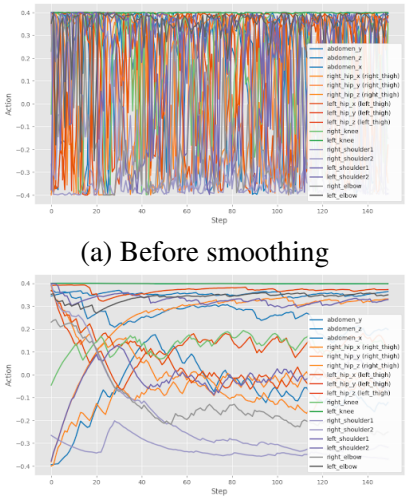


Figure 4: Actions with the largest rotations per each time step.



(a) Before smoothing

(b) After smoothing

Figure 5: Activation figures before and after smoothing.

interesting observation is that the movement of the humanoid's arms is also evident in certain goals. This occurrence may be attributed to the necessity of the humanoid to utilize its arms for maintaining balance and preventing falls during the execution of various tasks.

6.3 Largest rotations

The actions with the largest rotations at each time step are displayed in Figure 4. It can be seen that in nearly all instances, a different action is identified as the most influential. This pattern can be due to the potential risk of the humanoid falling if a single action is repeated continuously for multiple steps. Thus, the variation in the most influential action over time helps maintain stability.

6.4 Comparison between different torques

Figure 6 presents the activation figures showcasing the angle values of the torques over 800 steps, specifically focusing on the goal of moving to the right. The figures highlight that the angles of joints associated with the thighs, abdomen, and shoulders undergo frequent changes. The adjustments in the angles of the thigh and abdomen joints are essential for facilitating locomotion, while variations in the shoulder angles indicate the humanoid's efforts to maintain a stable standing pose. Activation figures for all of the actions in 150 steps of simulation are demonstrated in Figure 5. To facilitate the analysis of the plot, we applied a smoothing function that averages the current and previous angle values, employing a weight of 0.95. This smoothing technique allows us to observe certain joint angles converging to approximately -0.4 or 0.4, while other angles continue to exhibit fluctuations. The joints that fluctuate are more effective in enabling the movement of the humanoid in the desired direction.

7 Conclusions

This project demonstrated the effectiveness of reinforcement learning (RL) in training controllable humanoid and ant agents capable of seamlessly transitioning between various movement modes and styles. It was shown that the design and implementation of appropriate reward functions play a pivotal role in successfully training these models. Additionally, the findings revealed that the joints associated with the feet were not the sole contributors to the humanoid's ability to move in the specified direction. Other joints in the humanoid's body were also found to play significant roles in maintaining its standing pose.

References

Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ArXiv*, abs/1801.01290.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. 2023. Gymnasium.

A Appendix

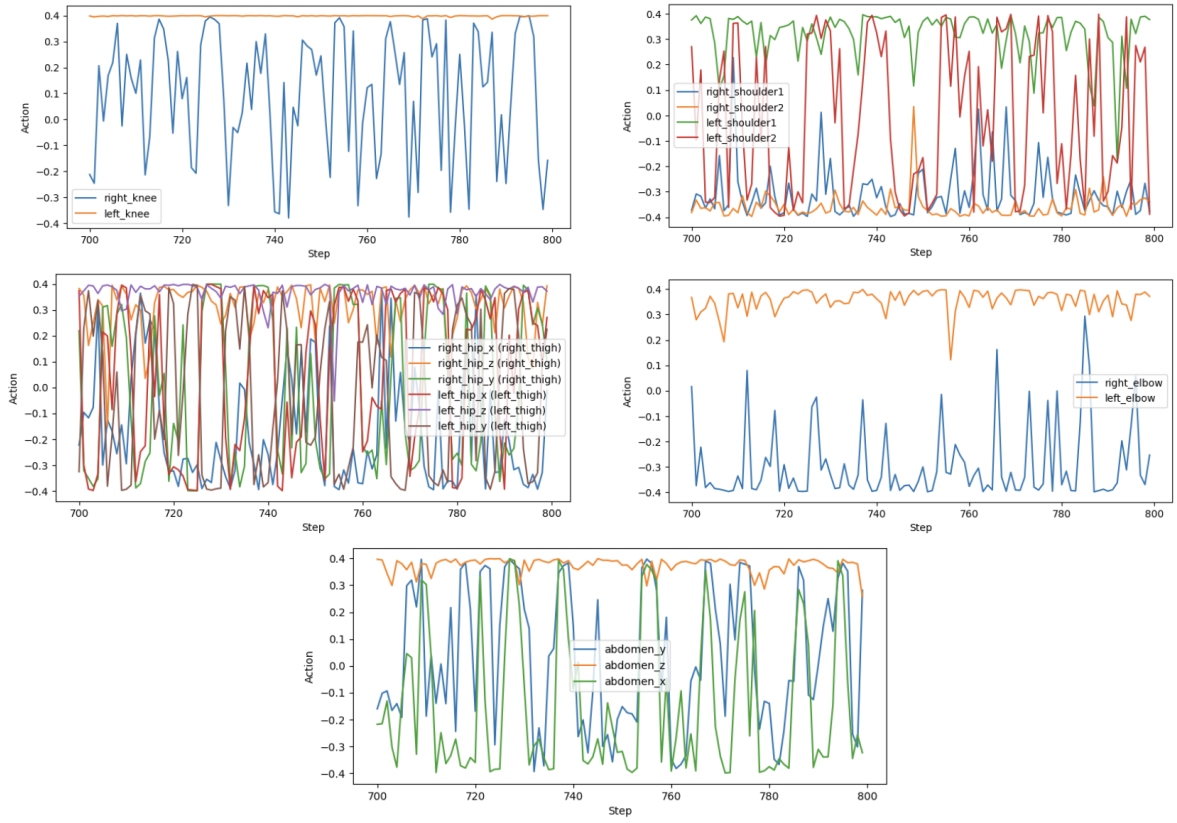


Figure 6: Angle values of the torques.