Comparative Analysis of stable ground pose of Cassie biped using SAC and TD3

Abhijit Kaluri, Sriram Jayakumar, Nagavarun Samala
kaluri.a@northeastern.edu, jayakumar.sr@northeastern.edu, samala.n@northeastern.edu

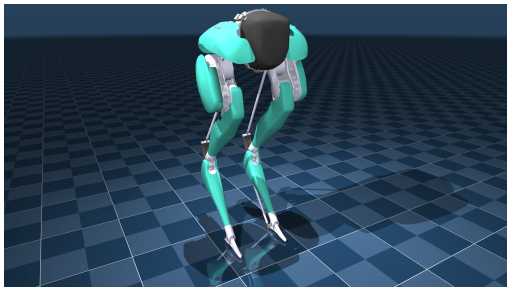

Fig. 1. Cassie

I. Introduction

In this project, we undertook the task of developing an algorithm to locomote our bipedal robot Cassie using Mujoco, a physics simulator and Myosuite, a simulation framework for locomotion which wraps and leverages Mujoco for high efficient simulation. The inpiration for this project was based on the Cassie robot model developed by a team of researchers at OSU to utilize the mechanical and muscular physics simulation libraries for the model. The robot consists of a hull and legs with multiple joints and their corresponding torques for manipulation. The use of SAC(Soft Actor Critic) algorithm aided in the task of achieving our objective. Our main goal was to address the challenge of enabling a humanoid robot to achieve stable pose ground contact, specifically the complex task of balancing to stand on it's own, within a physics-based simulator. This endeavor required us to build a custom environment from the ground up, prioritizing their capacity to tackle the intricate problem of human-like standing pose to simulate the biped robot.

II. Problem Statement

The pivotal problem of the project is that the robot needs to be trained to select actions to be able to achieve a stable ground pose. This utilizes SAC algorithm designed for continuous action spaces and where actor learns a policy and critic evaluates the actions taken by actor. This policy helps in encouraging exploration while maintaining some level of randomness in the policy. There are lapses in the reward functions which have to be tuned to obtain the desired reward ranges.

The Mujoco physics engine is used to obtain the fundamental physics and muscular simulation for multibody joint contacts and Myosuite dependency which is a framework enriched with built for contact dynamics of muscoskeletal motor control. The integration of Myosuite libraries to the Base V0 class which contains the action and observation space of multibody joint contacts to be utilized in RL algorithms to learn.

A. Choosing a Simulator

The choice of a simulator was predicated on its ability to effectively handle multibody joint simulations and its compatibility with reinforcement learning frameworks.

We opted for MuJoCo, an acronym for Multi-Joint dynamics with Contact. MuJoCo is a versatile physics engine designed specifically for scientific applications in fields such as robotics, biomechanics, and machine learning. As the version of MuJoCo previously used had been deprecated, we accessed it through MyoSuite. MyoSuite 2.0 is a framework enriched with contact dynamics for musculoskeletal motor control, which serves as a wrapper for MuJoCo, facilitating access to its features.

B. Selection of a High Fidelity XML Model

We opted to choose a high-fidelity XML model of Cassie, which was developed by the Robotics Department at Oregon State University. This model, which has been open-sourced, is exemplary of advanced robotics simulation. Alongside this, we analyzed the work done by them. Apex, a compact and modular library containing implementations of various continuous reinforcement learning algorithms. Apex is fully compatible with OpenAI Gym, enhancing its utility in our project. Previously, in the deprecated version of the software, efforts had been concentrated on employing algorithms such as Proximal Policy Optimization (PPO) and Twin Delayed DDPG (TD3) for simulation and training tasks.

C. Selection of Algorithm

We opted to experiment with the Soft Actor-Critic (SAC) algorithm, which was covered in our coursework. The decision to utilize the SAC algorithm was influenced by its proven capability to generate an appreciable gait for the Minotaur quadruped robot. Furthermore, since SAC had not been previously implemented by the team at Oregon State University for the Cassie model, we believed that applying this algorithm would provide valuable insights and contribute to innovative exploration in robotic locomotion.

D. Standing vs Walking

Our ambitions for Cassie included mastering both standing and walking behaviors. Initially, our primary objective was to enable Cassie to walk. However, we quickly recognized the complexities involved in balancing a bipedal robot post-initialization, particularly in the absence of an advanced control system. This realization prompted us to modify the XML configuration to initialize Cassie very close to the ground, addressing difficulties in achieving stable ground contact at startup.

Recognizing the foundational importance of standing, we chose to prioritize this capability as our initial focus. We developed a reward function specifically tailored for standing, employing reinforcement learning to secure stable standing postures in the simulator. This strategic approach not only facilitated a deeper understanding of the dynamics involved but also prepared us for the subsequent challenge of walking. By ensuring Cassie could first stand reliably, we laid the groundwork for advancing to more dynamic and complex locomotive behaviors.

III. System Architecture and Workflow

A. Simulation Environment Setup

1) Control Flow in Cassie Simulation Environment: The control flow within the Cassie simulation environment leverages a layered architecture facilitated by Myosuite, extending up to a Gym-compatible interface. Here, we delineate how control transits through the system:

1) **BaseV0 (Myosuite Base Class):** Serves as the foundational layer where the musculoskeletal simulations are initialized and managed using the Mujoco physics engine. It configures muscle conditions, action normalization, and simulations:
   - Muscle conditions such as sarcopenia, fatigue, and reafferentation are modeled to reflect realistic physiological constraints.
   - The `step` function simulates the environment's response to actions, returning the new state along with rewards and other telemetry.
2) **CassieEnvV0 (Specialization of BaseV0):** This layer adapts the BaseV0 for the specific requirements of the Cassie bipedal robot:
   - It adjusts the observation space and reward mechanisms to fit the dynamics and kinematics specific to Cassie.
   - Termination conditions are checked against Cassie's physical states like height and orientation, which dictate the cessation of a simulation episode upon critical failures like falling.
3) **CassieWrapper (Gym Environment Wrapper):** This top layer wraps the CassieEnvV0 to provide a standardized Gym interface for reinforcement learning applications:
   - It defines the action space (10-dimensional) and the observation space (20-dimensional) required by reinforcement learning algorithms.
   - Facilitates the `step` and `reset` functions to interact with the Cassie simulation, thereby integrating seamlessly with RL libraries that utilize the Gym interface.

This multi-layered setup ensures a robust framework where each layer adds specific functionalities, ensuring detailed simulation and effective control of the Cassie robot within a complex musculoskeletal simulation environment.

IV. Custom Environment Creation

A. BaseV0 Class (MyoSuite)

The 'BaseV0' class is a foundational component within the MyoSuite environment for musculoskeletal motor control simulations. It provides essential functionalities for setting up and running simulations, including handling observations, rewards, and termination conditions. Additionally, it supports various configurations such as muscle conditions (e.g., sarcopenia, fatigue), visualization options, and joint targets. Derived classes, such as **CassieEnvV0**, build upon **BaseV0** to create specific environments tailored to different tasks or scenarios, leveraging its robust framework for simulation management.**BaseV0** serves as a flexible and versatile starting point for developing custom environments within the MyoSuite framework.

B. Custom Env-1 Registered with ID: ***CassieWalker-V0***

CassieEnvV0 extends BaseV0 to create a specialized environment for the Cassie bipedal robot by integrating the xml Model. It defines the observation space to include joint positions (`qpos`), velocities (`qvel`), and errors in pose (`pose_err`). Reward signals are computed based on pose accuracy, energy efficiency (`act_reg`), bonuses for good poses, penalties for bad poses, and a custom standing reward. It also has functions to handle robot-specific termination conditions based on orientation and height, and to reset the robot to an initial standing position.

C. SB3 Integration via Custom Env-2 ID: CassieWrapper-V0

CassieWrapper is a gymnasium-compatible environment that allows the CassieEnvV0 to be used with Stable Baselines 3 (SB3), a set of reinforcement learning algorithms. It wraps the Cassie environment, providing the necessary interface for SB3 to interact with. It adapts the Cassie-specific observation and reward computations into a form expected by SB3 agents. It also handles the step, reset, render, and close functions which adhere to SB3 documentation format that allow an SB3 algorithm to train policies on the Cassie simulation.

D. Comparison of Rewards

- **Iteration-1:**
  The Iteration I: utilizes a sophisticated reward function for optimizing a simulated dynamic entity's balance, particularly focusing on its COM height and velocity. The COM height com_height is dynamically adjusted based

on a joint parameter,
text cassie_z , within the simulation, reflecting a position offset to match a predefined ideal. Deviations from this ideal height are penalized through an exponential decay function, which diminishes the height reward reward_height as deviations increase, especially when exceeding a specific tolerance height_tolerance. The decay rate is modulated by orientation_smoothing , ensuring that only substantial deviations from the target height substantially reduce the reward. Concurrently, the system monitors COM velocity com_vel, extracted from the joint cassie_x velocity data. The model aspires to a delicate equilibrium, valuing stability and balance equally, hence incentivizing the simulated system to minimize height variance and regulate horizontal velocity, critical for its steady-state performance.

- **Iteration-2:**
  In Iteration 2 the system's reward algorithm receives significant enhancements to better hone a simulated dynamic entity's ability to maintain balance and efficient locomotion. Key parameters such as ideal height and velocity smoothing are introduced, with a Gaussian function adjusting rewards based on the accuracy of height maintenance—promoting precision in achieving an ideal COM height of 0.75 meters within a 0.1-meter tolerance. The model considers roll and pitch, applying cosine penalties to orientation deviations, thus reinforcing motion stability. Energy efficiency is also evaluated, with a focus on minimizing joint velocities, indicative of the system's strategic movement and energy conservation. The adaptability feature imposes a strict penalty for falling or tilting, with zero reward, underscoring the importance of stability. Weighting across reward components is differentiated, prioritizing height, then velocity, orientation, and energy usage, aligning the model's rewards with the pivotal aspects of locomotion for a steady and optimized performance.

E. Termination Conditions, Observation spaces, action spaces and Reward Function Formulation

Our approach defines specific termination conditions and a composite reward function designed to train the Cassie biped for maintaining a stable, upright posture. The mathematical formulations of these components are integral to the reinforcement learning algorithm's ability to learn effectively.

1) Observation and Action Spaces: In the reinforcement learning environment designed for the Cassie bipedal robot, the observation space and the action space are crucial for training effective policies.

Observation Space: The observation space is 20-dimensional, encompassing various aspects of the robot's state. This includes the robot's joint angles, joint velocities, possibly orientation data from gyroscopes and accelerometers, and other sensors that provide a comprehensive view of the robot's current state. Each dimension in the observation space represents a specific physical or kinematic property of the robot, which the learning algorithm uses to make decisions.

Action Space: The action space consists of a 10-dimensional continuous vector, where each dimension corresponds to a control input for one of the robot's actuators. These dimensions typically represent the desired torques or actuator positions necessary for moving the robot's joints. The continuous nature of this space allows for precise control over the robot's movements, enabling the execution of complex locomotion patterns and maneuvers.

F. Termination Conditions

Termination conditions signal the end of an episode, guiding the agent to avoid undesirable states. The conditions are:

- **Height Constraint:** The biped's center of mass (COM) must remain within a vertical range, expressed as:
$$\text{height\_condition} = \neg(-0.2 < \text{height} < 2.0) \quad (1)$$
where height is the vertical position of the COM relative to the ground.
- **Tilt Constraints:** The biped must not exceed a tilt threshold, formalized as:
$$\text{tilt\_condition} = (|\text{pitch}| > \text{rad}(30)) \vee (|\text{roll}| > \text{rad}(30)) \quad (2)$$
with roll and pitch representing the rotation angles around the biped's lateral and longitudinal axes, respectively.

The episode terminates if either condition is true, signaling that the biped has either fallen or tilted excessively.

1) Reward Function: The reward function is a composite of multiple factors that encourage the biped to maintain a targeted posture:

- **Height Reward:** Encourages the COM to be at an ideal height, using a Gaussian-shaped function centered around the ideal height with a tolerance:
$$\text{reward\_height} = \exp\left(-\frac{(\text{height\_deviation} - \text{height\_tolerance})^2}{\text{orientation\_smoothing}}\right) \quad (3)$$
- **Velocity Reward:** Promotes minimal COM velocity:
$$\text{reward\_vel} = \exp\left(-\sqrt{\text{com\_vel}^2 + \text{velocity\_smoothing}}\right) \quad (4)$$
- **Orientation Penalty:** Penalizes deviation from the upright orientation:
$$\text{orientation\_penalty} = 1 - \cos(\text{roll}) \cdot \cos(\text{pitch}) \quad (5)$$
- **Energy Expenditure:** Incentivizes the minimization of joint velocities:
$$\text{reward\_energy} = \exp\left(-\sum \text{qvel}^2\right) \quad (6)$$

The total reward is a weighted sum of these components, emphasizing the importance of height maintenance, minimal velocity, energy efficiency, and orientation:

$$\begin{aligned}\text{total\_reward} =& w_{\text{height}} \cdot \text{reward\_height} \\ &+ w_{\text{velocity}} \cdot \text{reward\_vel} \\ &+ w_{\text{orientation}} \cdot \text{orientation\_penalty} \\ &+ w_{\text{energy}} \cdot \text{reward\_energy}\end{aligned} \tag{7}$$

Here, w_{height}, w_{velocity}, $w_{\text{orientation}}$, and w_{energy} represent the weights assigned to each reward component, reflecting their relative importance.

The careful balance of these reward components, coupled with the clearly defined termination conditions, equips the learning algorithm with the necessary structure to train the Cassie biped for successful posture control within the simulation environment.

2) Training: The wrapper The SAC model, which operates on the principle of maximizing both the cumulative reward and entropy for more robust learning, is applied to a custom simulation environment, 'CassieWrapper-v0'. Presumably, this environment is associated with a robotic platform, given the naming convention.

The training is set to run for 2 million timesteps, divided into 100 iterations of 20,000 timesteps each. Progress through these iterations is visually tracked via a progress bar provided by tqdm. After each iteration, the model's current state is saved, facilitating incremental training and allowing for the resumption of training without starting over. The script also configures tensorboard logging for detailed performance analytics. Upon completing the training loop, the script gracefully closes the simulation environment, which is a best practice for resource management and preventing memory leaks.

G. Frame Rendering During Training

The training process of the Soft Actor-Critic (SAC) model for the Cassie bipedal robot simulation not only focuses on improving the policy through reinforcement learning but also includes a visual evaluation component through frame rendering. This visualization plays a crucial role in understanding the behavior of the robot and the effectiveness of the learned policies over time.

1) Purpose of Rendering: Rendering frames during the training provides immediate visual feedback on the robot's performance under various policies. This helps in identifying any unexpected behaviors or anomalies in the simulation that may not be evident through numerical metrics alone. Furthermore, it allows researchers and developers to visually confirm the robot's adherence to desired locomotion patterns.

2) Implementation Details: The rendering process is integrated into the main training loop, where each frame of the robot's movement is captured after executing an action decided by the policy. The Python library *skvideo.io* is used to compile these frames into a video file, which can be reviewed after the training session. This setup is particularly useful for presentations and detailed analyses, where visual evidence of improvement or regression in robot behavior is necessary.

a) Conditional Frame Capture:: Frames are rendered conditionally based on the performance of each episode. If the cumulative reward of an episode surpasses the previously recorded best, the frames from this episode are stored as the new best performance sequence. This method ensures that only the most successful episodes, in terms of task completion and stability, are recorded and reviewed.

b) Technical Setup:: The rendering is performed using offscreen rendering techniques provided by the simulation environment's underlying graphics engine, which is configured to capture high-quality images of the robot's movements without displaying them on a screen. This approach is efficient and allows the training process to run faster by not rendering every frame onscreen but still capturing them for analysis.

3) Post-Processing and Storage: Once the best performing episodes are identified and their frames captured, *skvideo.io* compiles these frames into a video file, storing it with a naming convention that reflects the training iteration and the timestep, facilitating easy organization and retrieval. These video files serve as a chronological archive of the robot's performance improvements and are invaluable for longitudinal studies and reporting.

4) Registering CassieWalk-V0 and CassieWrapper-V0: The registration process for CassieWalk-v0 is meticulously conducted through MyoSuite, while also ensuring compatibility with the Gym interface, a standard within the reinforcement learning community. This dual-compatibility approach allows users to specify a diverse array of environment settings such as the model path, normalization actions, and reset conditions, directly within the MyoSuite framework and accessible via the Gym API.

For CassieWrapper-v0, the registration is done using Gymnasium. It follows the conventional process for registering a new environment but points to a wrapper class that adapts the MyoSuite environment for use with Stable Baselines 3. This wrapper ensures that the sophisticated simulation capabilities of MyoSuite can be utilized within the more generic SB3 algorithmic framework.

V. Reinforcement learning Implementation

This section aims to cover the choice of algorithm and presents the findings from our experiments using advanced reinforcement learning algorithms, specifically Soft Actor-Critic (SAC) and Twin Delayed Deep Deterministic policy gradient (TD3), on the Cassie bipedal robot simulation. We explore their performance in continuous control tasks, leveraging the capabilities of the Mujoco physics engine.

A. SAC algorithm based on Stable Baselines 3

The Soft Actor-Critic (SAC) algorithm is a state-of-the-art reinforcement learning method designed for tasks requiring continuous control. Implemented using Stable Baselines 3, a powerful learning framework, SAC optimizes the trade-off between expected return and policy entropy. The entropy term acts as a regularizer, encouraging exploration by making the policy stochastic.

Mathematically, SAC seeks to optimize the objective function:

$$J(\pi) = \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[\sum_t \gamma^t (r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t)))\right] \quad (8)$$

where $r(s_t, a_t)$ is the reward function, α is the temperature parameter that controls the importance of the entropy term $\mathcal{H}$ in the objective function, and γ is the discount factor. This formulation includes an entropy bonus $\mathcal{H}(\pi(\cdot|s_t))$ to ensure sufficient exploration. SAC's dual architecture, consisting of separate actor and critic networks, further stabilizes learning by allowing the actor to explore a variety of actions while the critic accurately evaluates those actions.

The critic's role is to evaluate the action proposed by the actor by estimating the soft Q-function, which incorporates the entropy term:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[V(s_{t+1})\right] \quad (9)$$

where $V(s_{t+1})$ is the soft value function estimated as:

$$V(s) = \mathbb{E}_{a\sim\pi}\left[Q(s,a) - \alpha\log\pi(a|s)\right] \quad (10)$$

In our implementation, SAC controls the motor speeds of each of the four joints in the Cassie model, with action values ranging from -1 to 1, using Stable Baselines 3 which provides an efficient and robust framework for policy optimization in complex environments like humanoid locomotion.

B. TD3 - Twin Delayed Deep Deterministic policy gradient

TD3, or Twin Delayed Deep Deterministic policy gradient, is an advanced reinforcement learning algorithm that enhances the Deep Deterministic Policy Gradient (DDPG) by addressing its tendency to overestimate action values. This enhancement is achieved through three critical improvements:

- **Double Q-learning:** TD3 employs two separate critic networks (and their corresponding target networks), which helps mitigate positive bias in the policy update by taking the minimum value of the two Q-functions for each action. Mathematically, if we denote the two Q-functions by Q_{θ_1} and Q_{θ_2}, the target value for the policy update is given by:

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \mu_{\phi'}(s' + \epsilon)) \quad (11)$$

  where θ_i' are the parameters of the target critics, $\mu_{\phi'}$ is the target policy, s' is the next state, r is the reward, γ is the discount factor, and ϵ is noise added to the target policy action to smooth out the Q estimates.
- **Delayed policy updates:** To further stabilize training, policy updates are performed less frequently than critic updates. This delay prevents the policy from chasing a moving target due to rapid Q-function updates. Typically, the policy and target networks are updated every d critic updates.
- **Target policy smoothing:** TD3 adds noise to the target policy to prevent deterministic exploitation of Q-function errors by smoothing out the actions used in the Q-update. This noise, typically Gaussian, makes the policy output less sensitive to the input state, promoting generalization across similar states.

These modifications collectively improve the robustness and efficiency of policy learning, particularly in environments with high-dimensional action spaces, such as controlling the Cassie bipedal robot. The addition of noise and the use of dual Q-functions ensure that the policy learning is based on reliable Q-value estimates, leading to more stable and consistent performance across various states. The deliberate delay in policy updates helps in avoiding the pitfalls of rapid, non-convergent policy learning cycles, ensuring more reliable and effective learning outcomes.

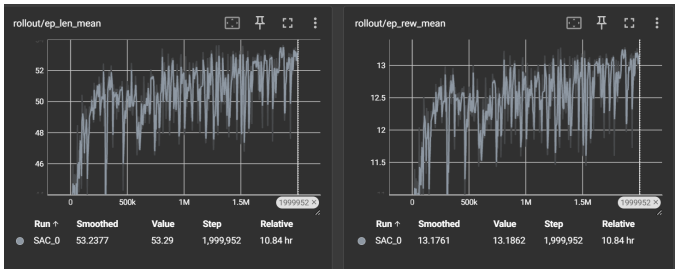

Fig. 2. Graph for basic reward function

VI. Analysis of SAC Policy Training on Cassie Biped

During the training of a Soft Actor-Critic (SAC) policy on the task of standing using the Cassie biped robot, TensorBoard logs were recorded to visualize the performance of 2 million training steps. We present an analysis of two key metrics: the mean length of each episode and the mean reward per episode.

A. Episode Length Mean (rollout/ep_len_mean)

- The average episode length remained between approximately 46 to 53 timesteps.
- Despite fluctuations, the smoothed line suggests a stable episode length throughout the training.
- However, the lack of a positive trend in episode duration may also imply that there is room for improvement in extending the standing time.

B. Actor and Critic Loss Analysis

- The actor loss after the reward update indicates a more stable and converged policy, as evidenced by the reduced variance and a relatively flat trend post-update. The lower loss values suggest that the policy is making more effective decisions, likely contributing to the improved posture maintenance observed.
- Critic loss, which reflects the temporal difference error in value estimation, shows a significant reduction in variance after the update. This reduction suggests that the updated reward function provides a clearer signal for value estimation, aiding the critic in making more accurate predictions about future rewards, which translates to better policy guidance.

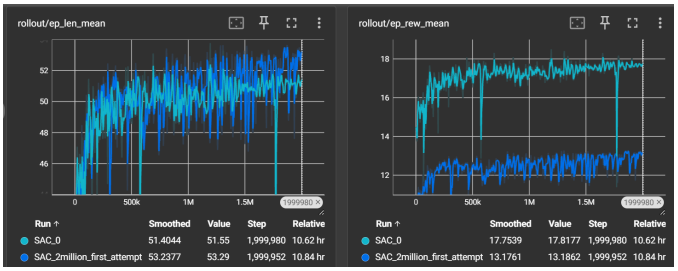

Fig. 3. Reward after updating reward function

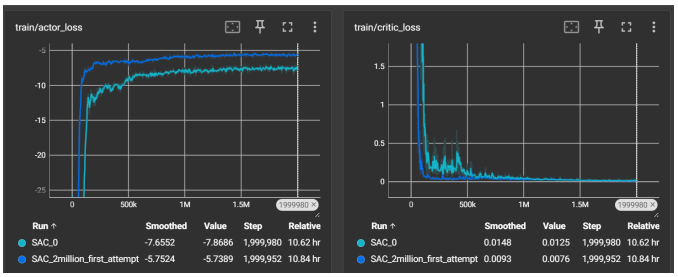

Fig. 4. Actor-critic losses

C. Episode Reward Mean (rollout/ep_rew_mean)

- The mean reward per episode fluctuated around 11.5 to 13, with noticeable variance.
- The flat trend in rewards suggests that the policy's performance did not significantly improve in terms of reward maximization.
- Given the task at hand, the basic reward function might need re-evaluation to better guide the policy toward prolonged standing.

VII. Comparative Analysis of SAC Policy Performance on Cassie Biped

After revising the reward function for the SAC policy training on the Cassie biped robot, we observed notable changes in the robot's performance. Although the biped did not achieve full standing, there were measurable improvements in posture as indicated by the following analysis of two primary performance metrics: mean episode length and mean reward per episode.

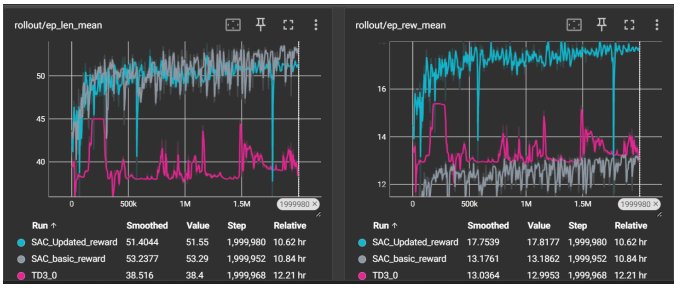

Fig. 5. SAC and TD3 results

A. Episode Length Mean (rollout/ep_len_mean)

- Post-update, the average episode length shows an upward shift in range, predominantly lying between 50 to 53 steps, compared to the initial 46 to 53 steps.
- The reduction in variance and the slight upward trend of the smoothed line suggest that the updated reward function has led to a more stable and sustained improvement in the biped's ability to maintain its posture.
- The convergence of actor loss further corroborates the increased episode length, indicating that the actor is making effective decisions that contribute to the biped's maintained posture.
- The decrease in critic loss supports the notion that the value estimates are becoming more accurate, allowing the agent to make more informed decisions that likely result in longer sustained postures.

B. Episode Reward Mean (rollout/ep_rew_mean)

- A marked increase in mean reward is observed, with values now oscillating around 17 to 18, compared to the prior range of 11.5 to 13.
- The upward trend and lower variance in the rewards indicate that the Cassie biped is achieving higher rewards more consistently. This is a sign of improved posture control in response to the enhanced reward signal.
- The stabilized actor and critic losses are reflective of a learning process that has become more effective at optimizing the rewards. The actor, influenced by better value estimates from the critic, is likely to be selecting actions that result in a more balanced and controlled posture.
- Improved loss metrics suggest not only better policy performance but also an enhanced learning stability, which is instrumental in achieving a consistently better posture for the Cassie biped.

C. Comparative Performance Analysis

We extended our analysis to include the Twin Delayed Deep Deterministic policy gradient (TD3) algorithm alongside the two SAC implementations. The insights drawn from the episode length mean and reward mean are as follows:

1) Episode Length Mean Insights:

- **SAC with Updated Reward**: Exhibits a stable range for episode length, primarily between 50 to 53 steps, indicative of consistent postural maintenance.
- **SAC with Basic Reward**: Initially, the episode length was slightly higher but showed greater variability, suggesting a less stable policy.
- **TD3**: The episode length mean was significantly lower, around 38 to 40 steps, pointing to challenges in maintaining posture as effectively as SAC.

2) Episode Reward Mean Insights:

- **SAC with Updated Reward**: Achieves higher and more consistent rewards, averaging around 17 to 18, highlighting the effectiveness of the revised reward function.

- **SAC with Basic Reward**: Demonstrates lower and more fluctuating rewards, suggesting that the basic reward may not adequately guide the policy for this task.
- **TD3**: Rewards were comparable to the SAC with a basic reward but with increased erratic behavior, suggesting potential instability in the learning process.

3) Overall Observations: The TD3 algorithm, when compared to the SAC implementations, exhibited difficulty in achieving similar performance levels. The SAC with an updated reward function outperformed the other methods in terms of both stability and the ability to maintain posture. The variability of the TD3 results could imply that this algorithm might require additional parameter tuning or a different approach to reward structuring to perform as well as SAC for this specific task.

VIII. Discussion and Future Improvement Strategies

While the Soft Actor-Critic (SAC) algorithm with the updated reward function has shown promise in enhancing the Cassie biped's posture control, there remain opportunities for further advancement. The following strategies are proposed to overcome current challenges and improve performance in future iterations of the project.

1) Imitation Learning: Imitation learning, a strategy where the policy is trained to mimic a pre-defined controller or expert behavior, presents a promising avenue:

- By leveraging expert demonstrations, the policy could quickly acquire complex behaviors that maintain posture and balance more effectively.
- Transferring these learned behaviors into the simulation environment may accelerate the training process and lead to more robust policies.

2) Hyperparameter Optimization: Adjusting the learning algorithm's hyperparameters can significantly influence the training outcome:

- Systematic tuning of hyperparameters, possibly through automated search techniques like grid search or Bayesian optimization, may uncover more optimal settings that enhance learning stability and performance.
- Attention should be given to parameters that control exploration-exploitation balance, learning rates, and the discount factor, as these directly affect policy development and convergence.

3) Reward Function Refinement: The design of the reward function is crucial in shaping the desired behaviors. Further refinement can provide clearer and more immediate feedback to the agent:

- Introducing reward shaping techniques, such as potential-based reward shaping, could make the reward signal more informative and guide the policy towards more precise postural adjustments.
- Considering sparse and dense reward components can encourage both short-term success in tasks and long-term strategic planning.

4) Comprehensive Strategy: A comprehensive approach that combines imitation learning, hyperparameter optimization, and reward function refinement stands to offer synergistic benefits:

- An initial phase of imitation learning can provide a solid foundation, which can then be fine-tuned through hyperparameter optimization.
- Throughout this process, a continuously refined reward function will ensure that the policy is always steered towards the most relevant behaviors for the task at hand.

In conclusion, while significant progress has been made, the full potential of the Cassie biped robot's control policy can be further unlocked by these strategic enhancements. Future work will incorporate these elements, aiming to establish a more capable and resilient control policy that brings the Cassie biped closer to achieving dynamic and autonomous standing and walking abilities.