**TD3\_train.py — Training the TD3 Agent in a PyBullet Environment**

**Overview**

This script implements the training pipeline for a Twin Delayed Deep Deterministic Policy Gradient (TD3) agent controlling a robotic arm in a simulated environment (MassageEnv) using PyBullet. It includes the neural network definitions for the actor and critic, replay buffer, training loop, and environment interaction.

**Key Components**

**1. Actor Network (Policy)**

* A neural network that maps states to continuous actions.
* Architecture: 3 fully connected layers with ReLU activations and a final tanh scaled by max\_action.
* Input: state vector (includes environment features).
* Output: action vector (3D continuous control).

**2. Critic Network (Q-function)**

* Two Q-networks (Q1 and Q2) to mitigate overestimation bias.
* Each network takes state and action concatenated as input and outputs a scalar Q-value.
* Architecture: 3 fully connected layers per Q-network with ReLU activations.

**3. Replay Buffer**

* Stores experience tuples (state, action, reward, next\_state, done).
* Supports random sampling for mini-batch training.

**4. TD3 Agent Class**

* Contains actor, critic, target networks, and optimizers.
* Implements the TD3 training algorithm with delayed policy updates, clipped noise for target policy smoothing, and soft target updates.

**5. Environment Interaction Helpers**

* local\_reset(env): Resets the human and robot to fixed initial states.
* local\_step(env, action): Executes an action by inverse kinematics and steps the simulation.
* Functions to check collisions and contacts for debugging.

**6. Training Loop (train\_td3())**

* Initializes environment, agent, replay buffer.
* Runs episodes where the agent interacts with the environment, collects experience, and trains the networks.
* Uses action smoothing and oscillation in the x-axis for exploration.
* Logs rewards and losses, saves models periodically.
* Visualizes training progress with matplotlib.

