ADR Simulation

Now I provide you with a simulation code that simulates adaptive domain randomization (ADR) of humanoid robots for experiments, which serves as a control group in this research. The following code is designed for Python to run the ADR simulator. As part of the experimental process, you need to strictly execute the code and conduct objective data analysis. Since the code is built internally and cannot read from user-supplied data, I will need to work with you on this. Please strictly simulate the environment and run the code first, and then I will submit the mechanical data converted into computer language for you to execute the code. After execution, you need to quantify the experimental results based on the following five objective indicators: Sample Efficiency, Adaptation Speed, Robustness, Policy Stability, and Domain Gap. Please configure the corresponding functions, processes and environments according to the code, and objectively execute the program in this dialog box and calculate the data analysis results. Are you ready? If not, no problem, I will upload the simulation code of ADR in the next conversation.

I am ready to assist with your simulation. Please upload the simulation code for the adaptive domain randomization (ADR) of humanoid robots, and I will carefully review, configure the required environment, and execute the code to simulate the process. Afterward, I will perform an objective analysis of the result sed on the five indicators: Sample Efficiency, Adaptation Speed, Robustness, Policy Stability, and Domain Gap. Let's proceed with your

Control Group - Adaptive Domain ...
Python

Now please execute the ADR simulation as the control group code.

The ADR simulation for the control group has been successfully executed. Here are the results:

- **Total Episodes:** 50
- **Success Rate:** 90%

You can now provide the mechanical data converted into computer language for further execution and analysis. Once the data is submitted, I will proceed with the calculations for the five objective indicators: Sample Efficiency, Adaptation Speed, Robustness, Policy Stability, and Domain Gap. Let me know when you're ready to proceed!

Mechanical Data Synthesis (Friction or Slope)

Mechanical Data Synthesis (Frictio...
文件

Okay. Next I will submit 500 mechanical synthetic data recorded in the document that have been converted into computer language. Please objectively calculate the Sample Efficiency, Adaptation Speed, Robustness, Policy Stability, and Domain Gap according to the calculation formula.

The mechanical data file has been successfully uploaded. I will now calculate the **Sample Efficiency**, **Adaptation Speed**, **Robustness**, **Policy Stability**, and **Domain Gap** based on the provided data using the agreed formulas. Please hold on while I process the data and generate the results.

Finalized ADR Mechanical Data Metrics

	Metric	Value
1	Sample Efficiency	0.568243
2	Adaptation Speed	0.03228
3	Robustness	0.598585
4	Policy Stability	0.485441
5	Domain Gap	0.047353

Environmental Data Synthesis (Lighting or Obstacles)

Environmental Data Synthesis (Lig...
文件

Okay. Next I will submit 500 environmental synthetic data recorded in the document that have been converted into computer language. Please objectively calculate the Sample Efficiency, Adaptation Speed, Robustness, Policy Stability, and Domain Gap according to the calculation formula.

The environmental data file has been successfully uploaded. I will now calculate the **Sample Efficiency**, **Adaptation Speed**, **Robustness**, **Policy Stability**, and **Domain Gap** based on the provided data using the agreed formulas. Please hold on while I process the data and generate the results.

Finalized ADR Environmental Data Metrics

	Metric	Value	
1	Sample Efficiency	0.478856	
2	Adaptation Speed	0.018318	
3	Robustness	0.543499	
4	Policy Stability	0.531288	
5	Domain Gap	0.04367	

TDR Integration

Now I provide you with a simulation code that simulates targeted domain refinement (TDR) of humanoid robots for experiments, which serves as a control group in this research. The following code is designed for Python to run the TDR simulator. As part of the experimental process, you need to strictly execute the code and conduct objective data analysis. Since the code is built internally and cannot read from user-supplied data, I will need to work with you on this. Please strictly simulate the environment and run the code first, and then I will submit the mechanical data converted into computer language for you to execute the code. After execution, you need to quantify the experimental results based on the following five objective indicators: Sample Efficiency, Adaptation Speed, Robustness, Policy Stability, and Domain Gap. Please configure the corresponding functions, processes and environments according to the code, and objectively execute the program in this dialog box and calculate the data analysis results. Are you ready? If not, no problem, I will upload the simulation code of TDR in the next conversation.

I am ready to assist with the simulation and analysis process. Please upload the simulation code for the TDR, and I will configure the necessary functions and environments to execute it. Once the mechanical data is provided, I will conduct the objective data analysis based on the specified indicators: Sample Efficiency, Adaptation Speed, Robustness, Policy Stability, and Domain Gap. Feel free to proceed with uploading the code, and I'll begin immediately.

Targeted Domain Refinement (Int...
Python

Now please execute the TDR simulation as the control group code.

The Targeted Domain Refinement (TDR) simulation executed successfully. Below are the results:

- **Total Episodes:** 20
- **TDR Success Rate:** 80% (0.80)
- **Validation Results:** Success rate of 80% across all episodes.
- **Safety Report:** Safe to deploy, with a success rate of 80%.
- **Deployment Record:** Successful through all deployment phases (lab test, restricted field, and full deployment).
- **Meta Parameters:**
  - Global Adaptation Rate: 0.1
  - Failure Ratio: 20% (0.2)
- **Aligned Knowledge:**
  - Bright environment: Transfer factor 0.0125, Average friction 0.743
  - Dim environment: Transfer factor 0.0100, Average friction 0.813
  - Normal environment: Transfer factor 0.0100, Average friction 0.752

Please provide the mechanical data converted into computer language for further simulations and data analysis based on the specified indicators.

Mechanical Data Synthesis (Friction or Slope)

Mechanical Data Synthesis (Frictio...
文件

Okay. Next I will submit 500 mechanical synthetic data recorded in the document that have been converted into computer language. Please objectively calculate the Sample Efficiency, Adaptation Speed, Robustness, Policy Stability, and Domain Gap according to the calculation formula.

The mechanical data file has been successfully uploaded. I will now calculate the **Sample Efficiency**, **Adaptation Speed**, **Robustness**, **Policy Stability**, and **Domain Gap** based on the provided data using the agreed formulas. Please hold on while I process the data and generate the results.

Mechanical Data Indicators

	Metric	Value
1	Sample Efficiency	0.744992
2	Adaptation Speed	0.04142
3	Robustness	0.945662
4	Policy Stability	0.717287
5	Domain Gap	0.074357

Environmental Data Synthesis (Lighting or Obstacles)

Environmental Data Synthesis (Lig...
文件

Okay. Next I will submit 500 environmental synthetic data recorded in the document that have been converted into computer language. Please objectively calculate the Sample Efficiency, Adaptation Speed, Robustness, Policy Stability, and Domain Gap according to the calculation formula.

The environmental data file has been successfully uploaded. I will now calculate the **Sample Efficiency**, **Adaptation Speed**, **Robustness**, **Policy Stability**, and **Domain Gap** based on the provided data using the agreed formulas. Please hold on while I process the data and generate the results.

Finalized Environmental Data Metrics

	Metric	Value
1	Sample Efficiency	0.606894
2	Adaptation Speed	0.02558
3	Robustness	0.820817
4	Policy Stability	0.673399
5	Domain Gap	0.063276

Data Analysis Formula

1. Redesigned Metrics and Formulas

(a) Sample Efficiency

- **Goal**: Quantify how effectively the policy learns under domain randomization, minimizing the required training episodes.
- **Proposed Formula**:

$$\text{Sample Efficiency} = \frac{\text{Cumulative Reward}}{\text{Total Episodes}}$$

- **Justification**:
  - This formula directly links the cumulative performance of the policy (reward) to the resources (episodes) consumed, a standard measure in reinforcement learning literature.
  - It aligns with practices in robotics, where minimizing training costs is critical.
- **Benefits**: Empirically grounded and easy to compute from training logs.

(b) Adaptation Speed

- **Goal**: Measure how quickly the policy adapts to changes in domain conditions.
- **Proposed Formula**:

$$\text{Adaptation Speed} = \frac{\Delta\text{Reward}}{\Delta\text{Episodes}}$$

or

$$\text{Adaptation Speed} = \frac{\text{Time to Reach Threshold Performance}}{\text{Episodes in New Domain}}$$

- **Justification**:
  - The first version tracks how quickly rewards improve after a domain change, reflecting learning speed.
  - The second version measures time-to-convergence to a predefined performance threshold, widely used in robotics and machine learning.
- **Benefits**: Reflects real adaptation dynamics, validated in ADR and transfer learning research.

(c) Robustness

- **Goal**: Evaluate how consistently the policy performs across diverse domain conditions.
- **Proposed Formula**:

$$\text{Robustness} = 1 - \frac{\text{Variance in Reward}}{\text{Mean Reward}}$$

- **Justification**:
  - Low variance in rewards across randomized domains indicates high robustness, aligning with standard definitions.
  - This approach directly measures performance consistency, often used in industrial robot testing.
- **Benefits**: Robustness becomes a measurable, dimensionless value between 0 and 1, with higher values reflecting better performance.

(d) Policy Stability

- **Goal**: Assess the policy's consistency across episodes within the same domain.
- **Proposed Formula**:

$$\text{Policy Stability} = 1 - \frac{\text{Standard Deviation of Actions}}{\text{Mean Actions}}$$

- **Justification**:
  - Stability is often quantified as the consistency of actions or trajectories taken by the robot across trials.
  - This formula captures deviations in policy outputs, widely referenced in trajectory planning literature.
- **Benefits**: Provides a direct measure of stability, tied to robot actions rather than environmental parameters.

(e) Domain Gap

- **Goal**: Quantify the transfer difficulty from simulation to real-world environments.
- **Proposed Formula**:

$$\text{Domain Gap} = \frac{\text{Real-World Performance} - \text{Simulated Performance}}{\text{Real-World Performance}}$$

- **Justification**:
  - Directly compares performance between simulation and reality, reflecting transfer success.
  - Commonly used in ADR studies (e.g., OpenAI, DeepMind) to evaluate sim-to-real techniques.
- **Benefits**: Grounded in ADR literature and aligns with reviewers' expectations for scientifically rigorous metrics.