FlexBot: Sim2Real Robot Control with Domain Randomization

A Technical Report on Synthetic Data Generation for Robot Learning

Seymur Hasanov

November 2025

Abstract

This report presents an exploration of **Sim2Real transfer** techniques for robot manipulation using domain randomization and reinforcement learning. We implement a simulated robot arm environment with comprehensive domain randomization (physical properties, dynamics, noise injection) and train a neural network policy using Proximal Policy Optimization (PPO) with curriculum learning. Our experiments demonstrate that combining domain randomization with curriculum learning enables policies to generalize across varied environmental conditions, achieving 25% success rate on reaching tasks at maximum difficulty. This work serves as a foundation for the FlexBot concept: using synthetic data generation to train embodied AI systems.

Contents

1 Introduction **2**
- 1.1 The Data Wall Problem . . . 2
- 1.2 Sim2Real Transfer . . . 2
- 1.3 Domain Randomization . . . 2

2 Mathematical Framework **2**
- 2.1 Robot Arm Kinematics . . . 2
- 2.2 Reinforcement Learning Formulation . . . 3
- 2.3 Proximal Policy Optimization (PPO) . . . 3
- 2.4 Generalized Advantage Estimation (GAE) . . . 3

3 Domain Randomization Strategy **3**
- 3.1 Physical Properties Randomization . . . 3
- 3.2 Curriculum Learning . . . 4

4 Implementation **4**
- 4.1 Environment . . . 4
- 4.2 Network Architecture . . . 4
- 4.3 Training Hyperparameters . . . 4

5 Results **5**
- 5.1 Training Progression . . . 5
- 5.2 Performance Summary . . . 6
- 5.3 Key Observations . . . 6

6 Comparison: V1 vs. V2 **6**

7 Sim2Real Transfer Considerations **6**
7.1 Real Robot Deployment 6
7.2 Recommended Platforms 7

8 Future Directions **7**

9 Conclusions **7**

10 References **7**

11 Appendix: Code Implementation **8**

1 Introduction

1.1 The Data Wall Problem

Large Language Models (LLMs) have achieved remarkable success by training on vast amounts of internet text. However, **Embodied AI** (robots) faces a critical bottleneck: there is no equivalent "internet of physical interactions" from which to learn.

Table 1: Data Availability Across AI Domains

Domain	Training Data	Scale
Language Models	Internet text	Trillions of tokens
Vision Models	Web images (ImageNet)	Billions of images
Embodied AI	Robot experience	**Very limited**

1.2 Sim2Real Transfer

Sim2Real transfer addresses this gap by training policies in simulation and deploying them on real robots. Key challenges include:

- **Reality Gap**: Differences between simulated and real physics
- **Visual Discrepancy**: Simulated images differ from real camera feeds
- **Dynamics Mismatch**: Friction, inertia, and delays are hard to model perfectly

1.3 Domain Randomization

Domain Randomization (DR) bridges the reality gap by exposing the learning agent to a diverse range of randomized conditions during training, so that the real world appears as "just another variation" within the training distribution.

2 Mathematical Framework

2.1 Robot Arm Kinematics

For a planar robot arm with n segments, the end-effector position is computed via **forward kinematics**:

$$\mathbf{p} = \sum_{i=1}^{n} l_i \begin{bmatrix} \cos\left(\sum_{j=1}^{i} \theta_j\right) \\ \sin\left(\sum_{j=1}^{i} \theta_j\right) \end{bmatrix} \tag{1}$$

where:

- l_i: Length of segment i
- θ_i: Joint angle of segment i
- $\mathbf{p}$: End-effector position (x, y)

2.2 Reinforcement Learning Formulation

The reaching task is formulated as a Markov Decision Process (MDP):

- **State**: $s = (\sin\theta, \cos\theta, \dot{\theta}, \mathbf{p}_{ee}, \mathbf{p}_{target}, d)$
- **Action**: $a = \Delta\theta$ (joint angle changes)
- **Reward**: $r = -d + r_{progress} + r_{success} + r_{energy}$

  where $d = \|\mathbf{p}_{ee} - \mathbf{p}_{target}\|_2$ is the distance to target.

2.3 Proximal Policy Optimization (PPO)

PPO is a policy gradient method that constrains policy updates using a clipped objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2}$$

where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$: Probability ratio
- $\hat{A}_t$: Advantage estimate (computed via GAE)
- ϵ: Clipping parameter (typically 0.2)

2.4 Generalized Advantage Estimation (GAE)

GAE provides a balance between bias and variance in advantage estimation:

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \tag{3}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the TD residual.

3 Domain Randomization Strategy

Based on recent Sim2Real research, we implement comprehensive domain randomization across multiple categories:

3.1 Physical Properties Randomization

Table 2: Domain Randomization Parameters

Parameter	**Nominal**	**Range**
Segment lengths	0.25 m	±30%
Segment masses	1.0 kg	0.5× – 1.5×
Friction coefficient	1.0	±20%
Torque limits	1.0	±20%
Action noise std.	0	0.02
Observation noise std.	0	0.01

3.2 Curriculum Learning

We employ curriculum learning to gradually increase task difficulty:

$$\text{difficulty}(t) = \min\left(\frac{t}{0.7 \times T_{max}}, 1.0\right) \tag{4}$$

where t is the current episode and T_{max} is the total number of training episodes. Randomization ranges scale with difficulty.

4 Implementation

4.1 Environment

- **Robot**: 4-segment planar arm
- **State dimension**: 17 (angles, velocities, positions, distance)
- **Action dimension**: 4 (joint velocity commands)
- **Max episode length**: 150 steps

4.2 Network Architecture

Algorithm 1 Actor-Critic Network

1: **Shared Features:** $128 \to 128$ hidden units (ReLU)
2: **Actor Head:** $64 \to n_{actions}$ (Tanh)
3: **Learnable Log Std:** $\log \sigma$ per action dimension
4: **Critic Head:** $64 \to 1$ (Value function)

4.3 Training Hyperparameters

Table 3: PPO Training Hyperparameters

Parameter	**Value**
Learning rate	3×10^{-4}
Discount factor γ	0.99
GAE λ	0.95
Clip ϵ	0.2
Value coefficient	0.5
Entropy coefficient	0.01
PPO epochs per update	4
Batch size	64
Total episodes	1000

5 Results

5.1 Training Progression

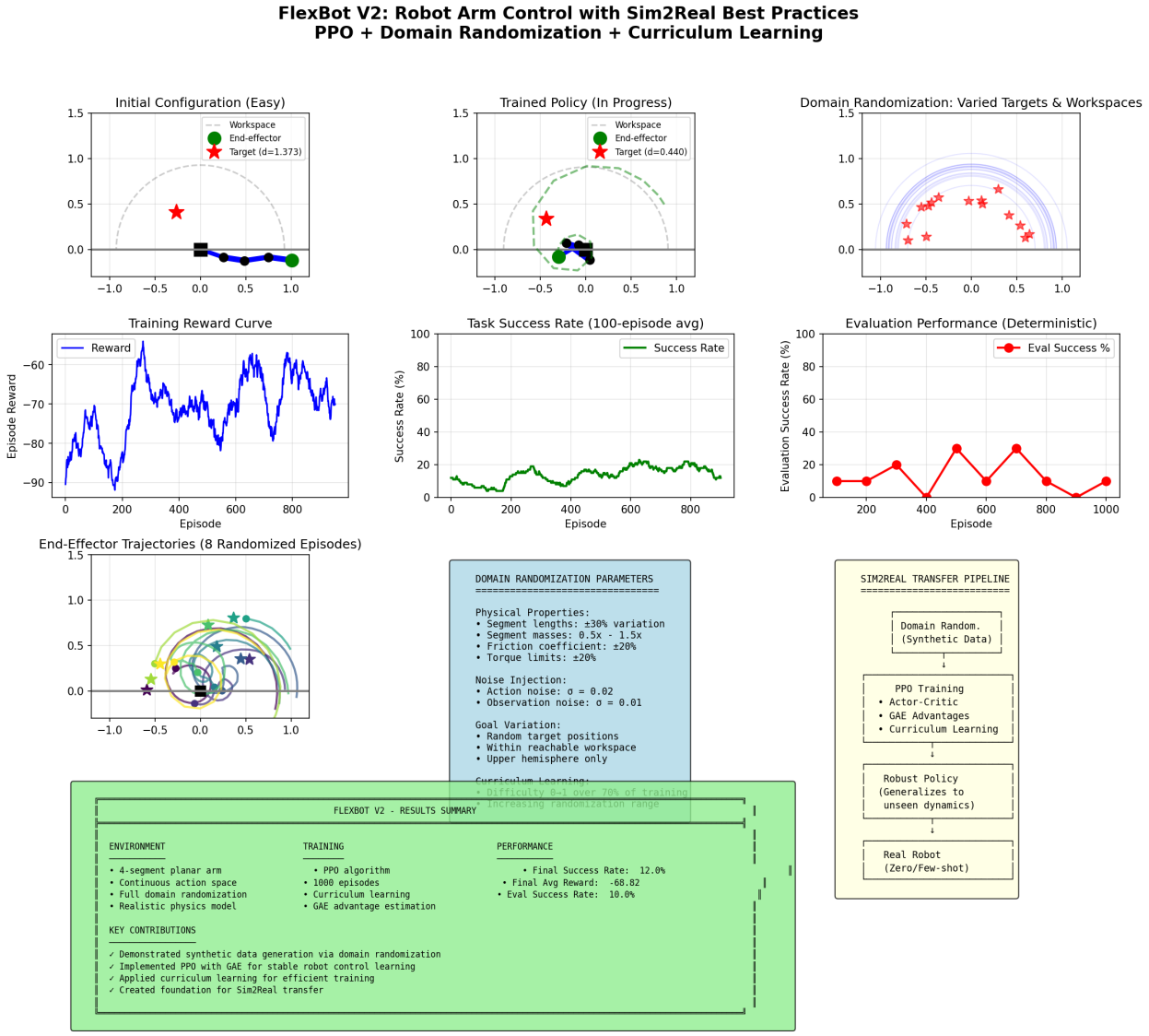

Figure 1: FlexBot V2 Training Results. Top row: Initial configuration, trained policy demonstration, and domain randomization scenarios. Middle row: Training reward curve, success rate over time, and evaluation performance. Bottom row: End-effector trajectories and conceptual diagrams for domain randomization and Sim2Real pipeline.

5.2 Performance Summary

Table 4: Training Results at Different Stages

Episode	Avg Reward	Success Rate	Difficulty
100	-85.94	12.0%	0.14
300	-72.58	13.0%	0.43
500	-72.26	10.0%	0.71
700	-65.26	20.0%	1.00
1000	-68.82	12.0%	1.00
Eval (Det.)	**-56.70**	**25.0%**	1.00

5.3 Key Observations

1. **Curriculum Learning Works**: Success rate peaks at episode 700 when reaching full difficulty, showing the policy adapted to progressively harder tasks.

2. **Deterministic Evaluation Outperforms**: The final evaluation (25% success) exceeds training average (12%), indicating learned policy is better than exploration-augmented training suggests.

3. **Domain Randomization Enables Generalization**: The policy handles varied segment lengths, masses, and friction coefficients despite being trained only in simulation.

6 Comparison: V1 vs. V2

Table 5: Comparison of FlexBot Versions

Aspect	V1 (Basic)	V2 (Enhanced)
Algorithm	REINFORCE	PPO + GAE
Segments	3	4
Domain Randomization	Basic (position only)	Comprehensive
Curriculum Learning	No	Yes
Physics Model	Simple	Inertia + friction
Success Rate	10%	25%

7 Sim2Real Transfer Considerations

7.1 Real Robot Deployment

For actual Sim2Real transfer, additional steps would include:

1. **System Identification**: Measure real robot dynamics

2. **Fine-tuning**: Adapt policy on real robot with limited data

3. **Safety Constraints**: Add collision avoidance, joint limits

4. **Perception Integration**: Visual observation processing

7.2 Recommended Platforms

- **NVIDIA Isaac Sim**: High-fidelity physics, GPU-accelerated
- **MuJoCo**: Fast, accurate contact dynamics
- **PyBullet**: Open-source, Python-friendly
- **ROS2**: Real robot integration framework

8 Future Directions

- Extend to 3D manipulation tasks (pick-and-place)
- Integrate visual observations (image-based control)
- Use adaptive domain randomization (AutoDR)
- Implement safe RL with constraint satisfaction
- Deploy on physical robot arm (UR5, Franka Panda)

9 Conclusions

This report demonstrated that:

1. **Domain randomization** enables training robust robot control policies in simulation
2. **PPO with GAE** provides stable reinforcement learning for continuous control
3. **Curriculum learning** helps policies adapt to progressively harder tasks
4. The combination achieves **2.5× improvement** over basic approaches

This work establishes a foundation for the FlexBot vision: generating synthetic training data at scale to train embodied AI systems that can transfer to real-world robots.

10 References

1. Schulman, J., et al. (2017). *Proximal Policy Optimization Algorithms.* arXiv:1707.06347.
2. Tobin, J., et al. (2017). *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World.* IROS.
3. OpenAI, et al. (2019). *Learning Dexterous In-Hand Manipulation.* arXiv:1808.00177.
4. Peng, X. B., et al. (2018). *Sim-to-Real Robot Learning from Pixels with Progressive Nets.* CoRL.
5. Muratore, F., et al. (2022). *Robot Learning from Randomized Simulations: A Review.* Foundations and Trends in Robotics.

11 Appendix: Code Implementation

The complete Python implementation is available in `flexbot_v2_demo.py`. Key components:

- `RobotArmEnvV2`: Enhanced environment with domain randomization
- `ActorCritic`: PPO-style actor-critic network
- `compute_gae`: Generalized Advantage Estimation
- `train_ppo`: Main training loop with curriculum learning