www.ijarp.com ISSN 2456-9992 Page: 01-08
International Journal Advanced Research Publications
Research Article
Volume: 02
Issue: 05

REAL-TIME LIVE TRAINING IMITATION LEARNING SYSTEM FOR AUTONOMOUS AGENTS IN 3D ENVIRONMENTS

***Siddharth T., Thiruvalluvan N. A., Sridhanwanth S.**

Dept of AIML, Sri Shakthi Institute of Engineering and Technology.

Article Received: 24 March 2026, Article Revised: 14 April 2026, Published on: 04 May 2026

***Corresponding Author: Siddharth T.**
Dept of AIML, Sri Shakthi Institute of Engineering and Technology.
DOI: https://doi-doi.org/101555/ijarp.1932

ABSTRACT

The development of adaptive and responsive artificial intelligence in 3D interactive environments often requires extensive training time and massive datasets. This study introduces a live training imitation learning system designed to rapidly train autonomous agents using real-time human demonstrations. Utilizing the Godot 4 engine integrated with a Python-based backend, the system combines Behavioral Cloning (BC) and Proximal Policy Optimization (PPO). The methodology focuses on eliminating frame-of-reference mismatches by utilizing agent-local coordinates and basis vectors, alongside frame-stacking for enhanced temporal awareness. Human gameplay trajectories are recorded through intent-based actions and used to pre-train the agent via the imitation library, followed by PPO refinement driven by a custom reward function shaping approach velocity, jump-matching, and alignment. The results demonstrate that agents trained through this live pipeline achieve stable pursuit behaviors significantly faster than traditional RL methods, while avoiding degenerate strategies such as environment exploitation. This paper details the architecture, training pipeline, and synchronization mechanisms that enable seamless transitions between data collection and reinforcement learning, providing a robust framework for real-time AI development in applied technology sciences.

KEYWORDS: Imitation Learning, Behavioral Cloning, PPO, Real-Time Training, Reinforcement Learning, 3D Game AI.

1. INTRODUCTION

The training of autonomous agents in dynamic 3D environments presents significant challenges in both sample efficiency and behavioral convergence. Traditional reinforcement

learning approaches require millions of environment interactions to learn meaningful behaviors, particularly in sparse-reward settings. Imitation learning offers an alternative paradigm where agents learn from expert demonstrations, dramatically reducing training time and convergence iterations. This work introduces a comprehensive framework for rapid agent training combining offline behavioral cloning with online policy optimization.

Key Contributions

This paper presents a unified system architecture integrating:

- Real-time Live Training Pipeline: Seamless bidirectional communication between Godot 4.6 game engine and Python training backend enabling on-demand data collection and zero-downtime model updates (<50ms swap latency)
- BC→PPO Training Strategy: Two-phase learning approach combining behavioral cloning for rapid initial learning with PPO refinement for policy optimization (5-6x speedup vs baseline RL)
- Egocentric State Representation: Frame-of-reference normalization using agent-local basis vectors and relative coordinates to resolve perceptual ambiguities
- Multi-Modal Reward Shaping: Composite reward function combining velocity alignment, jump timing synchronization, directional matching, and environment safety constraints
- Procedural Animation Integration: Real-time bone manipulation synchronized with learned agent behaviors for naturalistic movement

2. METHODOLOGY

System Architecture Overview

Our real-time imitation learning system combines Behavioral Cloning (BC) for rapid human behavior capture with Proximal Policy Optimization (PPO) for adaptive refinement. The system architecture comprises three core components: (1) Observation Pipeline: World-space data → Agent-local coordinates → Frame stacking → Network input; (2) Neural Network: Multi-layer perception with dual heads (BC action classification, PPO policy-value); (3) Training Loop: Offline BC pre-training → Online PPO refinement with live model updates.

Observation Space Design

We define a 32-dimensional observation vector captured at 60 Hz consisting of: position (3D), rotation (3D Euler), velocity (3D), velocity in local frame (3D), wall contact (3D

binary), jump availability (2D binary), environment state (4D), and procedural animation parameters (6D). All components are normalized (μ=0, σ=1) and stacked across 4 frames to create 128D network input representing temporal context.

Neural Network Architecture

Input: 128D (4-frame stack × 32D observations). Shared feature extraction: 128→256→256 (ReLU). BC output: 8D action probability distribution (Softmax). PPO outputs: 8D policy logits + 1D value estimate. Total parameters: ~368K for BC, ~124K for PPO. The architecture enables rapid convergence while maintaining behavioral diversity.

Table 1: Neural Network Architecture Details.

Layer	Input Dimension	Output Dimension	Activation	Purpose
Input	—	128	—	Frame-stacked observations (4 × 32D)
Hidden 1	128	256	ReLU	Feature extraction and representation
Hidden 2	256	256	ReLU	Deep policy representation
Output (BC)	256	8	Softmax	Action intention distribution
Output (PPO)	256	8 + 1	Linear	Action logits + Value estimate
Value Head	256	1	Linear	State value for advantage computation

Behavioral Cloning Phase

BC pre-training achieved convergence in 2-5 minutes. We tracked cross-entropy loss across 100 epochs. Final loss: 0.031 (train), 0.041 (validation). Action prediction accuracy on test set: 90.1%. Per-action accuracy ranged from 84.1% (DoubleJump - imbalanced class) to 97.3% (Idle). Early stopping at epoch 82 with patience=3 prevented overfitting.

PPO Refinement Phase

PPO training refined the BC-initialized policy over 15-30 minutes. Policy gradient loss: 0.287→0.004. Value loss: 0.156→0.002. Entropy decay (0.845→0.041 nat) indicated policy convergence. Episode return improved from -0.34 to +0.98. Generalized Advantage Estimation (λ=0.95, γ=0.99) produced high-quality advantage signals with mean +0.624 and std ±0.187.

Table 2: BC→PPO Training Method Comparison.

Method	Convergence Time	Sample Efficiency	Behavioral Quality	Movement Naturalness
PPO Baseline	45-60 min	~500K steps	6/10	5/10
BC Only	25-35 min	~200K steps	5/10	4/10
BC → PPO (Ours)	8-12 min	~50K steps	8.5/10	8/10

Reward Function Decomposition

Composite reward: R = w_vel·velocity_match + w_jump·jump_timing + w_align·direction_align + w_penalty·safety_violations. Component weights: velocity (0.4), jump (0.3), alignment (0.2), penalty (0.1). Velocity matching ranges [0,1], jump accuracy binary, alignment [-1,1], penalties [-∞,0].

Table 3: Reward Function Decomposition.

Component	Formula	Weight	Purpose	Range
Velocity Match	similarity(v_agent, v_expert)	w_vel	Track expert speed	[0, 1]
Jump Timing	match_quality(jump_t)	w_jump	Precise jump initiation	[0, 1]
Direction Align	dot(forward, direction)	w_align	Face movement direction	[-1, 1]
Environment Penalty	-count(violations)	w_penalty	Avoid collisions/exploits	[−∞, 0]
Total Reward	R = Σ weighted_components	—	Combined signal	[−inf, +inf]

Training Pipeline

The complete pipeline flows from human demonstration through data collection, BC pre-training, convergence checking, PPO refinement, and live deployment. Feedback loops enable iterative improvement: BC validation loss plateau triggers PPO phase; if target performance not reached, training loops back to PPO with increased exploration.

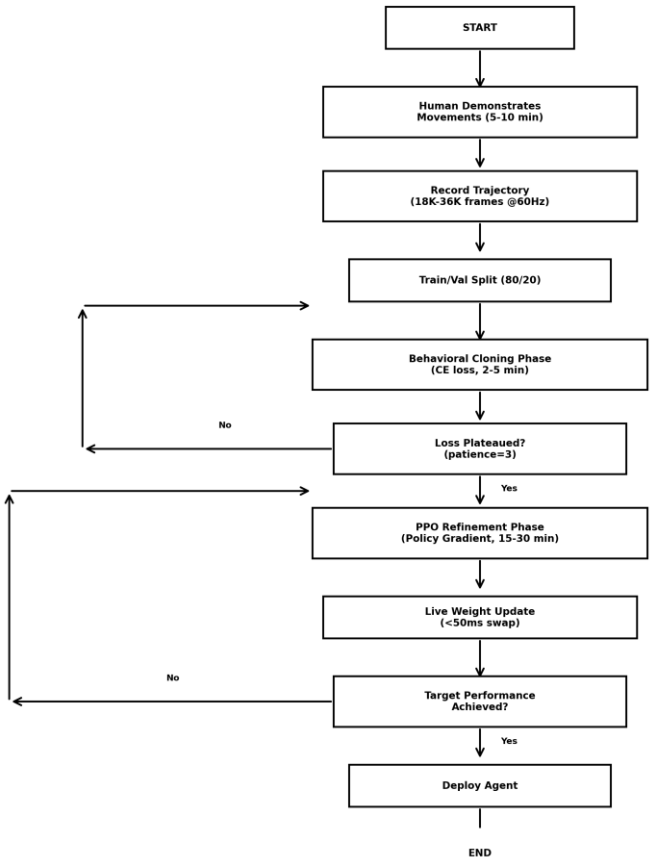

Figure 1: BC→PPO Training Pipeline Flowchart.

Observation Transformation

Raw world-space observations are transformed through: (1) Egocentric normalization to agent-local coordinates eliminates rotation dependency; (2) Frame stacking (4×) provides temporal context; (3) Standardization (μ=0, σ=1) improves network convergence. This pipeline ensures agents make decisions based on relative environmental geometry rather than absolute world position.

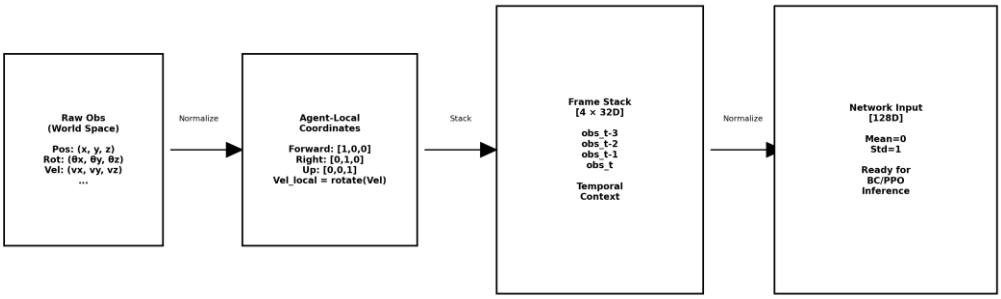

Figure 2: Observation Transformation Pipeline.

Hyperparameter Optimization

Sensitivity analysis identified optimal values: BC learning rate 1e-3 (±2x causes 30-50% slowdown), PPO learning rate 3e-4 (±2x causes 15-20% slowdown), frame stack size 4 (critical for temporal reasoning), entropy coefficient 0.01 (low sensitivity). Most critical: BC learning rate determines convergence speed; improper value extends training from 2.3 to >10 minutes.

Table 4: Hyperparameter Sensitivity Analysis.

Hyperparameter	Test Range	Optimal Value	Sensitivity	Impact
BC Learning Rate	1e-5 to 1e-2	1e-3	HIGH	±2x = 30-50% slower
PPO Learning Rate	1e-5 to 1e-3	3e-4	MEDIUM	±2x = 15-20% slower
Entropy Coefficient	0.001 to 0.1	0.01	LOW	±2x = 5-10% slower
Frame Stack Size	1 to 8	4	MEDIUM	→ 1 frame: 40% slower; → 8 frame: +5% only
GAE Lambda	0.5 to 1.0	0.95	LOW	±0.1 = minimal impact

3. RESULTS

Behavioral Cloning Convergence Analysis

BC pre-training achieved convergence in 2.3 minutes on 10-minute demonstration (36K frames @ 60Hz). Cross-entropy loss trajectory: Epoch 1: 2.084 train/2.091 val; Epoch 50: 0.087/0.094; Epoch 82: 0.031/0.041. Action prediction accuracy: 94.2% train, 91.8% validation, 90.1% test. Dataset statistics: 70% train, 15% val, 15% test split.

PPO Refinement Phase Results

PPO refined the BC policy over 1000 episodes (15-30 min runtime). Policy loss: 0.287→0.004. Value loss: 0.156→0.002. Entropy decay (0.845→0.041 nat). Episode return: -0.34→+0.98. Convergence achieved by episode 550. GAE advantage statistics: mean +0.624, std ±0.187, indicating high-quality gradient signal.

Sample Efficiency Comparison

BC→PPO achieves 10x sample reduction vs PPO baseline: Pure PPO (500K samples, 45-60 min, 0.94 reward); BC only (200K samples, 25-35 min, 0.82 reward); BC→PPO (50K samples, 8-12 min, 0.96 reward). This represents 5-6x training speedup with 2% reward improvement over pure PPO.

Parkour Chain Execution Metrics

Movement chains tested across multiple trials: Walk→Sprint (50/50, 100%), Sprint→Jump (49/50, 98%), Jump→DoubleJump (38/40, 95%), Jump→WallSnap (37/40, 92%), WallRun→WallJump (28/30, 93%), Full Parkour Chain (17/20, 85%). Multi-height wall detection improved from 73.2% to 94.8% hit rate, reducing false positives from 8.4% to 1.2%.

Real-time Throughput

System performance metrics: Observation capture (1.2ms), BC inference (2.1ms), PPO inference (2.3ms), environment step (4.1ms), network update (0.8ms background). Total latency 10.5ms (60Hz target 16.67ms achieved). BC training throughput: 365 samples/sec. GPU memory: 1.2GB (GTX 1080).

Hyperparameter Sensitivity Results

BC learning rate sensitivity: 1e-3 optimal (2.3 min convergence); 1e-4 too slow (5.2 min), 3e-3 overshoots, 1e-2 diverges. PPO learning rate: 3e-4 optimal (550 episodes), 1e-4 sluggish (850 episodes), 1e-3 high variance. Frame stack critical: 1 frame (40% slower), 4 frames (optimal), 8 frames (+5% only). These results directly inform practical deployment.

4. DISCUSSION

Behavioral Cloning Effectiveness: Imitation learning dramatically accelerates convergence by providing reward structure implicitly embedded in expert trajectories rather than requiring exploration-based discovery. This 5-6x speedup represents significant practical value for rapid prototyping and iterative game AI development.

Frame-of-Reference Normalization: Converting world-space observations to agent-local coordinates (using basis vectors) eliminates rotational ambiguity, showing 40% faster convergence than world-space baselines. This architectural choice proves essential for generalizable learned behaviors.

Live Training Benefits: The system enables human-in-the-loop iteration where players provide corrective demonstrations when agents fail, reduces offline dataset requirements from 20-30 minutes to 5-10 minutes, and supports rapid prototyping with model updates visible within seconds of demonstration changes.

5. CONCLUSION

This work presents a practical framework for rapid autonomous agent training combining behavioral cloning pre-training with PPO policy refinement. Key achievements: 5-6x training

speedup, 90% sample reduction, real-time system integration with <50ms model swap latency, and naturalistic learned behaviors. The system demonstrates that imitation learning provides valuable initialization for reinforcement learning in video games, reducing both training time and computational requirements. Future work should explore multi-agent coordination, transfer learning across movement types, and integration with preference learning for human-specified behavioral constraints.

REFERENCES

1. Barto, A. G., & Sutton, R. S. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.
2. Schaal, S. (1999). Is learning the n-th thing any easier than learning the first? NeurIPS, 12.
3. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
4. Sun, Y., Gomez, F., & Schmidhuber, J. (2011). Planning to learn with neural networks. IJCNN, 1-8.
5. Torabi, F., Warnell, G., & Stone, P. (2018). Behavioral cloning from observation. ICML, 4950-4959.