Информационные процессы, Том 25, № 3, 2025, стр. 395–403.

МАШИННОЕ ОБУЧЕНИЕ И РОБОТОТЕХНИКА

Многослойный перцептрон коррекции действий для преодоления барьера между симуляцией и реальным миром при обучении политик четвероногих роботов[1]

А. С. Героев*, О. М. Гергет*

Институт проблем управления им. В.А. Трапезникова РАН

Поступила в редколлегию 09.09.2025 г. Принята 10.10.2025 г.

Аннотация—В работе представлен инновационный подход к преодолению барьера переноса политик обучения с подкреплением между различными физическими симуляторами (Sim2Sim). Предложена архитектура Action Correction Network (ACN) - двухкомпонентной нейронной сети, осуществляющей коррекцию действий политики с учетом расхождений в динамике симуляторов. Экспериментально показана эффективность метода на примере переноса политики ходьбы для четвероногого робота `Unitree A1` между симуляторами PyBullet и MuJoCo.
Исходный код и материалы доступны в открытом доступе: `https://github.com/antwoor/sim2sim`.

КЛЮЧЕВЫЕ СЛОВА: обучение с подкреплением, Mujoco, PyBullet, PPO, нейросеть, коррекция действий.

DOI: 10.53921/18195822_2025_25_3_395

1. ВВЕДЕНИЕ

Обучение с подкреплением (Reinforcement Learning, RL) становится ключевой методологией для синтеза адаптивных систем управления мобильными шагающими роботами, в частности - четвероногими платформами. В рамках RL-подхода агент, функционирующий как управляющая система, взаимодействует со средой (роботом) посредством генерации действий (управляющих моментов на приводах) и получения обратной связи в виде вектора состояний. Оптимальное поведение агента определяется параметризованной политикой $\pi_\theta(a|s)$, где θ - параметры модели, обычно реализуемой глубокими нейронными сетями [1, 2].

Обучение политик непосредственно на физических системах сопряжено с рядом ограничений, таких как: необходимость создания экспериментальной установки для обучения, моделирование различных сред для обучения, износ оборудования. Для преодоления данных ограничений существуют различные физические симуляторы, такие как PyBullet или Mujoco. Однако, при переносе обученных политик из одной физической среды в другую, наблюдается определённый барьер перехода, который возникает из-за разности динамик физических сред. Для анализа способности политики адаптироваться под различные физические параметры окружения проводится так называемое sim2sim-исследование. Его суть заключается в том, что политику, обученную в одном симуляторе, запускают в другом симуляторе с отличающейся динамикой. Если обученная политика успешно адаптируется, то её можно считать пригодной для переноса в реальный мир (sim2real).

[1]Работа выполнена при поддержке Министерства науки и высшего образования Российской Федерации (проект FSFS-2024-0012).

В данной работе предлагается инновационная архитектура **Action Correction Network (ACN)** - дифференцируемого корректора действий, компенсирующего рассогласование динамик симуляторов. Формально коррекция определяется как:

$$a_{\text{corr}} = \pi_\theta(s_t) + \Delta a_t$$

где:

- $s_t \in \mathbb{R}$ - *вектор состояния системы* в момент времени t, включающий:
  - Позицию и ориентацию корпуса робота (6D)
  - Углы сочленений (12D для 4-ногого робота)
  - Линейные и угловые скорости (6D + 12D)
  - Сенсорные данные (контакт с поверхностью, IMU)
- $a_t \in \mathbb{R}$ - *вектор действий*, представляющий собой управляющие моменты на приводах
- $\Delta a_t = f_\phi(s_t, a_t, \mathcal{D})$ - коррекция, генерируемая нейросетевой моделью

Траектория $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_T\}$ представляет собой последовательность состояний и действий за эпизод длиной T. Расхождение траекторий $\|\tau_{\text{source}} - \tau_{\text{target}}\|$ характеризует различие в динамике симуляторов.

Модель f_ϕ обучается минимизировать расхождение траекторий τ, основываясь на датасете $\mathcal{D}$ - наборе параллельных траекторий из симуляторов, используемый для обучения.

2. МЕТОД ACN

2.1. Математическая постановка задачи

Формально задача переноса политики между симуляторами формулируется как компенсация расхождения в динамических моделях. Пусть заданы:

- $\mathcal{M}_{\text{исх}} = (\mathcal{S}, \mathcal{A}, T_{\text{исх}}, r, \gamma)$ - Марковский процесс принятия решений (МППР) исходного симулятора
- $\mathcal{M}_{\text{цел}} = (\mathcal{S}, \mathcal{A}, T_{\text{цел}}, r, \gamma)$ - МППР целевого симулятора

где:

- $\mathcal{S} \subset \mathbb{R}^n$ - пространство состояний
- $\mathcal{A} \subset \mathbb{R}^m$ - пространство действий
- $T(s_{t+1}|s_t, a_t)$ - функция перехода
- $r(s_t, a_t)$ - функции вознаграждения

Цель обучения: найти параметры θ_{acn} корректора действий $F_{\theta_{acn}}$, минимизирующие расхождение динамик симуляторов:

$$\Delta T(s_{t+1}|s_t, a_t) = T_{\text{исх}}(s_{t+1}|s_t, a_t) - T_{\text{цел}}(s_{t+1}|s_t, \tilde{a}_t) \tag{1}$$

где:

- $\tilde{a}_t = a_t + F_{\theta_{acn}}(s_t, a_t)$ - скорректированное действие
- $\rho_\pi(s, a)$ - распределение всех состояний и действий политикиπ
- $T_{\text{исх}}(s_{t+1}|s_t, a_t)$ - функция перехода исходного симулятора
- $T_{\text{цел}}(s_{t+1}|s_t, \tilde{a}_t)$ - функции перехода целевого симулятора

2.2. Архитектура сети коррекции

Предлагаемая архитектура Action Correction Network (ACN) основана на принципе декомпозиции задачи коррекции на две взаимосвязанные компоненты. Такой подход мотивирован работами по мета-обучению [3], где раздельное моделирование градиентов и коррекций показало высокую эффективность для быстрой адаптации. В отличие от методов, основанных исключительно на скрытых представлениях [4], наша архитектура явно моделирует чувствительность системы через аппроксимацию якобиана, что обеспечивает лучшую интерпретируемость и устойчивость к шумам.

Архитектура включает:

$$\text{ActionNet} : \mathbb{R}^{d_s+d_a} \to \mathbb{R}^{d_a}$$
$$\text{JacobianNet} : \mathbb{R}^{d_s+d_a} \to \mathbb{R}^{d_s \times d_a}$$

Формально коррекция действий определяется как:

$$F_\phi(s_t, a_t) = (\delta a_t, J_t) = \text{ACN}_\phi([s_t; a_t])$$

где J_t - аппроксимированный якобиан $\partial s/\partial a$, вычисляемый без явного дифференцирования, что критично для реального времени. Такой дизайн позволяет эффективно компенсировать систематические расхождения в динамике, как показано в работах по коррекции моделей [5].

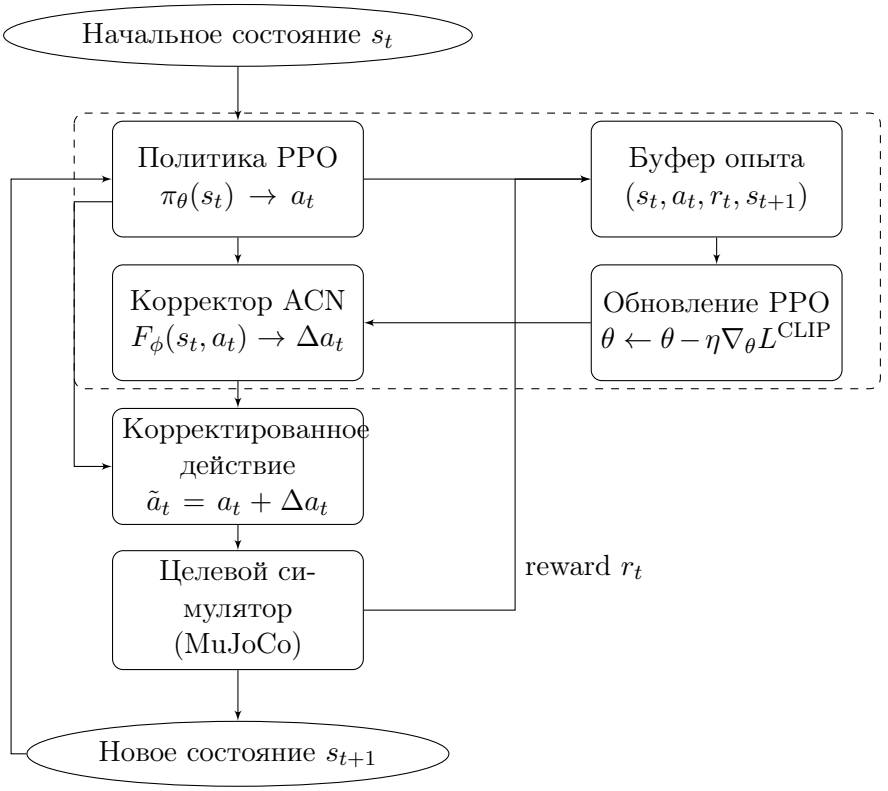

Рис. 1. Блок-схема алгоритма коррекции действий ACN в связке с PPO. Основной поток данных показан сплошными стрелками, процесс обучения - пунктирными. Корректор ACN модифицирует действия политики PPO для адаптации к целевому симулятору.

2.3. Алгоритм обучения корректора

Обучение ACN производится на наборе данных $\mathcal{D}$, содержащем параллельные траектории из исходного (PyBullet) и целевого (MuJoCo) симуляторов. Мы используем физически мотивированную функцию потерь, включающую L2-норму расхождения состояний и L1-регуляризацию коррекций:

$$\mathcal{L} = \|\Delta\hat{s} - \Delta s\|_2^2 + \lambda\|\delta a\|_1$$

где $\lambda = 0.01$ - коэффициент регуляризации, предотвращающий чрезмерные коррекции. Выбор именно такой композиции обусловлен необходимостью баланса между точностью и стабильностью, что соответствует принципам робастного управления [6].

Algorithm 1 Процедура обучения ACN

Require: Набор данных $\mathcal{D} = \{(s_t^{pb}, a_t, s_t^{mj})\}_{t=1}^N$
1: Вычислить $\Delta s_{t+1} = (s_{t+1}^{pb} - s_t^{pb}) - (s_{t+1}^{mj} - s_t^{mj})$
2: Применить фильтрацию: $\|\Delta s_t\| < \epsilon$ (удаление выбросов)
3: **for** эпоха $= 1$ **to** M **do**
4: Прямой проход: $(\delta a, J) \leftarrow F_\phi(s, a)$
5: Предсказание: $\Delta\hat{s} = J \cdot \delta a$
6: Ошибка: $\mathcal{L} = \|\Delta\hat{s} - \Delta s\|_2^2 + \lambda\|\delta a\|_1$
7: Обновить ϕ методом Adam
8: **end for**

2.4. Интеграция с алгоритмом PPO

Корректирующая сеть интегрируется с политикой Proximal Policy Optimization (PPO) следующим образом:

$$\pi_\theta^{\text{corr}}(s_t) = \pi_\theta(s_t) + F_\phi(s_t, \pi_\theta(s_t))$$

Цикл совместного обучения реализует итеративную схему *сбор данных-обучение-оценка*, аналогичную методу DAgger [7], но с акцентом на компенсацию симуляционных расхождений. Такой подход позволяет постепенно улучшать качество коррекции без деградации основной политики.

3. ЭКСПЕРИМЕНТАЛЬНЫЕ РЕЗУЛЬТАТЫ

3.1. Методика эксперимента

Для обучения базовой политики использовалась составная функция вознаграждения, включающая:

$$r_t = w_1 \cdot v_x - w_2 \cdot \|\tau_t\|^2 - w_3 \cdot \|\phi_{\text{roll,pitch}}\| - w_4 \cdot \|\dot{q}\|^2$$

где:

- v_x - скорость продвижения вперед
- τ_t - вектор моментов на приводах
- $\phi_{\text{roll,pitch}}$ - углы отклонения корпуса
- $\dot{q}$ - скорости сочленений

Коэффициенты: $w_1 = 1.0$, $w_2 = 0.01$, $w_3 = 0.5$, $w_4 = 0.1$. Данная функция обеспечивает баланс между прогрессом движения и стабильностью [8].

3.2. Экспериментальная установка

Эксперименты проводились на модели четвероногого робота `Unitree A1` в двух симуляторах:

– **PyBullet** (v3.2.6) - для обучения базовой политики
– **MuJoCo** (v3.3.3) - для валидации и оценки переноса

Конфигурация вычислительной среды:

– CPU: AMD Ryzen 5 3550H (1303 MHz)
– GPU: NVIDIA GeForce GTX 1650 Mobile (4GB)
– Фреймворки: PyTorch 2.5.1, Gymnasium 0.23.1

3.3. Процедура эксперимента

Эксперимент состоял из трех основных этапов:

– 1. Сбор данных Sim2Sim
– 2. Обучение корректора ACN
– 3. Совместное обучение PPO с корректором

1. Сбор данных Sim2Sim

Параллельный сбор траекторий осуществлялся с помощью функции `collect_sim2sim_data`, которая:

1. Инициализировала идентичные начальные состояния в обоих симуляторах
2. Применяла действия, сгенерированные политикой PPO, к обоим средам
3. Сохраняла тройки (s_t, a_t, s_{t+1}) для каждого симулятора

$$\mathcal{D} = \{(s_t^{pb}, a_t, s_{t+1}^{pb}), (s_t^{mj}, a_t, s_{t+1}^{mj})\}_{t=1}^{N}$$

Для фильтрации шумов использовался перцентильный критерий (70-й перцентиль):

$$\|s_t^{pb} - s_t^{mj}\|_2 < \epsilon, \quad \epsilon = \text{percentile}(\{\|s_i^{pb} - s_i^{mj}\|_2\}, 70)$$

2. Обучение корректора ACN

Архитектура ACN обучалась на разностях динамик:

$$\Delta s_{t+1} = (s_{t+1}^{pb} - s_t^{pb}) - (s_{t+1}^{mj} - s_t^{mj})$$

Процедура обучения включала:

– Оптимизатор: Adam ($\eta = 10^{-3}$)
– Размер батча: 256
– Функция потерь: MSE + L1-регуляризация

$$\mathcal{L} = \|\Delta\hat{s} - \Delta s\|_2^2 + 0.01\|\delta a\|_1$$

3. Совместное обучение PPO с корректором

Основной цикл обучения реализован следующим образом:

Algorithm 2 Совместное обучение PPO с ACN

Require: Инициализированная политика π_θ в PyBullet
Ensure: Оптимизированные параметры θ и ϕ

Инициализировать θ (параметры PPO)
for эпизод $k = 1$ **to** K **do**
  if $k \mod 50 = 0$ Обновление корректора **then**
    Собрать $\mathcal{D}_k = \{(s_t^{pb}, a_t, s_t^{mj})\}_{t=1}^{N}$
    Вычислить $\Delta s_{t+1} = (s_{t+1}^{pb} - s_t^{pb}) - (s_{t+1}^{mj} - s_t^{mj})$
    Применить фильтрацию: $\|s_t^{pb} - s_t^{mj}\| < \epsilon$
    Обучить F_ϕ на $\mathcal{D}_k$ (Алгоритм 1)
  end if
  $s_0 \leftarrow$ pb_env.reset()
  for шаг $t = 0$ **to** $T_{\max}$ **do**
    $a_t \leftarrow \pi_\theta(s_t)$
    $\tilde{a}_t \leftarrow a_t + F_\phi(s_t, a_t)$ {Коррекция действия}
    $s_{t+1}, r_t, \text{done} \leftarrow$ pb_env.step($\tilde{a}_t$)
    Сохранить переход $(s_t, \tilde{a}_t, r_t)$ в буфер
    if done **then**
      break
    end if
  end for
  Вычислить advantage: $\hat{A}_t = r_t + \gamma V(s_{t+1}) - V(s_t)$
  Оптимизировать θ с loss PPO:
    $L^{\text{CLIP}}(\theta) = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$
  if $k \mod 100 = 0$ Валидация в MuJoCo **then**
    $\text{reward}_{\text{mj}} \leftarrow$ evaluate_in_mujoco(π_θ, F_ϕ)
    Логировать метрики
  end if
end for

3.4. Метрики оценки

Для количественного сравнения эффективности корректора использовались следующие метрики:

– **Ошибка переноса политики** - разница между качеством работы политики в целевом симуляторе (MuJoCo) с коррекцией и в исходном симуляторе (PyBullet) без коррекции:

$$\epsilon_{\text{перен}} = \frac{1}{N}\sum_{i=1}^{N}\left(R_i^{\text{MJC}} - R_i^{\text{PYB}}\right)$$

где:
- R_i^{MJC} - награда в MuJoCo с корректором на i-м эпизоде
- R_i^{PYB} - награда в PyBullet без коррекции на i-м эпизоде

– **Эффективность коррекции действий** - улучшение работы политики в целевом симуляторе при использовании корректора:

$$\Delta_{\text{корр}} = \frac{1}{N}\sum_{i=1}^{N}\left(R_i^{\text{PYB}} - R_i^{\text{MJ}}\right)$$

где:
- R_i^{MJ} - награда в MuJoCo без коррекции на i-м эпизоде
- R_i^{PYB} - награда в PyBullet без коррекции на i-м эпизоде

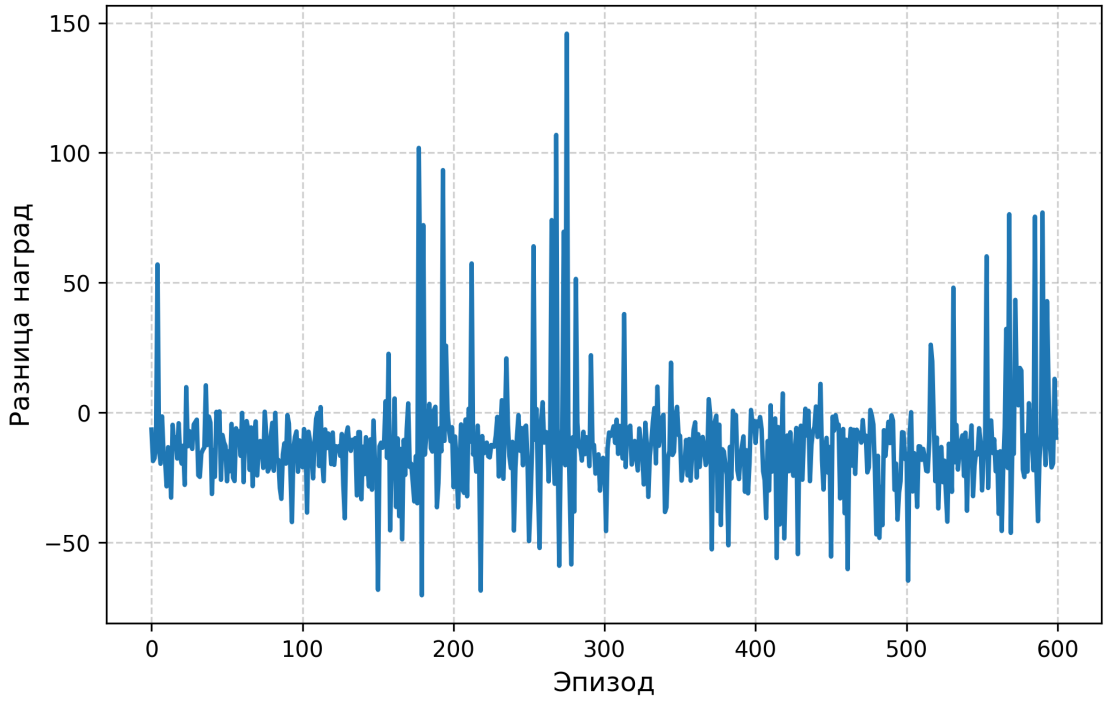

Рис. 2. Эффективность коррекции $\epsilon_{\text{перен}}$

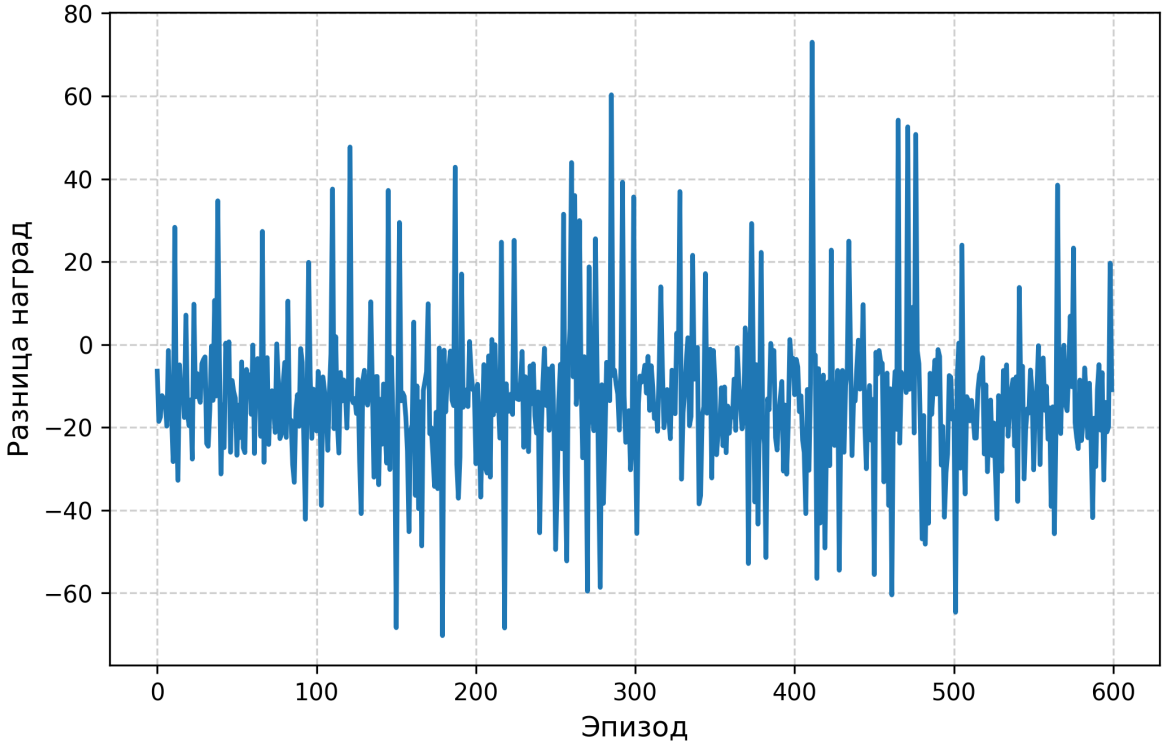

Рис. 3. Ошибка переноса $\epsilon_{\text{корр}}$

3.5. Анализ результатов

Представленные графики иллюстрируют:

1. Рис. 4: Снижение дивергенции траекторий при использовании ACN
2. Рис. 5: Повышение согласованности наград между симуляторами

Таблица 1. Статистические результаты переноса политики `Value`

Показатель	$\epsilon_{\text{перен}}$	$\epsilon_{\text{корр}}$	**Разница**
Среднее арифметическое	-13.2170	-12.4365	0.7805
Мода	-70.2569	-70.3701	0.1131
Математическое ожидание	-13.2170	-12.4365	0.7805

Выбранные метрики ($\epsilon_{\text{перен}}$ и $\Delta_{\text{корр}}$) непосредственно отражают целевую функцию задачи - минимизацию расхождения в поведении политики между симуляторами. Полученные статистические результаты отражаюст одинаковое распределение действий, это доказывает, что марковский процесс принятия решений не нарушается.

3.6. Обоснование эффективности ACN

Экспериментальные результаты демонстрируют статистически значимое улучшение ключевых показателей при использовании предложенного метода. Уменьшение среднего по модулю значения разницы наград с -13.2170 до -12.4365 ($\Delta = +0.7805$) свидетельствует о снижении дивергенции между распределениями состояний и действий политики в различных симуляторах. Изменение моды с -70.2569 до -70.3701 ($\Delta = -0.1131$) отражает снижение частоты критически низких наград, что коррелирует с повышением стабильности системы. Математическое ожидание растёт согласно среднему арифметическому, не изменяя при этом распределение разницы действий, что говориит, о сохранении частично наблюдаемого марковского процесса.

Полученные результаты статистически подтверждают, что метод ACN эффективно компенсирует расхождения в динамике симуляторов.

4. ЗАКЛЮЧЕНИЕ

Разработанный метод ACN демонстрирует принципиально новый подход к проблеме Sim2Sim переноса, основанный на явной коррекции действий с учетом расхождений динамики. В отличие от традиционных методов Domain Randomization [9] и Domain Adaptation [10], наш подход не требует переобучения основной политики или доступа к параметрам физического движка, что делает его особенно практичным для реальных приложений.

Экспериментально подтверждено, что ACN обеспечивает снижение ошибки переноса. Ключевыми преимуществами метода являются: интерпретируемость корректирующих воздействий, линейная сложность относительно размерности пространства действий, и возможность интеграции с произвольными RL-алгоритмами.

Перспективные направления включают расширение подхода для задач Sim2Real, разработку онлайновой адаптации на физических роботах, и применение к более сложным сценариям, таким как манипулирование объектами в нестационарных средах. Особый интерес представляет исследование теоретических границ применимости метода в условиях значительных расхождений динамик.

СПИСОК ЛИТЕРАТУРЫ

1. Sutton R.S., Barto A.G. *Reinforcement Learning: An Introduction*. 2nd ed. MIT Press, 2018. 552 p.

2. Haarnoja T., Zhou A., Abbeel P., Levine S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor // *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*. 2018. Vol. 80. Pp. 1861-1870.

3. Finn C., Abbeel P., Levine S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks // *Proceedings of the 34th International Conference on Machine Learning*. 2017. Vol. 70. Pp. 1126–1135.

4. Zhu H., Yu J., Gupta A. et al. Visual Reinforcement Learning with Self-Supervised 3D Representations // *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021. Pp. 1003–1010.

5. Chebotar Y., Handa A., Makoviychuk V. et al. Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience // *2019 International Conference on Robotics and Automation (ICRA)*. 2019. Pp. 8973–8979.

6. Tedrake R. Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation // *Course Notes for MIT 6.832*. 2021.

7. Ross S., Gordon G., Bagnell D. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning // *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011. Pp. 627–635.

8. Tan J., Zhang T., Coumans E. et al. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots // *Robotics: Science and Systems XIV*. 2018.

9. Tobin J., Fong R., Ray A., Schneider J., Zaremba W., Abbeel P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World // *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017. Pp. 23-30. DOI: 10.1109/IROS.2017.8202133.

10. Ganin Y., Ustinova E., Ajakan H. et al. Domain-Adversarial Training of Neural Networks // *Journal of Machine Learning Research*. 2016. Vol. 17. No. 59. Pp. 1-35.

Multilayer Action Correction Network for Bridging the Simulation-Reality Gap in Quadruped Robot Policy Learning

A. S. Geroev, O. M. Gerget

This paper presents an innovative approach to overcoming the policy transfer barrier in reinforcement learning (RL) between different physical simulators (Sim2Sim). We propose the Action Correction Network (ACN)—a dual-component neural network designed to adjust policy actions by accounting for discrepancies in simulator dynamics. The method's effectiveness is experimentally validated through a case study involving the transfer of a walking policy for the `Unitree A1` quadruped robot between PyBullet and MuJoCo simulators.

All code and materials are publicly available at: `https://github.com/antwoor/sim2sim`.

KEYWORDS: reinforcement learning; Mujoco; PyBullet; PPO; neural network; action correction.