Министерство науки и высшего образования Российской Федерации

Государственное Автономное Образовательное Учреждение Высшего Образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет по тестовому заданию
Методы обучения с подкреплением для реализации энергоэффективной локомоции

Автор : Терещенко Евгений Константинович

Номер ИСУ: 507632

Преподаватель: Колюбин Сергей Алексеевич

Дата: 9 февраля 2026 г.

1 Введение

Локомоция шагающих роботов — это управление системой с *гибридной* динамикой (контакт/полет), сильной нелинейностью, неполной наблюдаемостью, задержками приводов и трудно-моделируемыми эффектами взаимодействия с опорой (трение, комплаенс, микроскольжения). На практике это означает, что даже при наличии хорошей модели для модельно-ориентированных методов (MPC/оптимальное управление) остаётся существенный «хвост» немоделируемых эффектов, а при усложнении поведения (скорость, резкие повороты, неструктурированный рельеф, переходы между режимами) ручная настройка контроллера быстро становится узким местом.

Обучение с подкреплением (reinforcement learning, RL) стало практической альтернативой/надстройкой над классическими подходами, потому что позволяет методом *"проб и ошибок* синтезировать policy для управления в задачах, где (i) контакты делают градиенты нестабильными и плохо определёнными, (ii) точная модель недоступна или слишком дорога для онлайн-оптимизации, (iii) «правильное» поведение проще *выучить* из взаимодействия, чем выписать в виде набора эвристик. В современном контуре управления quadruped-роботов обучение с подкреплением часто используется как компонент, который учится компенсировать систематические ошибки модели и расширять рабочие режимы, в то время как «скелет» устойчивости и ограничений задаёт модельно-ориентированный блок (например, MPC). Это иллюстрирует RL-augmented MPC, где MPC обеспечивает предсказуемость и работу с ограничениями, а RL добавляет адаптивное поведение и робастность в сложных динамических режимах.[1]

Формально стандартная постановка RL задаётся как марковский процесс принятия решений (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, где $s_t \in \mathcal{S}$ — состояние (или наблюдение), $a_t \in \mathcal{A}$ — действие, $p(s_{t+1} \mid s_t, a_t)$ — динамика, $r(s_t, a_t)$ — награда, $\gamma \in (0, 1)$ — дисконт. Ищется политика $\pi_\theta(a \mid s)$, максимизирующая ожидаемую

дисконтированную сумму наград:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\Big[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\Big], \qquad \tau = (s_0, a_0, s_1, \dots). \tag{1.1}$$

В задачах локомоции важны (a) физически согласованный интерфейс управления (что именно выдаёт политика), (b) корректные метрики качества (устойчивость, энергетическая стоимость действия (CoT), гладкость, ударные нагрузки), поскольку именно они определяют переносимость на реальный робот и энергоэффективность.

В рамках данного отчёта рассматриваются State of the Art подходы в следующих направлениях:

1. **Sim-to-real gap**: перенос политики из симуляции в реальность через робастификацию (domain randomization), онлайн-адаптацию (контекст/латентные параметры) или гибридные контуры управления.[1]
2. **Energy-efficient locomotion**: оптимизация энергоэффективности через физически-информированные цели/регуляризаторы и архитектурные ограничения, дающие измеримый выигрыш в CoT и мощности.[2]
3. **Hierarchical systems**: long-horizon задачи требуют разделения на уровни (план/параметры/низкоуровневые навыки) и механизмов устойчивого «склеивания» навыков.[3]
4. **Physically-consistent pipelines**: интеграция физической структуры (MPC, импеданс/PD-интерфейс, динамические объекты) снижает хрупкость обучения и упрощает перенос.[1], [4]

1.1 Цель практической части

После анализа SOTA подходов и выведения гипотезы, которую хочется рассмотреть в данном отчете, будут так же представлены практические результаты работы, которые должны подтвердить или опровергнуть гипотезу. К сожалению, в данной работе в связи с ограниченностью по времени не получилось простроить полноценный пайплайн с большими языковыми моделями, поэтому

для грубой проверки целесообразности гипотезы использовалась лишь часть пайплана основанная на Proximal Policy Optimization (PPO) алгоритме и блоке MPC.

2 State of the Art подходы

2.1 Интерфейсы управления и физическая согласованность (action/observation design)

Даже при одинаковом RL-алгоритме поведение радикально зависит от того, *что именно* выдаёт политика. В роботах часто используют позиционные команды суставов (удобно для реального железа и симуляторов), а торки реализуются PD-контуром:

$$a_t = \hat{q}_t \in \mathbb{R}^{n_q}, \qquad \tau_t = K_p(\hat{q}_t - q_t) + K_d(\hat{\dot{q}}_t - \dot{q}_t), \tag{2.1}$$

где обычно $\hat{\dot{q}}_t = 0$, а K_p, K_d фиксированы. Такой интерфейс целенаправленно выбирается в работах, ориентированных на перенос: например, в RMA (Rapid Motor Adaptation)[4] статье политика выдаёт целевые углы суставов, которые реализуются через PD-контроль с фиксированными коэффициентами.[4]

Альтернативы: (i) прямое управление торками, (ii) команды в пространстве опорных реакций/ускорений базы, (iii) «остаточное» управление (residual), где RL выдаёт поправку к модельно-ориентированному контроллеру. Последний вариант особенно важен для *physically-consistent pipelines*: базовый блок гарантирует ограничения и устойчивость, а RL учится компенсировать систематические ошибки модели.

Показательный пример — RL-augmented MPC: MPC решает задачу по опорным силам/планированию, а RL добавляет *динамические компенсации* и *рефлексы swing-ноги*, расширяя рабочий диапазон скоростей, поворотов и сценариев контакта.[1]

2.2 Метрики качества: от устойчивости до энергоэффективности

2.2.1 Базовые метрики локомоции

Для low-level локомоции почти всегда оценивают:

- **успешность/время до падения**: success rate, time-to-fall (TTF), distance covered;
- **треккинг команд**: ошибки по линейной/угловой скорости (RMSE/MAE), ориентация базы (roll/pitch);
- **гладкость**: нормы Δa_t, jerk, производная торков;
- **робастность**: распределения по трению, массе, задержкам, рельефу, payload.

Например, в RMA приводится сравнительный набор метрик (success, TTF, distance, torque, smoothness, ground impact и др.) для различных baselies.

2.2.2 Энергетические метрики и Cost of Transport

Для энергоэффективности важны метрики, которые (a) физически интерпретируемы, (b) сопоставимы между роботами/скоростями. Стандартный выбор — **Cost of Transport (CoT)**. В типичном виде:

$$\mathrm{CoT} \;=\; \frac{1}{mgv}\,\Big\langle \sum_{i=1}^{n_q} \big|\tau_i\,\dot{q}_i\big| \Big\rangle, \tag{2.2}$$

где m — масса робота, g — ускорение свободного падения, v — скорость, $\langle\cdot\rangle$ означает усреднение по времени (или по пройденной дистанции). В работах по энергоэффективности CoT прямо вычисляют через механическую мощность $\tau^\top \dot{q}$ и анализируют улучшения при добавлении физически информированных термов.[2]

2.2.3 Метрики long-horizon

В long-horizon задачах метрика треккинга скорости недостаточна: задача может включать манипуляцию и последовательность действий. Поэтому используют **success rate** выполнения всей задачи и дополнительные меры «достижения цели» (например, нормированную дистанцию до цели).[3]

2.3 Чистый model-free RL как моторный контроллер

Наиболее распространённый алгоритмический базис для локомоции — policy gradient семейства PPO/SAC. Для PPO оптимизируется суррогатная цель с клиппингом:

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)\hat{A}_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big], \qquad r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)}. \tag{2.3}$$

Эта форма удобна для отчёта как «де-факто стандарт» обучения устойчивых локомоторных политик в симуляции.

Сильные стороны чистого RL:

- минимальные предположения о модели контакта/трения;
- возможность «впитать» сложные стабилизирующие реакции из опыта;
- естественная параметризация под high-dimensional наблюдения.

Слабые стороны:

- высокая стоимость данных (без симуляции);
- хрупкость к sim-to-real разрыву;
- чувствительность к reward shaping и нежелательным «читерским» стратегиям.

Отсюда возникают два класса улучшений: (i) физически-информированные цели/приоры, (ii) архитектурная интеграция с модельно-ориентированными блоками и/или иерархией.

2.4 Energy-efficient locomotion: физически-информированные цели и измеримые выигрыши

2.4.1 Физически-информированные значения(например IMF) и снижение CoT

В работах по энергоэффективности ключевая идея состоит в том, чтобы дополнять «базовые» награды (трекинг + стабилизация) значениями, которые напрямую отражают физическую причину потерь энергии (удары, чрезмерная жёсткость, высокочастотные торки, неэффективная работа с инерцией). В частности,[2] показывает, что добавление физически-информированного терма к базовым схемам (включая имитационные приоры) приводит к систематическому снижению CoT и/или механической мощности при сохранении устойчивости. Например, в[2] reported снижение CoT на плоской поверхности на 18.4% при скорости 1 м/с (для одной из конфигураций), а также улучшения на неровном рельефе с одновременным улучшением метрик ориентации/треккинга. Эти результаты важны как эмпирическое подтверждение тезиса: *физически согласованный reward shaping* может улучшать энергоэффективность без деградации качества локомоции.

2.4.2 Архитектурные ограничения на управляющий сигнал (пример: Tacit Learning)

Другой класс идей — улучшать энергоэффективность не только через reward, но и через структурирование управляющего сигнала. В[5] предлагается представ-

лять итоговое управление как сумму RL-выхода и дополнительной компоненты, использующей историю управления:

$$u(t) = u_{\mathrm{RL}}(t) + u_{\mathrm{TL}}(t), \tag{2.4}$$

где u_{TL} строится как функционал/фильтрация прошлых u_{RL} (формула задаётся в[5]). В[5] сообщаются улучшения энергоэффективности порядка 7.07% (simulation, среднее) и 11.02% (real, среднее), с максимальными улучшениями, достигающими $\approx$ 13–14%. Для обзора это полезно как иллюстрация: часть энергетических потерь может быть связана с высокочастотной «суетой» приводов, которую можно подавлять архитектурно (в дополнение к reward).

2.5 Hybrid control: RL-augmented MPC как physically-consistent pipeline

2.5.1 Архитектура и интерфейс

RL-augmented MPC объединяет модельно-ориентированный контур (MPC) и обучаемый RL-модуль, который добавляет *адаптивное поведение*: (i) компенсацию неопределённостей динамики и внешних возмущений (adaptive balancing), (ii) реактивную модификацию swing-траекторий (swing foot reflection) при неожиданных контактах.[1] Общая архитектура разделена на два уровня частот (Fig. 2.1): высокоуровневый контур на $\approx$ 33 Hz (синяя рамка) и низкоуровневый контур на $\approx$ 1 kHz (красная рамка).[1]

Архитектура данного подхода выглядит следующим образом:

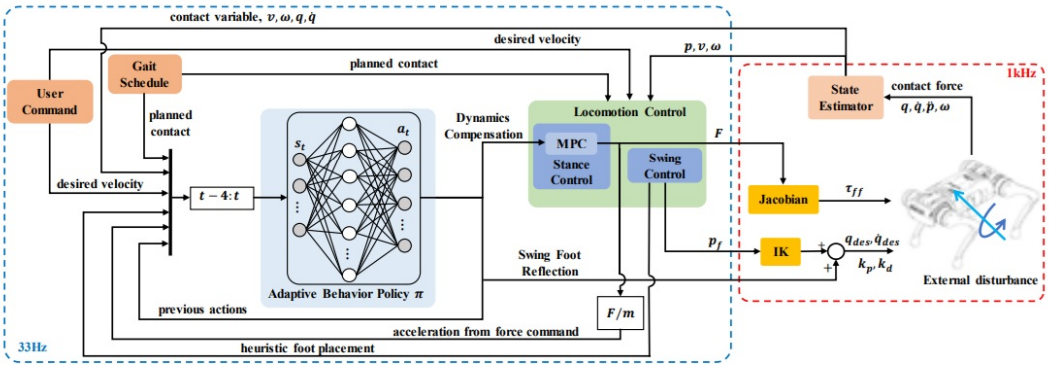

Рис. 2.1: Архитектура RL-augmented MPC

С данной архитектурой проще всего разобраться, если рассматривать ее как композицию блоков:

1. **Команда пользователя и расписание походки.** Пользователь задаёт желаемые команды по скорости/угловой скорости $(v_{\text{des}}, \omega_{\text{des}})$. Event-driven конечный автомат (FSM) формирует *gait schedule* и план контактов s_ϕ (planned contact) для каждой ноги.

2. **Контроль локомоции** Внутри данного блока разделены:

   - **Stance control (MPC).** Конвексный MPC оптимизирует опорные реакции F_i (GRF) для ног в фазе опоры (stance), используя упрощённую модель (SRB/centroidal) и план контактов.[1]

   - **Swing control.** Для swing-ног вычисляются желаемые положения стоп p_f на основе эвристической постановки ноги (heuristic foot placement) и параметризации swing-траектории (Bezier).[1]

3. **RL-модуль как "плагин" к MPC.** Центральный элемент — *adaptive behavior policy* π, работающая на 33 Hz. Её задача — выдавать *добавочные* управляющие воздействия, которые накладываются на номинальный MPC-контур:

   $$a_t = \left[\Delta\alpha_t,\ \Delta a_t,\ \Delta q_t\right],$$

   где $\Delta\alpha_t \in \mathbb{R}^3$ и $\Delta a_t \in \mathbb{R}^3$ — угловая и линейная *компенсации-ускорения* для stance-контроля (динамическая компенсация), а Δq_t — смещение суставных углов для *swing foot reflection*.[1]

4. **Низкоуровневое управление ногами и hardware loop (1 kHz).** Силовые команды F из MPC переводятся в feedforward-торки через Якобиан:

$$\tau_{\text{ff}} = J(q)^\top F,$$

а желаемые позиции стоп p_f преобразуются в целевые суставные углы через inverse kinematics (IK):

$$q_{\text{IK}} = \text{IK}(p_f).$$

Далее целевые углы для swing-ног модифицируются отражением:

$$q_{\text{des}} = q_{\text{IK}} + \Delta q_t,$$

и итоговый момент формируется PD-контуром (вместе с τ_{ff}):

$$\tau = \tau_{\text{ff}} + K_p\,(q_{\text{des}} - q) + K_d\,(\dot{q}_{\text{des}} - \dot{q}),$$

где состояние $(q, \dot{q})$ и оценки базы поступают от state estimator (Kalman filter), работающего в 1 kHz.[1]

Ключевой идеей статьи является выдача политикой *ускорения* $\Delta\alpha, \Delta a$, которые встраиваются в уравнения SRB-модели, используемой внутри MPC[1] вместо того чтобы обучать политику выдавать силы/моменты (зависящие от m и I конкретного робота). В статье это реализовано как добавка к непрерывной модели:

$$\frac{d}{dt}\begin{bmatrix}\Theta\\ p\\ \omega\\ \dot{p}\end{bmatrix} = A(\psi)\begin{bmatrix}\Theta\\ p\\ \omega\\ \dot{p}\end{bmatrix} + B(r_{1:n}, \hat{I}, m)\begin{bmatrix}F_1\\ \vdots\\ F_n\end{bmatrix} + \begin{bmatrix}0\\ 0\\ \Delta\alpha\\ \Delta a + g\end{bmatrix}, \tag{2.5}$$

где $\Theta = [\phi, \theta, \psi]^\top$ — ориентация (Euler), p — положение COM, ω — угловая скорость базы, $\dot{p}$ — линейная скорость COM, r_i — вектор от COM до стопы i, F_i — GRF, $\hat{I}$ — оценка инерции, m — масса, g — гравитация.[1] Авторы подчёркивают, что ускорения являются более «универсальной валютой» возмущений и естественно обобщаются между роботами с разной массой/инерцией.[1] В схеме Fig. 2.1 это дополнительно поддержано блоком F/m, который нормирует силовые команды MPC в ускорения перед подачей в policy (robot-agnostic observation).[1]

Стоит отметить, что в baseline MPC постановка ноги задаётся эвристикой, зависящей от команды скорости и времени опоры:[1]

$$p_i^{\text{heur}} = p_i^{\text{hip}} + \frac{T_{\text{stance}}}{2}\, v + k\,(v - v_{\text{cmd}}) + \frac{1}{2}\sqrt{\frac{z_0}{\|g\|}}\,(v \times \omega_{\text{cmd}}), \tag{2.6}$$

где p_i^{hip} — положение бедра в world frame для ноги i, T_{stance} — время фазы опоры, v — скорость COM, v_{cmd} — команда скорости, z_0 — номинальная высота, ω_{cmd} — команда yaw-rate, а k — коэффициент (в статье $k = 0.03$).[1] Номинальная swing-траектория интерполируется Bezier-кривой и задаёт целевые положения стоп p_f.[1] Новизна RL-augmented MPC в том, что политика выдаёт Δq — смещение суставов *относительно номинальной swing-траектории*, тем самым *реактивно* меняя форму траектории и финальную постановку ноги при контакте с дискретными препятствиями (foot entrapment).[1] Важно, что тот же RL-модуль одновременно выдаёт и $\Delta\alpha, \Delta a$, т.е. синхронизирует swing-адаптацию с stance-силовым контролем, устраняя их «развязку», типичную для стандартного MPC пайплайна.[1]

2.5.2 Количественные результаты

Работа демонстрирует режимы, где baseline MPC деградирует, а гибридный контур сохраняет устойчивость:

- **поворот:** команды yaw-rate до ± 7 рад/с, с наблюдаемыми пиками порядка 8.5 рад/с;
- **бег:** команды скорости до 3.5 м/с, с пиковыми достигнутыми скоростями порядка 3 м/с;
- **бег+поворот:** при команде 2.5 м/с и yaw-rate 0.5 рад/с гибрид лучше удерживает режим, тогда как baseline вынужденно замедляется для предотвращения падения.

Также подчёркивается переносимость (в т.ч. по payload и между платформами).[1] Для данного отчёта важен вывод: *гибридный контур даёт физическую структуру и стабильность, позволяя RL-модулю оптимизировать «поверх»*

неё (в том числе потенциально энергию) без риска мгновенного развала локомоции.

2.6 Sim-to-real: доменная робастность и онлайн-адаптация (RMA как практический шаблон)

2.6.1 Идея и архитектура

Даже если политика локомоции демонстрирует высокое качество в симуляции, при переносе на реального робота она часто деградирует из-за *sim-to-real gap*: расхождений в трении, массе, силе приводов, контакте и локальном профиле рельефа. Rapid Motor Adaptation (RMA) предлагает практический шаблон: разделить управление на (i) **быстрый моторный контроллер** (base policy) и (ii) **медленный модуль адаптации**, оценивающий низкоразмерный латентный контекст среды/робота.[4] Ключевое допущение состоит в том, что параметры среды и «экстинсики» робота меняются существенно *медленнее*, чем состояние системы (позы/скорости), поэтому их можно оценивать асинхронно.[4]

Архитектура RMA приведена на Fig. 2.2.

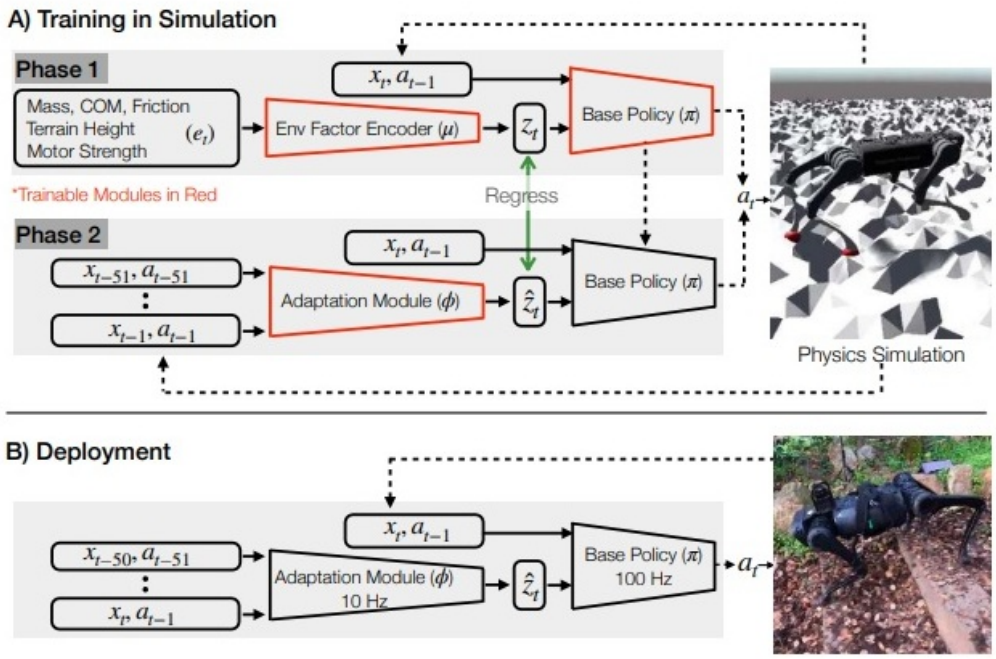

Рис. 2.2: Архитектура RMA[4]

Рассмотрим каждый блок данной архитектуры:

1. **Факторы среды** В симуляции доступен вектор факторов среды/робота e_t включающий, например, массу и положение COM, трение, локальную высоту рельефа и масштаб сил приводов.[4] В статье e_t имеет размерность 17: масса и её положение (3), motor strength (12), friction (1), local terrain height (1).[4]

2. **Кодировка факторов** Факторы e_t кодируются в латентный контекст (extrinsics) z_t:

$$z_t = \mu(e_t). \tag{2.7}$$

   Этот контекст затем используется base policy.[4]

3. **Моторный контроллер на базовой policy** π Base policy генерирует действие на высокой частоте, используя текущее состояние x_t, предыдущее действие a_{t-1} и контекст z_t:

$$a_t = \pi(x_t, a_{t-1}, z_t). \tag{2.8}$$

   На этапе *первой фазы* в симуляции обучаются μ и π (model-free RL), причём z_t формируется из privileged факторов.[4]

4. **Модуль адаптации** После фиксации base policy, на *Phase 2* обучается адаптационный модуль ϕ, который по короткому окну истории состояний и действий регрессирует оценку контекста:

$$\hat{z}_t = \phi\Big(\{x_{t-k}, a_{t-k}\}_{k=1}^{H}\Big), \tag{2.9}$$

минимизируя ошибку предсказания относительно z_t из Phase 1. Тем самым Phase 2 сводится к supervised learning «на траекториях», а веса π не меняются.

5. **Переход на железо** На реальном роботе истинный e_t недоступен, поэтому base policy получает $\hat{z}_t$:

$$a_t = \pi(x_t, a_{t-1}, \hat{z}_t). \tag{2.10}$$

При этом ϕ работает на более низкой частоте (в статье $\approx$ 10 Hz), а π генерирует команды на более высокой частоте (в статье $\approx$ 100 Hz) . Такая асинхронность снижает требования к on-board вычислениям.[4]

Отдельно важно отметить, что интерфейс управления выбран под переносимость: состояние x_t имеет размерность 30 и включает суставные углы (12), суставные скорости (12), roll/pitch торса (2) и бинарные контакты (4). Действие — целевые углы 12 суставов $a_t = \hat{q}_t \in \mathbb{R}^{12}$, реализуемые через PD-контур:

$$\tau = K_p(\hat{q} - q) + K_d(\hat{\dot{q}} - \dot{q}), \qquad \hat{\dot{q}} = 0, \tag{2.11}$$

где коэффициенты фиксированы (в статье $K_p = 55$, $K_d = 0.8$).

2.6.2 Результаты

В симуляции RMA сравнивается с baseline'ами по набору метрик (Success, TTF, Reward, Distance, Torque, Smoothness, Ground Impact). RMA достигает **Success 73.5%** при **TTF 0.85**, превосходя Robust (domain randomization без контекста) с **Success 62.4%** и приближаясь к верхней границе Expert (privileged z_t) с **Success 76.2%**.[4]

В реальных испытаниях RMA демонстрирует высокую устойчивость на сложных покрытиях: **memory foam mattress** и **slightly uneven foam** с **100%**

success, а также **step down 15 cm** с **80% success**. Также рассматривается сценарий со скользким участком (oily patch): reported **90% success**, причём компоненты предсказанного контекста $\hat{z}$ изменяются при входе на скользкую область, что интерпретируется как обнаружение/оценка смены условий контакта модулем адаптации.

Ключевой вывод работы: вынесение адаптации в отдельный модуль ϕ и асинхронный inference контекста позволяют уменьшить консерватизм чистой робастификации (domain randomization) и сохранить качество поведения при переносе.

2.7 Иерархические системы и long-horizon: от HRL к LLM-планированию

2.7.1 Постановка задачи

Long-horizon задачи для квадропеда требуют не только устойчивой краткосрочной локомоции, но и *планирования последовательности* действий: комбинировать локомоцию и манипуляцию, выбирать порядок подзадач, корректировать план после ошибок исполнения. В[3] подчёркивается, что такие задачи плохо решаются «плоской» политикой и требуют разделения на reasoning layer и библиотеку низкоуровневых навыков.

2.7.2 Архитектура системы

В работе[3] предлагается следующая иерархия из двух уровней:

1. **High-level reasoning layer (LLM-based).** High-level получает описание задачи и наблюдения сцены, после чего строит *hybrid discrete-continuous plan* в виде исполняемого кода, который оркестрирует навыки. Reasoning layer реализован как композиция LLM-агентов:[3]

- **Semantic planner (S)**: строит структуру плана (включая ветвления);
- **Parameter calculator (P)**: вычисляет параметры для вызовов навыков;
- **Code generator (C)**: генерирует исполняемый код для робота;
- **Replanner (R)**: выполняет перепланирование при ошибках выполнения.

Для компактной записи можно представлять этот слой как построение программы $\mathcal{P}$:

$$\mathcal{P} \;=\; C\big(S(\text{task}),\, P(\text{task}, \text{scene})\big), \qquad \mathcal{P} \leftarrow R(\mathcal{P}, \text{failures}), \tag{2.12}$$

где $\mathcal{P}$ — исполняемый "robot code", который вызывает low-level навыки.

2. **Low-level skill layer (RL-based).** На нижнем уровне используется набор специализированных навыков (локомоция и манипуляция), обученных RL. High-level не генерирует непрерывное управление напрямую, а вызывает навыки и задаёт их параметры через $\mathcal{P}$.

2.7.3 Метрики и результаты

В данной работе оценка была проведена по двум метрикам :

- **Success** — доля траекторий, в которых задача полностью выполнена.
- **NormDist** — нормированная дистанция до цели (задача-специфическая), нормированная начальной дистанцией:

$$\text{NormDist} \;=\; \frac{d(\text{current}, \text{goal})}{d(\text{init}, \text{goal})}. \tag{2.13}$$

В работе приводится абляция reasoning layer на четырёх задачах (100 траекторий, 3 seed). Полная система **LLM(S+P+C+R)** достигает:

- Light switching: Success 0.877 ± 0.015, NormDist 0.024 ± 0.003;
- Package delivery: Success 0.823 ± 0.035, NormDist 0.204 ± 0.049;

- Bridge building: Success 0.770 ± 0.052, NormDist 0.100 ± 0.021;
- Elevator ride: Success 0.580 ± 0.020, NormDist 0.136 ± 0.008.

Из численных результатов данной работы (взятых из таблицы 1 в статье) baseline **Hierarchical RL** показывает **Success** 0.0 ± 0.0 на всех четырёх задачах. Это прямой аргумент, что long-horizon требует более выразимого механизма планирования, чем наивный HRL-выбор навыков. Также абляции показывают, что удаление replanning ухудшает метрики, т.е. closed-loop перепланирование является значимым компонентом общей устойчивости.

3 Открытая проблема и гипотеза

Представленный обзор показывает, что современные полходы для локомоции на базе RL уже включают сильные и практически проверенные компоненты: (i) гибридные контуры, где MPC обеспечивает ограничения и устойчивость, а RL расширяет режимы и компенсирует модельные ошибки;[1] (ii) sim-to-real шаблоны с асинхронной онлайн-адаптацией латентного контекста (RMA), позволяющие сохранять качество при переносе без тяжёлой онлайн-идентификации;[4] (iii) иерархические системы для long-horizon задач, где высокоуровневое планирование (вплоть до LLM) управляет библиотекой низкоуровневых навыков;[3] (iv) подходы к энергоэффективности, показывающие измеримое снижение CoT при добавлении физически-информированных термов/структур.[2],[5]

Однако эти направления в основном решают *разные* части задачи и не дают единого "принципа сборки" для случая, когда требуется одновременно:

1. **long-horizon энергооптимальность** то есть минимизация CoT/мощности/потерь по траектории на десятках секунд и более;
2. **short-horizon устойчивость и ограничения** реализуемые через MPC на миллисекундных масштабах;
3. **переносимость на реальный робот** без деградации оптимальности (через адаптацию контекста или робастификацию).

3.1 Открытая проблема: согласование целей на разных горизонтах в иерархии planner → policy → MPC

В реальных системах "длинный горизонт" и "короткий горизонт" почти неизбежно разделяются архитектурно. Примером является RL-augmented MPC, где

политика (33 Hz) влияет на целевую динамику, а MPC/low-level контур (1 kHz) обеспечивает реализацию и устойчивость.[1] С другой стороны, long-horizon постановки требуют планировщика, который выбирает последовательность навыков и их параметры.[3] Наконец, перенос в реальность выигрывает от вынесения "экстринсиков" в отдельный адаптационный модуль, работающий асинхронно.[4]

Возникает системная проблема: **нет общепринятого способа согласовать энергооптимальность на длинном горизонте с гарантией устойчивости на коротком горизонте** так, чтобы энергокритерий оставался *валидным после замыкания MPC* и переносился при sim-to-real. На практике это проявляется в трёх типовых сбоях:

1. **Конфликт наград между уровнями** Верхний уровень (планировщик) оптимизирует энергию и/или скорость достижения цели, но нижний уровень оптимизирует устойчивость и ограничения. Если энергетический критерий сформулирован в терминах действий высокоуровневой политики (например, штраф на a_t), то после MPC-фильтра фактические торки/GRF и, следовательно, реальная энергия могут существенно отличаться от того, что оптимизировал high-level.

2. **"Поглощение" энергии корректировками MPC.** Даже если high-level предлагает "энергетически выгодные" команды, MPC может регулярно вносить corrective GRF/торки для компенсации контакта и неопределённостей модели (ситуация особенно вероятна при sim-to-real gap). В итоге наблюдаемая механическая мощность и CoT определяются не high-level решениями, а частыми стабилизирующими поправками низкого уровня.

3. **Непереносимость энергетического оптимума при смене условий.** Domain randomization часто делает походку консервативной (робастность ценой оптимальности), а без адаптации энергооптимальная политика может "ломаться" на реальном контакте. RMA показывает, что адаптация восстанавливает качество локомоции,[4] но в классической постановке адаптация оптимизируется под *успешность/робастность*, а не под *энергию* как целевую функцию.

Данная проблема имеет прежде всего прикладную важность. Энергоэффективность определяет автономность, что является особо важным критерием, например, в космической области, когда энергоресурсы сильно ограничены, тепловой режим приводов, а также запас устойчивости. При этом в современных научных работах энергоэффективность демонстрируется либо в относительно "плоских" постановках reward shaping,[2],[5] либо вне явной многоуровневой иерархии (planner → policy → MPC), либо без явной привязки адаптации результатов симуляции в реальный мир. Следовательно, вопрос согласования энергии и устойчивости в многоуровневом контуре остаётся недоопределённым.

3.2 Гипотеза

Гипотеза: в иерархических системах управления локомоцией вида

$$\text{planner} \rightarrow \text{policy} \rightarrow \text{MPC},$$

можно добиться устойчивой и переносимой энергоэффективности, если:

1. энергетический критерий оптимизируется **на том уровне, где он физически наблюдаем** (через фактические торки/скорости после MPC, т.е. через измеряемый power/CoT), а не только через штрафы на действия policy;

2. высщий уровень выбирает **энерго-параметризованные** цели для нижнего уровня (например, gait-параметры, допустимую "жёсткость"/агрессивность, целевую скорость, duty factor), а MPC решает задачу реализации с ограничениями;

3. sim-to-real обеспечивается **адаптацией контекста** (по шаблону RMA) как отдельным модулем, причём контекст используется для согласования модели/ограничений MPC и для корректной оценки энергии в reward.

Таким образом конечную гипотезу можно сформулировать следующим образом:

Отсутствует общепринятый способ согласовать long-horizon энергооптимальность с short-horizon устойчивостью в иерархии planner → policy → MPC так, чтобы (a) избежать reward conflict между уровнями, (b) не допустить "сведания" энергетического выигрыша стабилизирующими корректировками MPC, и (c) обеспечить переносимость решения в реальность через адаптацию контекста. Можно предположить, что это достигается при оптимизации энергии по ***закрытому контуру*** *(post-MPC power/CoT) и при вынесении неопределённостей в отдельный контекстный адаптер, влияющий как на low-level реализацию, так и на оценку энергетической цели.*

3.3 Новизна гипотезы

Данная гипотеза не вводит каких-либо новых алгоритмов обучения с подкреплением. Ее ценность заключается в **согласовании** уже сильных компонентов, разработанных другими учеными и исследователями, в единый критерий и протокол обучения: энергия должна оптимизироваться по фактическому post-MPC поведению, а переносимость обеспечиваться контекстной адаптацией, влияющей на весь контур.

4 Практическая часть: прототип проверки гипотезы на MuJoCo Walker2d

4.1 Постановка эксперимента и реализация

Практическая часть задумывалась как минимальная проверка идеи из гипотезы: в многоуровневом контуре управления локомоцией энергия оптимизируется на "длинном" горизонте, а устойчивость обеспечивается "короткогоризонтным" стабилизатором (например, MPC). В идеале это должен был быть пайплайн вида *planner на базе LLM→ policy → MPC* с разными частотами, корректной энергометрикой (мощность/CoT) и, желательно, адаптацией к доменным сдвигам. Из-за ограничений по времени полноценную систему такого уровня собрать не удалось.

Вместо этого был выполнен воспроизводимый прототип на стандартной задаче MuJoCo/Gymnasium `Walker2d-v4`, который позволяет увидеть, как на энергию влияет (i) reward shaping в PPO и (ii) низкоуровневая пост-обработка действий, имитирующая роль стабилизатора. Алгоритм обучения — **PPO**. Сравнивались три конфигурации:

(1) **PPO baseline** (`walker_base_clean`): обучение на стандартной награде среды без специальных энергетических штрафов.

(2) **PPO + energy shaping** (`walker_energy_clean`): обучение с модифицированной наградой, где добавлена энерго-регуляризация по действию и по высокочастотным изменениям управления:

$$r'_t \;=\; r_t \;-\; w_E \,\|a_t\|_2^2 \;-\; w_J \,\|a_t - a_{t-1}\|_2^2. \tag{4.1}$$

Здесь $\|a_t\|^2$ используется как прокси "усилия/энергии" (в Walker2d действие соответствует моментам/нормированным торкам), а $\|a_t - a_{t-1}\|^2$ штрафует "дерганое" управление.

(3) **PPO + energy shaping + MPC-like filter** (`walker_energy_clean_mpc`): та же обученная политика, но при инференсе действие проходит через простую низкоуровневую пост-обработку, которая уменьшает амплитуду управления при плохом балансе и добавляет демпфирование по суставным скоростям:

$$\alpha_t = \mathrm{clip}\Big(1 - \frac{|\theta_t|}{\theta_{\max}},\ \alpha_{\min},\ 1\Big), \tag{4.2}$$

$$a_t^{\mathrm{damp}} = a_t - k_v\,\dot{q}_t, \tag{4.3}$$

$$a_t^{\mathrm{filt}} = \mathrm{clip}\big(\alpha_t\, a_t^{\mathrm{damp}},\ a_{\min}, a_{\max}\big). \tag{4.4}$$

Это не настоящий MPC: скорее “короткогоризонтный стабилизатор”, который гарантированно сглаживает рискованные действия. Он введён именно как прокси того, что в реальной системе делает MPC/low-level контур: обеспечивая устойчивость, он *меняет* фактическое управление относительно того, что выдала policy.

Поскольку в этой постановке нет аккуратного доступа к физической мощности приводов и CoT в строгом смысле, использовалась согласованная с наградой прокси-метрика энергии. Энергия на эпизод определялась как

$$E_{\mathrm{ep}} \;=\; \sum_{t=0}^{T-1} \|a_t\|_2^2, \tag{4.5}$$

а “энергия на метр” (proxy) — как

$$E_{\mathrm{m}} \;=\; \frac{E_{\mathrm{ep}}}{\Delta x + \varepsilon}, \qquad \Delta x = x_T - x_0. \tag{4.6}$$

Дополнительно для интерпретации фиксировались длина эпизода и пройденная дистанция. Для каждой конфигурации было выполнено $N = 500$ эпизодов.

Визуализация действий агента была реализована на базе MuJoco MPC, который по сути является интерактивным MPC-демо/вьюер, который идёт в составе MuJoCo. Скриншот из данной среды представлен на изображении ниже:

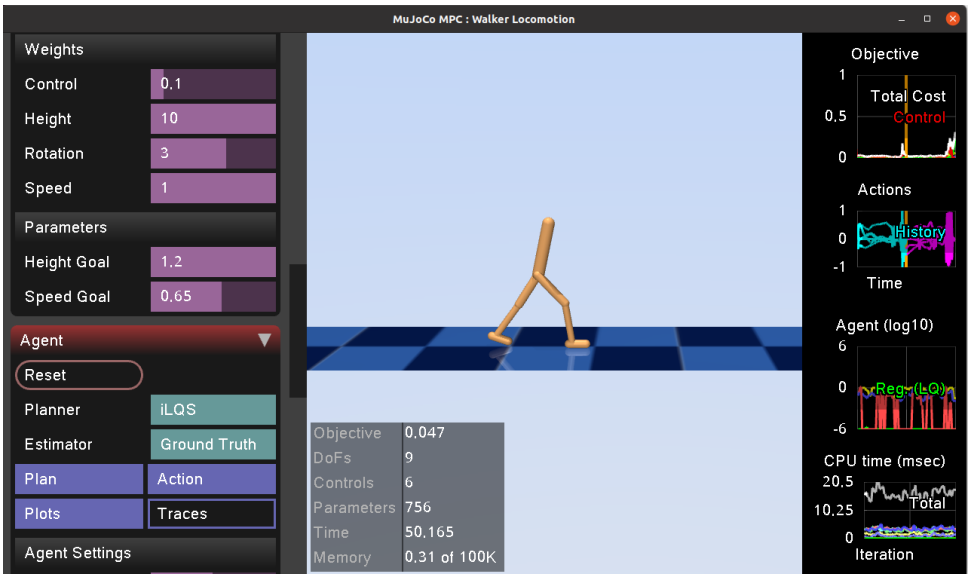

Рис. 4.1: Движение агента в среде MuJoco MPC

4.2 Результаты и связь с гипотезой

Главный результат показан на рисунке. 4.2: средняя метрика E_{m} (proxy, меньше — лучше) уменьшается при добавлении “стабилизатора” поверх policy. В числах (среднее $\pm$ std):

E_{m} : baseline 186.05±19.22 $\rightarrow$ + energy shaping 176.83±19.07 $\rightarrow$ + energy + MPC-filter 145.51±

То есть energy shaping даёт умеренное улучшение порядка $\approx 5\%$ относительно baseline, а пост-обработка действий “MPC-like” даёт существенно больший эффект: $\approx 22\%$ относительно baseline.

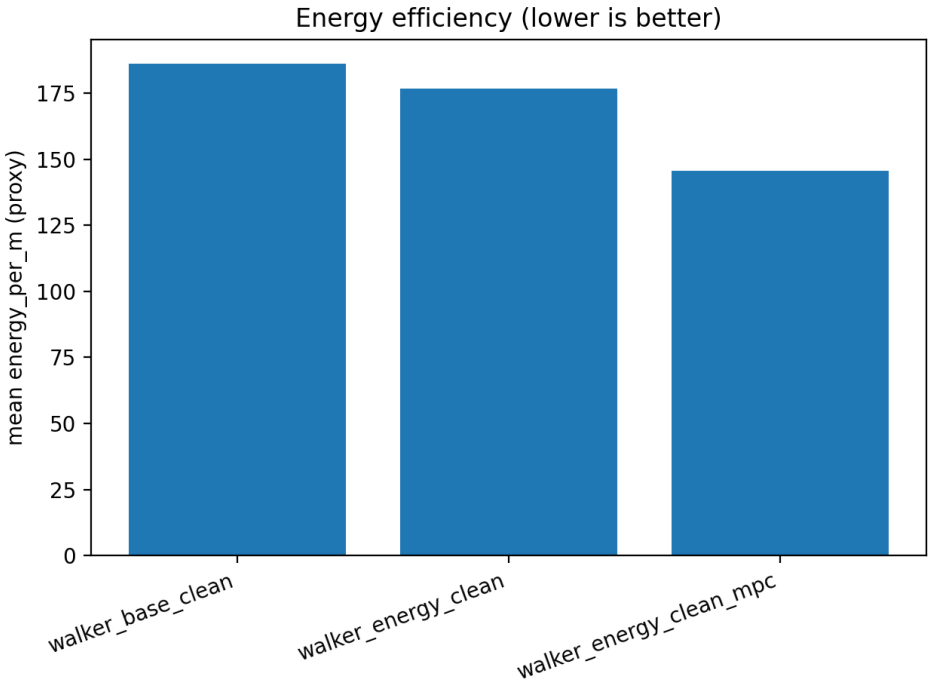

Рис. 4.2: Энергоэффективность для трех конфигураций

Рисунок. 4.3 дополняет картину: распределение E_{ep} по эпизодам показывает, что `PPO + energy shaping` может иметь даже более высокий E_{ep}, чем baseline, что не противоречит улучшению E_{m}: в среднем такие эпизоды и длиннее, и проходят большую дистанцию. Для `PPO + energy + MPC-filter` распределение E_{ep} заметно сдвинуто вниз, но это связано и с более короткими эпизодами. Поэтому в этой постановке ключевая метрика для сравнения — именно E_{m} как “энергия на пройденный путь”, а не “энергия на эпизод”.

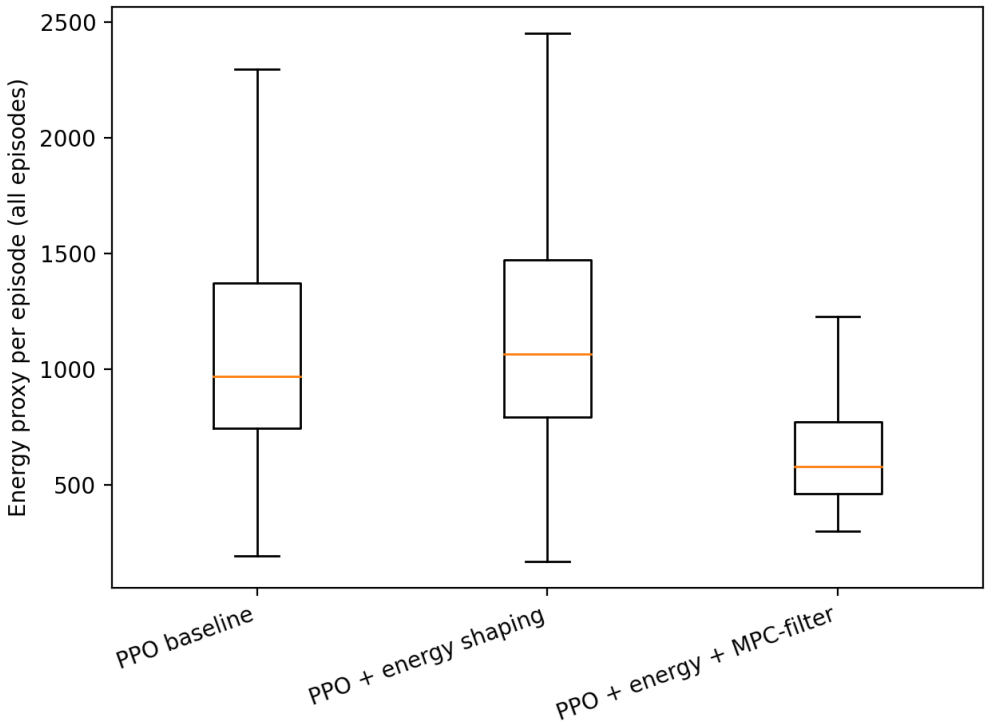

Рис. 4.3: Распределение E_{ep} по эпизодам для трех конфигураций

Эти результаты хорошо ложатся в общую логику отчёта и напрямую подсвечивают мотив гипотезы. Наивная попытка “сделать политику энергоэффективной” через reward shaping (аналогично идеям энерго-регуляризации в литературе) даёт ограниченный эффект. При этом добавление низкоуровневого стабилизатора, который *изменяет* действие policy, резко меняет энергетические метрики. Это и есть минимальная демонстрация того, почему в иерархии “длинный горизонт (энергия) $\rightarrow$ короткий горизонт (устойчивость)” возникает риск несогласованности: policy оптимизирует одно, но фактическое управление после “короткогоризонтного” слоя становится другим. В полноценной системе вместо пост-фильтра должен стоять MPC с физически согласованным интерфейсом (как в RL-augmented MPC[1]), а энергия должна считаться и оптимизироваться по фактическому пост-MPC поведению, иначе неизбежны reward mismatch и “съедание” энергетического выигрыша корректировками низкого уровня.

Итог практической части может быть сформулирован следующим образом: полноценную проверку исходной гипотезы (с настоящим MPC в контуре, корректной мощностью/CoT и модулем адаптации в стиле RMA) сделать не удалось,

но даже упрощённый прототип показывает, что “энерго-оптимальная policy” и “стабилизирующий низкий уровень” не складываются автоматически. Это подтверждает, что поставленная в гипотезе проблема (согласование энергии и устойчивости в многоуровневой архитектуре) действительно практическая и проявляется уже на простом стенде.

Список иллюстраций

Литература

[1] C. Yiyu and N. Quan, “Learning agile locomotion and adaptive behaviors via rl-augmented mpc,” *Research Gate*, 2024.

[2] C. Wang, A. Viswanathan, E. Sihite, and A. Ramezani, “Guiding energy-efficient locomotion through impact mitigation rewards,” 2025.

[3] O. Yutao, L. Jinhan, L. Yunfei, L. Zhongyu, Y. Chao, S. Koushil, and W. Yi, “Long-horizon locomotion and manipulation on a quadrupedal robot with large language models,” 2025.

[4] K. Ashish, F. Zipeng, P. Deepak, and M. Jitendra, “Rma: Rapid motor adaptation for legged robots,” 2021.

[5] L. Guanda, “Bioinspired deep reinforcement learning for energy-efficient control of redundant robotic systems,” *Tohoku University*, 2025.