Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Воронежский государственный технический университет»
(ФГБОУ ВО ВГТУ)

На правах рукописи

Кашко Василий Васильевич

АДАПТИВНАЯ СИСТЕМА УПРАВЛЕНИЯ ИНТЕЛЛЕКТУАЛЬНЫМ АГЕНТОМ НА ОСНОВЕ ГЛУБОКОГО ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Специальность 2.3.1. Системный анализ, управление и обработка информации, статистика

Диссертация
на соискание учёной степени
кандидата технических наук

Научный руководитель:
доктор технических наук, профессор
Олейникова Светлана Александровна

Воронеж-2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы. Одной из центральных проблем в современной теории управления, в особенности при работе с динамически изменяющимися неструктурированными средами в контексте киберфизических систем, является задача построения системы управления интеллектуальным агентом. Классические подходы, базирующиеся на точных моделях окружающих сред и предварительно определённых алгоритмах, зачастую оказываются недостаточно эффективными для управления агентом в сложных, динамически изменяющихся или заранее не полностью определённых средах эксплуатации. Применение глубокого обучения с подкреплением представляет собой альтернативу, обеспечивающую киберфизическую систему возможностью самостоятельно обучаться и реализовывать адаптацию к изменениям в среде посредством опыта, получаемого от взаимодействия с окружающей средой на основе метода проб и ошибок.

Проблематика управления интеллектуальным агентом входящем в состав киберфизической системы активно развивается с 1960-х готов. В настоящее время значимый вклад в развитие направления внесло применение глубокого обучения с подкреплением, что в значительной степени повысило автономность и адаптивность к динамически изменяющимся условиям. В частности, задаче обучения множеству стратегий (Multitask/Multi-skill) посвящены труды X. B. Peng, P. Abbeel, S. Levine и др. Обеспечением устойчивости и адаптации (Robust control) занимались J. Hwangbo, M. Hutter, G. Marcius, L. Righetti и др. В работах J. Peters, M. Hutter, C. Finn, S. Laura, S. Levine рассматривается возможность создания универсальных стратегий и переноса обученного в симуляции агента на реальную систему. Среди отечественных учёных, внёсших значимый вклад в изучение управления киберфизическими системами, в частности на основе алгоритмов обучения с подкреплением, являются А.И. Панов, Р.В. Мещеряков, А.П.

Преображенский и др.

Однако не все проблемы в данной области решены. В частности, среди проблем обучения выделяются забывание при добавлении новых задач, конфликт градиентов между навыками, ухудшение качества при масштабировании и сложность балансировки между общими и специфичными представлениями. Кроме того, при переносе агента из симуляции на реальные киберфизические системы возникает сложность с обеспечением одновременного решения базовых задач, нестабильность обучения из-за роста, отсутствие масштабируемых архитектур для универсальных политик, высокие трудозатраты валидации на реальных системах.

В результате, существующие подходы либо демонстрируют ограниченную обобщающую способность, либо страдают от деградации при переносе обученного агента из симуляции. Таким образом, проведённое исследование отвечает современным вызовам в области построения адаптивных самообучающихся систем управления и вносит вклад в решение актуальной научной проблемы построения системы управления интеллектуальным агентом на базе глубокого обучения с подкреплением.

Целью работы является разработка адаптивной системы управления интеллектуальным агентом на основе обучения с подкреплением, обеспечивающей обучение множеству стратегий с минимизацией коллизий на опыте, полученном при непосредственном взаимодействии киберфизической системы с окружающей средой и автономную адаптацию под изменения среды на примере решения задачи управления локомоцией шагающего робота.

Задачи исследования. Достижение поставленной цели диссертационного исследования требует решения следующих задач:

1. Произвести анализ существующих подходов в области управления интеллектуальным агентом на основе глубокого обучения с подкреплением, включая проблематику управления локомоцией шагающих

роботов как частного случая киберфизических систем с целью определения наилучшего алгоритма обучения с подкреплением.

2. Разработать математическое обеспечение системы управления интеллектуальным агентом, обеспечивающее повышение уровня безопасности и устойчивости функционирования управляемой киберфизической системы.

3. Разработать специальное алгоритмическое обеспечение системы управления интеллектуальным агентом, обеспечивающее ускорение обучения и переноса агента на реальную систему за счёт увеличения времени устойчивого функционирования и сокращения числа нештатных режимов (коллизий).

4. Разработать архитектуру глубокой нейронной сети, обеспечивающей агента, основанного на глубоком обучении с подкреплением, возможностью обучения нескольким стратегиям в контексте одной глубокой нейронной сети и ликвидирующей эффект «забывания» и потерю деталей каждой из стратегий.

5. Разработать структуру системы управления интеллектуальным агентом, обеспечивающую автономное адаптивное безопасное обучение множеству стратегий с последующим переносом на реальную киберфизическую систему и безопасной настройкой на примере формирования локомоции шагающего робота.

6. Произвести экспериментальное исследование эффективности разработанного математического и алгоритмического обеспечения, архитектуры многофункционального агента и структуры системы управления.

Объект исследования: система управления интеллектуальным агентом, основанным на глубоком обучении с подкреплением, входящим в состав киберфизической системы.

Предмет исследования: математическое и алгоритмическое

обеспечение системы управления интеллектуальным агентом на примере генерации шагающей локомоции киберфизической системы шагающего робота, включая формализацию задачи управления, методы обучения с подкреплением, архитектурные решения по построению нейросетевого агента и алгоритмы формирования и использования множества стратегий в условиях непосредственного взаимодействия с окружающей средой.

Методы исследования. В процессе решения задач диссертационного исследования использовались методы: системного анализа, принятия решений, теории обучения с подкреплением, математического моделирования, теории автоматического управления, структурного анализа, теории глубокого обучения, теории Марковских процессов принятия решений, динамического программирования, теории вероятностей и математической статистики, методов оптимизации, имитационного моделирования, объектно-ориентированного программирования и проектирования.

Тема диссертационного исследования соответствует следующим пунктам паспорта специальности 2.3.1 «Системный анализ, управление и обработка информации, статистика»: п.5 «Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта»; п.9 «Разработка проблемно-ориентированных систем управления, принятия решений и оптимизации технических объектов»; п.10 «Методы и алгоритмы интеллектуальной поддержки при принятии управленческих решений в технических системах».

Научная новизна работы. В контексте диссертационного исследования были получены следующие результаты, характеризующиеся научной новизной:

1. Математическое обеспечение системы управления интеллектуальным агентом, отличающееся внедрением в агента возможности

предсказания риска возникновения нестабильных состояний, основанной на выделенном множестве критических состояний, и обеспечивающее повышение безопасности обучения и стабилизацию функционирования управляемой киберфизической системы.

2. Специальное алгоритмическое обеспечение системы управления интеллектуальным агентом, отличающееся фокусированием обучения на границе устойчивости, обеспечивающее повышение безопасности обучения и переноса агента с последующей безопасной настройкой в реальной среде.

3. Архитектура глубокой нейронной сети агента, отличающаяся возможностью аппроксимации множества стратегий в рамках одной полносвязной нейронной сети, и обеспечивающая реализацию многофункционального управления с ликвидацией эффекта «забывания» и потери деталей каждой из стратегий.

4. Структура системы управления, базирующаяся на нейросетевом кластере с переключающими нейронами, обеспечивающая автономное адаптивное обучение множеству стратегий с последующей безопасной настройкой в реальной среде.

Практическая значимость исследования заключается в создании и экспериментальной апробации проблемно-ориентированной системы управления интеллектуальным агентом, отвечающим за формирование шагающей локомоции киберфизической системы шагающего робота, реализующей разработанные алгоритмы и архитектурные решения. Разработанная система обеспечивает адаптивное управление движением в условиях неопределенности внешней среды. Предложенные решения целесообразно использовать при разработке и управлении робототехническими системами, предназначенными для функционирования в изменяющихся условиях, а также в образовательных и исследовательских комплексах.

Положения, выносимые на защиту

1. Математическое обеспечение системы управления интеллектуальным агентом обеспечивает повышение безопасности обучения и стабилизацию функционирования киберфизической системы за счет внедрения в агента возможности предсказания риска возникновения нестабильных состояний, основанной на выделенном множестве критических состояний.

2. Алгоритмическое обеспечение системы управления интеллектуальным агентом позволяет повысить безопасность обучения и переноса агента с последующей безопасной настройкой в реальной среде.

3. Архитектура глубокой нейронной сети агента решает проблему низкой репрезентативной способности сети и обеспечивает реализацию многофункционального управления с ликвидацией эффекта «забывания» и потери деталей каждой из стратегий за счет их аппроксимации в рамках одной полносвязной нейронной сети.

4. Структура системы управления обеспечивает автономное адаптивное безопасное обучение множеству стратегий с последующей безопасной настройкой в реальной среде за счет использования нейросетевого кластера с переключающими нейронами и учёта критических состояний.

Результаты внедрения. Разработанные в диссертационном исследовании схемы и механизмы были внедрены в образовательный процесс ФГБОУ ВО «Воронежский государственный технический университет» в виде лабораторного практикума и лекционного курса по дисциплинам "Интеллектуальные системы", "Системы искусственного интеллекта". Кроме того, алгоритмическое и программное обеспечение для построения адаптивных систем управления интеллектуальными агентами, основанного на методах обучения с подкреплением внедрены в деятельность компании ООО «Девелоперс». Эффект от внедрения заключается в

повышении точности принятия решений в изменяющихся условиях и сокращении времени на ручную настройку параметров алгоритмов управления за счёт автоматического обучения.

Апробация работы. Рассмотрение и обсуждение основных положений диссертационного исследования проводились в контексте следующих научно-практических конференций: VI Всероссийской научно-практической конференции «Современные информационные технологии. Теория и практика» (Череповец, 2023), XV Международная интернет-конференция молодых учёных, аспирантов и студентов «Инновационные технологии: теория, инструменты, практика» (Пермь, 2023), Международного форума профессионального образования «Антропоцентрические науки в образовании: вызовы, трансформации, ресурсы» (Воронеж, 2024), Международной научной конференции «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 2024), Международной научно-практической конференции «Управление программным инжинирингом» (Воронеж, 2025, 2026), Международной научно-практической конференции «Интеллектуальные информационные системы» (Воронеж, 2025), Международной научной конференции «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 2025, 2026), XVII Международная интернет-конференция молодых учёных, аспирантов и студентов «Инновационные технологии: теория, инструменты, практика» (Пермь, 2025), II Научно-практическая конференция «Интеллектуальные технологии цифровой инженерии» (Воронеж, 2026).

Публикации. В результате выполнения диссертационного исследования было опубликовано 19 научных публикаций, в том числе 6 – в изданиях, рекомендованных ВАК РФ, 1 свидетельство о регистрации программы для ЭВМ. В опубликованных в соавторстве работах, перечень которых приведён в конце автореферата, автором персонально был получен следующий перечень результатов: [24, 26, 35] – теоретический анализ

существующих подходов и алгоритмов обучения с подкреплением, применяемых для реализации управления движением шагающих роботов, [31, 38, 109] – математическое обеспечение системы управления локомоцией шагающего робота, отличающееся применением выделенного множества критических состояний и обеспечивающее безопасное обучение, и последующий безопасный перенос обученного агента на реальную систему; [23, 25, 27, 29, 30, 37, 39] – экспериментальное исследование специфики существующих и эффективности применения разработанных алгоритмов управления локомацией шагающего робота; [32, 33, 36] – алгоритмическое обеспечение управления локомоцией шагающего робота; [28] – построение архитектуры нейронной сети, обеспечивающей агента, основанного на глубоком обучении с подкреплением, многофункциональностью в контексте одной полносвязной нейронной сети; [34] – построение многофункционального агента на базе полносвязной сети-кластера с использованием переключающих нейронов.

Структура и объем работы. В состав диссертационной работы входят: введение, четыре главы, заключение, список литературы, состоящий из 167 наименований. Изложение основной части представлено на 167 страницах, с использованием 20 рисунков и 4 таблиц.

ГЛАВА 1. ОСОБЕННОСТИ СОВРЕМЕННОГО СОСТОЯНИЯ ПРОБЛЕМЫ УПРАВЛЕНИЯ ЛОКОМОЦИЕЙ МОБИЛЬНОГО ШАГАЮЩЕГО РОБОТА КАК ЧАСТНОГО СЛУЧАЯ КИБЕРФИЗИЧЕСКОЙ СИСТЕМЫ

В данной главе производится анализ литературных источников, посвящённых существующим подходам к формированию управления локомоцией шагающих мобильных роботов как частного случая киберфизических систем, обладающих конечностями шарнирного типа. Акцент делается на рассмотрении открытых проблем, связанных с применением обучения с подкреплением к управлению реальными системами. Основные проблемы управления движением мобильных шагающих роботов рассматриваются в первой части. Во второй части главы анализируются существующие подходы к решению задачи управления движением шагающих роботов. Алгоритмы глубокого обучения с подкреплением как основа обучения интеллектуальных агентов проанализированы в третьей части. В четвертой части главы исследуются особенности реализации многофункционального управления мобильным шагающим роботом. Проблемы, связанные с переносом обученного в симуляции агента на реального робота приведены в пятой части главы. В заключение главы производится постановка целей и задач, а также формируется дизайн исследования.

1.1. Основные проблемы управления движением мобильных шагающих роботов

В настоящее время развитие технологий позволило осуществить переход от традиционных автоматизированных систем, представляющих собой оборудование, управляемое встроенным компьютером, к

киберфизическим системам [1 – 7, 42, 43, 47, 53], основанным на непрерывном взаимодействии вычислительных и физических компонентов. В общем случае, киберфизические системы основаны на иерархической структуре, состоящей из трёх основных уровней: физического – представляющего реальный объект физического мира, вычислительно-управляющего – реализующего обработку сенсорной информации, вычисления и управление, и уровня коммуникационной инфраструктуры, обеспечивающей непосредственное взаимодействие между двумя предыдущими уровнями. Киберфизические системы реализуют непрерывный цикл адаптации к изменениям физической среды путём непрерывного взаимодействия с ней. Посредством датчиков производится определение текущего состояния, на основании которого производится принятие решения и генерация управляющего воздействия, реализуемого исполнительными механизмами.

Частным случаем киберфизической системы является робот [12, 13, 15, 20, 45, 86, 116]. Помимо автоматизации, тесного взаимодействия физической платформы с программным обеспечением и адаптацией на основе показаний сенсоров и непосредственного взаимодействия с физической средой, для роботов характерны специфические черты. К ним относятся: наличие исполнительных механизмов, автономность и мобильность, целеустремлённость – направленность на решение прикладных задач. В мобильной робототехнике по способу перемещения выделяются следующие категории роботизированных платформ: **колёсные, гусеничные, шагающие и гибридные** [70, 110]. Использование колёсного шасси характеризуется простотой управления и энергоэффективностью, но, в то же время, предъявляет особые требования к рельефу, что накладывает ограничения на применение к сложным типам местности [74, 109]. Гусеничные платформы обладают большей проходимостью и манёвренностью, энергоэффективностью, высокой грузоподъёмностью и тяговой

способностью, но очень громоздки и сложны с точки зрения ремонта и эксплуатации [48, 109]. Наибольший интерес представляет шагающая локомоторная система, эволюционно апробированная и являющаяся универсальной по отношению к различным формам и типам рельефа, способная преодолевать препятствия, соизмеримые с собственными габаритами [74, 109]. Шагающая платформа обеспечивает максимальную гибкость без необходимости в постоянной траектории перемещения, что позволяет осуществлять движение по самым сложным типам местности, обладающим резкими перепадами высот, неровностями поверхности, ямами, наклонами и различными препятствиями [74]. При всех имеющихся преимуществах, с точки зрения универсальности и проходимости, её применение в робототехнике сопряжено с большой сложностью в управлении. Поскольку, каждая конечность шагающего механизма состоит из множества независимых шарниров, возникает проблема непрерывного управления в больших масштабах [74]. Конечности шагающей платформы представимы в единообразной манере посредствам системы рычагов, приводимых в движение специализированным аппаратом [10, 51]. Последний, в робототехнике, может быть реализован при помощи поступательных (гидравлических или пневматических приводов) или вращательных движителей (сервопривод) [70]. Для любого из упомянутых случаев, управление конечностью сводится к изменению величин углов её сочленений в дискретные моменты времени [21, 30, 108].

С другой стороны, существует великое множество форм и разновидностей рельефа местности. Для обеспечения корректного перемещения шагающей платформы необходимо брать во внимание каждый конкретный случай, что подразумевает вариативность управляющих стратегий.

В робототехнике распространены двуногая, четвероногая, шестиногая (гексапод) и восьминогая (октопод) конфигурации. Двуногий робот обладает

максимальной подвижностью и манёвренностью в процессе передвижения при наименьшей устойчивости конструкции, что требует построения сложной системы управления для поддержания равновесия. Робот октопод имеет преимущество перед двуногими, четвероногими и шестиногими роботами с точки зрения его способности преодолевать широкий спектр разновидностей рельефа местности, с сохранением статической устойчивости, что исключает требование к использованию сложной динамической балансировки в структуре управления [117]. Особенности конфигурации накладывают определённые условия при формировании контроллера шагающей платформы. Зачастую приходится учитывать частные характеристики механизма, которые оказывают прямое воздействие на качество и оптимальность его функционирования. Наличие индивидуальных особенностей является основным препятствием, порождающим вариативность подходов и отсутствие чёткого обобщения при построении управляющих систем. Стоит отметить, что чем большим числом конечностей обладает шагающая платформа, тем выше вероятность статической устойчивости в процессе передвижения и тем сложнее управление [117].

В результате, с учётом распространённости в природе, гибкости и высокой проходимости шагающих конфигураций, формирование управления шагающим механизмом, обеспечивающего стабильное передвижение в пространстве и обладающего вариативностью стратегий, в соответствии с видом и формой рельефа, является актуальной задачей. Вариативность поведения киберфизической системы обеспечивается вычислительно-управляющим уровнем, а именно структурой системы управления, подходом к её построению и применяемыми управляющими алгоритмами. Следует отметить, что ещё одним открытым на сегодняшний день вопросом является построение математического обеспечения управления шагающим роботом, которое является универсальным для различных типов конфигурации

шагающих систем. Поскольку робототехнические комплексы представляют собой дорогостоящее оборудование, актуальным является построение специального математического и алгоритмического обеспечения, способствующего формированию «безопасного» управления, при котором значительно снижен или полностью исключён риск выхода из строя всей шагающей платформы или её подвижных узлов, а так же коллизий в процессе функционирования.

Проблематика многофункциональности [74, 89, 96, 99, 101, 136, 137, 147, 151] заключается в поиске способов построения архитектуры управляющего агента, способной аппроксимировать множество различных стратегий с удобным механизмом их селекции в зависимости от условий окружающей среды. С ней тесно связана проблема низкой репрезентативной способности глубокой нейронной сети и эффект «забывания». В большинстве работ, связанных с проблематикой многофункциональности агента, упоминается факт неспособности сети запомнить более одной стратегии, но соответствующая проблема не решается «напрямую», а предлагаются иные способы формирования многофункциональности.

Последнее время наблюдается активная тенденция в применении обучения с подкреплением, а именно его глубокой разновидности, при построении систем управления роботами. По своей природе, обучение с подкреплением представляет собой метод оптимизации, производящий максимизацию целевой функции, базирующейся на получаемых в процессе взаимодействия со средой сигналах вознаграждения. Каждому алгоритму глубокого обучения с подкреплением, обеспечивающему адаптивность системы управления, на начальном этапе настройки стратегии свойственно генерировать действия, вызывающие коллизии. Для случая обучения с использованием имитационной среды, данный эффект не является критичным, в отличие от управления реальной физической системой. В имитационных средах возникновение коллизий приводит к замедлению

обучения, тогда как для управления реальной системой, зачастую требуется наличие третьей стороны производящей восстановление дальнейшего функционирования и велик риск выхода из строя дорогостоящего оборудования. В настоящее время, распространён подход предварительной настройки агента в имитационной среде с последующим переносом на реальную систему [77, 84, 92, 99, 101, 128, 130, 139, 140, 145, 154]. После полноценного обучения и реализации переноса велика вероятность ошибок принятия решений, в результате которых могут возникнуть коллизии. Несмотря на хорошие результаты, демонстрируемые применением предварительной симуляции, всё ещё остаётся актуальным поиск подхода, способного обеспечить стабильное и безопасное обучение и функционирование агента на реальных системах.

Следующей важной проблемой, тесно связанной с обеспечением безопасного обучения и функционирования, является проблема переноса из симуляции на реальную систему (Sim to real) [92, 128, 130, 139, 145]. Несмотря на то, что агент может быть хорошо обучен в имитационной среде, при переносе на реальную систему зачастую наблюдается значительно снижение качества реализации стратегии. Это связано с неполнотой имитационных моделей и отсутствием учёта полного спектра параметров окружающей среды. В настоящее время существуют различные подходы, предназначенные для повышения качества моделирования, которые показывают хорошие результаты на практике. Несмотря на это остаётся риск возникновения коллизий, поскольку каждый из предложенных подходов не гарантирует «безопасное» функционирование.

Для решения проблемы низкой скорости обучения, упомянутой ранее, предпринимались попытки использования «продвинутых» алгоритмов, использующих асинхронное обновление [85], мягкое обучение, с учётом энтропии [149] и подход с применением детерминированной стратегии [128, 150]. Они позволили незначительно ускорить обучение, но, не смотря на это,

проблема является актуальной, поскольку задача нахождения баланса между отсутствием смещения и выборочной эффективностью, по причине которой управляющему агенту требуется большое количество выборок, остаётся не решённой [69, 92, 99, 140].

Важной проблемой применения глубокого ОП к реальным роботам является точность настройки функции вознаграждения [92]. Её качество напрямую влияет на функционирование алгоритма и его сходимость [24, 56, 69]. Поэтому формирование чёткой методологии её построения является актуальной задачей [92, 128, 139, 154]. Поскольку сигнал награды выступает в качестве указателя цели для агента, не следует допускать её *разреженности*, так как это, в лучшем случае, замедлит обучение, а в худшем, обеспечит расхождение алгоритма [56, 69]. В основном, в контексте прикладных задач робототехники применяются параметрические функции, где в роли параметров выступают значения определённых характеристик, влияющих на оптимальность выполнения поставленной перед агентом задачи [24, 69, 90, 109, 111, 128]. Ещё одной проблемой является энергоёмкость. При некачественном сигнале вознаграждения, агент может выбирать «дорогостоящие» действия, способные значительно уменьшить время работы робота. Для решения проблемы точной настройки функции награды был предложен подход обратного обучения с подкреплением (Inverse Reinforcement Learning, IRL), идея которого заключается в определении минимизируемой функции вознаграждения на основе эталонных траекторий агента, полученных при различных обстоятельствах [88, 126, 163]. Применение дополнительного шага снижает скорость обучения. Для ликвидации данного недостатка предлагается преобразовать задачу в имитацию поведения оригинала и заменить шаг восстановления функции вознаграждения на оптимизацию политики на основе эталонных стратегий [156]. Или, в качестве альтернативы, для ускорения обучения, применить метод управляемого поиска оптимальной стратегии, позволяющего избежать

«плохих» локальных оптимумов за счёт управления процессом изучения стратегии [113]. Несмотря на результаты применения обратного обучения с подкреплением, актуальным является поиск метода построения функции вознаграждения без использования эталонных траекторий и дополнительных шагов.

1.2. Существующие подходы к решению задачи управления движением шагающих роботов

Рассмотрим подходы к управлению локомоцией шагающего робота в разрезе данных проблем более подробно. В процессе анализа литературных источников были выявлены следующие основные подходы:

1. Подход, основанный на математической модели (классический) [9, 11, 19, 54, 55, 58, 60, 62, 65, 68, 78, 80, 98, 102, 103, 105, 112, 117, 119, 121, 125, 138, 146, 153, 158, 161, 164];
2. Подход, основанный на имитации (копирование поведения оригинала) [73, 79, 106, 107, 120, 124, 129, 133];
3. Биоинспирированный подход (центральный генератор паттернов) [75, 81, 93, 100, 165];
4. Применение адаптивного управления на базе алгоритмов обучения с подкреплением [39, 50, 59, 74, 76, 77, 83 – 85, 90, 92, 97, 99, 101, 109, 111, 114, 127, 128, 131, 134, 135, 139, 140, 147 – 150, 154, 157, 166].

Главной проблемой, влияющей на построение контроллера локомоции шагающей системы, является сложность управления и координации двигательных действий в динамически изменяющейся и не предсказуемой среде [128]. Организация передвижения по неровному рельефу, как обладающему структурой, так и не структурированному, представляет сложную задачу. В таких условиях, реализация автономной работы контроллера требует сложного многоконтактного планирования и

управления им [157]. Контроль конечностей является сложной проблемой, требующей в каждом цикле походки правильного выбора траектории для реализуемого локомоторного паттерна.

Методы управления, основанные на моделях (традиционные) демонстрируют хорошие показатели манёвренности шагающих роботов, но предназначены для решения конкретной задачи, жёстко привязаны к физическим характеристикам конкретного механизма и требуют обширных знаний о выполняемом движении и динамике окружающей среды [149]. Походкам, порождённым традиционными методами, не хватает гибкости движений, вариативности форм, стабильности и плавности переходов [75]. Для аналитических моделей характерна низкая точность, что способствует внесению неопределённостей в результирующую динамику. Для решения соответствующих проблем, классического управления часто бывает недостаточно [99].

Метод имитации нуждается в трудоёмком процессе сканирования биомеханики движений оригинала и последующей постобработки. Результат применения метода напрямую зависит от качества полученной информации, а наличие морфологических различий между роботом и его оригиналом вносят ошибки в динамику, уменьшение которых требует применения дополнительных процедур (переназначение)[106].

Биоинспирированный подход, в виде применения центрального генератора паттернов, широко применяется в управлении локомоцией шагающих роботов и демонстрирует хорошие результаты. Несмотря на это, для создания адаптивной походки требуется модуляция параметров CPG, что на сегодняшний день остаётся по-прежнему сложной задачей. Во многих исследованиях они определяются либо опытным путём, либо посредством методов оптимизации, основанных на данных, например, оптимизация роя частиц [128].

В настоящее время большое распространение получил подход на базе машинного обучения, при котором формирование контроллера производится автономно на основе данных [40, 129]. Его применение позволяет автоматизировать большую часть ручной работы, связанной с построением системы управления и повысить манёвренность шагающих роботов за счёт высокой адаптивности [128]. Машинное обучение состоит из трёх основных направлений: обучение с учителем, обучение без учителя и обучение с подкреплением [56, 67]. Обучение с учителем формируется на основе маркированных данных, составленных квалифицированным сторонним специалистом (учителем). Каждый элемент данных представлен в виде некоторых значений входных параметров и метки, демонстрирующей правильное решение, которое система должна сгенерировать для конкретной ситуации. Цель обучения заключается в экстраполяции (обобщении) принятия решений на данные, не задействованные в обучающей выборке [56, 67]. Данный тип машинного обучения не подходит для построения системы управления роботом на базе взаимодействия [56]. Обучение без учителя преследует цель обнаружения скрытых структур и зависимостей в наборе не помеченных данных [56, 67]. Данный класс алгоритмов прекрасно выполняет классификацию, но не применим для формирования управления.

Обучение с подкреплением основывается на принципах поведенческой психологии, а именно на механизме поощрения, функционируя на базе вознаграждений и штрафов, получаемых при непосредственном взаимодействии со средой, что делает его применение в контексте управления локомоцией шагающих роботов актуальным и перспективным [18, 23, 34, 41, 44, 49, 56, 64, 67, 109].

Согласно анализу литературных источников можно сделать вывод, что для решения задачи управления локомоцией шагающего робота алгоритмы ОП могут быть применены к ранее описанным подходам в качестве вспомогательного инструмента или же в «чистом» виде, при определённой

формализации решаемой задачи. Например, подход, основанный на параметризованной математической модели с использованием алгоритмов обучения с подкреплением для поиска оптимальных значений параметров [39, 83]. Формирование центрального генератора паттернов, настраиваемого или управляемого методом ОП [93, 111, 128]. Реализация имитации путём построения стратегии, копирующей поведение оригинала на основе данных о движении [84, 92, 101, 131, 149, 154]. Помимо выполнения функции вспомогательного инструмента, алгоритмы ОП выступают в качестве основного ядра системы управления [50, 59, 74, 77, 85, 90, 99, 127, 134, 135, 139, 140, 147, 148, 150, 157]. При этом выделяются две основные формы реализации по виду производимого результата: генерация высокоуровневой команды, выполняемой исполнительной системой [39, 50, 90, 92, 128] и непосредственного контроля, в результате которого алгоритм генерирует низкоуровневые цели управления механизмов (углы сочленений) [85, 99, 114, 139, 140, 147, 150, 157].

Согласно анализу литературных источников, для решения задачи управления локомоцией шагающих роботов применяются следующие алгоритмы глубокого обучения с подкреплением: REINFORCE [39], DQN [59], TRPO [99, 140, 157], PPO [77, 84, 90, 92, 114, 131, 134, 139, 147, 148, 154], DDPG [128, 150], SAC [149], A3C [85]. Их подавляющее большинство относится к методам градиента стратегии. Это связано с тем, что градиентные алгоритмы, согласно концепции «смертельной триады» обладают большими шансами на сходимость, исходя из их принадлежности к методам с единой стратегией. В работах, посвящённых реализации управления локомоцией шагающего робота, использование алгоритма глубокого Q-обучения менее распространено и представлено в контексте формирования высокоуровневой управляющей команды, выбираемой из определённого множества малой размерности. Причина связана с невозможностью применения метода к задачам с большими или

непрерывными пространствами действий. Так же существенным фактором является наличие всех составляющих «смертельной триады», влияющих на сходимость алгоритма. Стоит отдельно отметить применение табличного Q-обучения к решению локомоторной задачи [97, 166]. Оно возможно исключительно в узком контексте, основанном на множестве допущений и с заранее определёнными малоразмерными множествами, как, например, в работе [97], где предварительно определена совокупность «устойчивых» состояний и действий.

1.3. Алгоритмы глубокого обучения с подкреплением

Первым, практически применимым масштабируемым алгоритмом глубокого обучения с подкреплением является DQN (Deep Q Network) [18, 41, 44, 49, 64, 123, 144]. Для решения проблемы корреляции данных в нём был применён буфер большого размера, называемый *буфером воспроизведения опыта*, а для ликвидации *нестационарности* отдельная нейронная сеть, генерирующая целевые значения. В результате, в структуру агента входят две нейронные сети: *онлайновая*, которая непрерывно обновляется, взаимодействуя с окружающей средой, и *целевая*, выполняющая обновление через каждые N итераций. Первый недостаток алгоритма заключается в том, что целевая сеть представляет собой устаревшую версию онлайновой, что приводит к замедлению процесса обучения [41, 123]. Вторым является проблема завышения оценки функции ценности по причине использования операции определения максимума. При неравномерном завышении и большой разнице в ошибках, происходит *избыточная коррекция весовых коэффициентов* сети и возникает *эффект расхождения*, что приводит к снижению устойчивости и качества найденной стратегии [18, 25, 41, 56, 63, 66].

На фоне успеха DQN были разработаны различные модификации, преследующие цель повышения его качества функционирования, устойчивости и эффективности. Алгоритм Double DQN (DDQN) сконцентрирован на решении проблемы завышения оценки [18, 41, 44, 49, 144, 159]. Для этого предлагается выполнить разделение процесса оценивания действия от его выбора. Определение следующей лучшей альтернативы осуществляется онлайновой (основной) нейронной сетью. После реализации выбора действия, целевая сеть оценивает его, генерируя значение Q-функции. Согласно экспериментальным данным, в результате алгоритм приобретает большую устойчивость [41, 159].

Ещё одной модификацией DQN является Dueling DQN [18, 41, 44, 49, 144, 160]. В контексте алгоритма производится формирование двух оценщиков: для ценности состояния и значений функции преимущества для каждого возможного действия, предпринимаемого из него [41]. Функция преимущества показывает насколько выбранная альтернатива лучше среднего из всех возможных [41, 56]. Далее полученные оценки используются для расчёта значений Q-функции каждого возможного действия из текущего состояния. Согласно первоисточнику [160], предлагаемая архитектура позволяет определять оценки состояний без непосредственного предсказания абсолютных значений ценности для каждого действия, что в значительной степени повышает качество функционирования алгоритма.

Все ценностные алгоритмы ОП оперируют ограниченным количеством действий. Когда в решаемой задаче возникает потребность в работе с очень большим множеством альтернатив или их непрерывным пространством, методы на основе ценности испытывают большие трудности, поскольку становится проблематичным или вовсе невозможным определение альтернативы с максимальным значением Q-функции [18, 41, 44, 49, 56, 64].

Алгоритмы градиента стратегии (АГС) являются альтернативой ценностных алгоритмов и базируются на аппроксимации не функции ценности, а непосредственно самой стратегии, при этом с лёгкостью решают проблему большого пространства действий [18, 41, 44, 49, 56, 64]. Если их множество не велико, то для реализации операции выбора применяется функция softmax. В случае наличия непрерывного пространства производится обучение статистике распределения вероятностей действий вместо вычисления отдельных обученных вероятностей для каждого из них [56]. Например, альтернативы могут быть представлены в виде вещественных чисел, а их выбор осуществляться из нормального распределения [56]. Градиентные методы базируются на целевой функции, представленной в виде суммарного дохода, который требуется максимизировать путём применения градиентного подъёма (gradient ascent) [18, 41, 44, 49, 56, 64]. Основной принцип АГС заключается в том, что алгоритм подталкивает агента к выбору наиболее «хороших» стратегий путём увеличения вероятности их появления и одновременно снижает вероятность «плохих» [41]. Градиентные методы являются представителями алгоритмов с единой стратегией, что с учётом «*смертельной триады*» повышает их шансы на сходимость, но, тем не менее, снижает *выборочную эффективность* [41]. В результате, после обновления стратегических весов, предыдущий опыт отбрасывается и для обеспечения поиска стратегии требуется гораздо больше опыта, чем алгоритмам с разделённой стратегией, таким как DQN.

Первым разработанным стохастическим градиентным методом является REINFORCE [18, 22, 29, 41, 44, 49, 56, 64, 162]. Он обладает хорошими показателями сходимости, поскольку обновление, происходящее после завершения эпизода, следует в том же направлении, что и градиент меры качества, тем самым обеспечивая приближение к локальному оптимуму (при соблюдении стандартных условий стохастической аппроксимации) и

улучшение качества при малых значениях шага [56, 162]. Данный метод имеет множество недостатков. Являясь алгоритмом Монте-Карло, он обладает низкой выборочной эффективностью, что влияет на скорость обучения и количество требуемого опыта. Существует тесная зависимость качества функционирования от длины эпизода. При возрастании размерности задачи растёт и вариативность выбора, что приводит к ощутимой разнице в длинах эпизодических траекторий, порождающей высокую дисперсию случайной величины суммарного дохода [17, 28, 40, 55]. Наличие дисперсии делает метод нестабильным. В попытках её устранения был разработан алгоритм REIFORCE with base, где предлагается ввести некоторое значение – базу, предназначенную для сравнения с ней предсказываемых ценностей выбираемых действий [18, 41, 56]. В качестве базы может выступать любая функция или случайная величина, не зависящая от выбора действия, но обладающая зависимостью от состояния [56]. На практике она представляется в виде нейронной сети, обучаемой на тех же данных, что и сеть стратегии. Внедрение базы позволило значительно уменьшить дисперсию, при этом сохранив отсутствие смещения случайной величины суммарного дохода за эпизод, но породило новую проблему – медленную сходимость. Для её решения было предложено использовать бутстреппинг. Данное изменение позволило обеспечить алгоритм асимптотической зависимостью от качества аппроксимации функции и внедрило смещение, что позволило уменьшить дисперсию и ускорить обучение. В результате, алгоритм был назван одношаговым исполнителем-критиком (Actor-Critic), где исполнителем является аппроксимируемая стратегия, а критиком прогнозируемая функция ценности следующего состояния [18, 26, 28, 29, 41, 44, 49, 56, 64]. Поскольку для ценностной оценки требуется только текущее вознаграждение, то метод относится к инкрементным и полностью онлайновым. Его недостатком является большое смещение. Для решения данной проблемы был предложен n-шаговый исполнитель-критик,

представляющий собой нечто среднее между методами TD(0) и Монте-Карло. Идея заключается в подборе такого количества шагов n до обновления, при котором достигается баланс между преимуществами, свойственными, как методам с единой стратегией, так и с разделённой. Изменение значения n позволяет регулировать величину смещения и дисперсии. Несмотря на свой потенциал, алгоритм исполнитель-критик имеет большой недостаток, связанный с нежелательными колебаниями распределения действий, что приводит к резкому падению производительности, после которого агент не может стабилизировать полноценное функционирование [18, 41, 44, 49, 56, 64].

Исполнитель-критик и REINFORCE демонстрируют хорошие результаты на небольших и средних задачах. С ростом сложности возникает множество проблем, которые в контексте соответствующих алгоритмов никак не могут быть решены [18, 26, 41, 49]. Во-первых, возникают трудности с определением правильного размера шага обновления. Она обусловлена нестационарностью, свойственной алгоритмам ОП. В процессе функционирования происходит непрерывное изменение распределения данных, при котором обученному агенту через время приходится исследовать другое множество состояний [26, 41]. Второй проблемой является неустойчивость. К сожалению, агент не имеет представления о качестве изменения стратегии. Неконтролируемое обновление способно кардинально изменить политику, в результате чего происходит резкое падение производительности, на восстановление которой может потребоваться значительное количество времени [26, 41]. Последняя проблема была освещена ранее. Это низкая выборочная эффективность, связанная с принадлежностью к алгоритмам с единой стратегией.

Для ликвидации неустойчивости, первоначально был предложен метод естественного градиента стратегии (Natural Policy Gradient, NPG) [41, 56, 104]. В данном алгоритме, дополнительно применяется вторая производная

целевой функции, демонстрирующая кривизну поверхности пространства распределений действий. Чем больше её значение между двумя точками, тем значительнее в них будет изменение градиента. Это позволяет обнаружить «обрывы» поверхности, являющиеся первопричиной неустойчивости. В результате, данная информация позволяет регулировать величину шага обучения, что не допускает резкого падения производительности. Основная идея метода заключается в гарантии малых изменений распределения действий при резких изменениях градиента [104]. Для этих целей используются информационная матрица Фишера (ИМФ), представляющая собой ковариационную матрицу целевой функции и расхождение Кульбака-Лейблера (КЛ), предназначенное для определения различия между распределениями действий [18, 41, 49, 104]. Благодаря ИМФ и КЛ становится возможным контроль размера и направления шага алгоритма, обеспечивающих постоянное расхождение КЛ и позволяющих выполнить «осторожное» обновление весовых коэффициентов сети стратегии [104]. При большом значении ИМФ шаг будет мал, а при малом ИМФ он будет большим. Недостатками NPG являются его применение исключительно к линейной аппроксимации и ресурсоёмкость вычисления ИМФ [41].

Алгоритм оптимизации стратегии в доверительной области (Trust Policy Optimization, TRPO) базируется на идеях NPG и является первым успешным алгоритмом, использующим несколько аппроксимаций, вычисляющих естественный градиент, которые предназначены для повышения устойчивости и управляемости стратегии, основанной на глубокой нейронной сети [41, 49, 140, 143]. Метод TRPO решает проблему вычисления ИМФ путём использования суррогатной функции. В контексте алгоритма вводится ограничение между старой и новой стратегиями на расхождение КЛ, образующее некоторую *доверительную область*. Это позволяет агенту оставаться в её пределах, выполняя более крупные шаги обновления [41]. Суррогатная целевая функция представлена таким образом,

что использует распределение состояний старой политики для произведения максимизации, относительно параметров новой на основе выборки по значимости [18, 41, 56]. Исходя из наличия доверительной области, алгоритм TRPO представляет собой оптимизационную задачу с ограничением, которая приближённо может быть решена на основе линейной аппроксимации целевой функции и квадратичной аппроксимации ограничения [41]. Решение производится при помощи метода сопряжённых градиентов, в результате чего получается локальная аппроксимация ожидаемого дохода. В момент улучшения нелинейной целевой функции, для гарантии выполнения нелинейного ограничения в момент улучшения, производится линейный поиск шага обновления максимальной длины, удовлетворяющего ограничению. TRPO может быть включён как в REINFORCE, так и в метод исполнитель-критик и использоваться для решения сложных задач управления [41, 140, 143]. Алгоритм демонстрирует эффективность использования ограничения доверительной области, но является очень сложным, поскольку содержит множество настраиваемых параметров.

В попытках снижения сложности был разработан алгоритм проксимальной оптимизации стратегии (Proximal Policy Optimization, PPO), использующий оптимизацию первого порядка без снижения надёжности в сравнении с TRPO, допускающий несколько обновлений на мини-пакетах и обладающий лучшей выборочной эффективностью [41, 49, 64, 142]. В случае большого отклонения PPO преследует цель не ограничить суррогатную целевую функцию, как это делается в TRPO, а обрезать её, что не позволит чрезмерно обновиться стратегии и резко снизить производительность. Целевая функция определяет отношение вероятностей старой и новой политик. Если оно больше или меньше, чем некоторая пороговая постоянная ε, то функция выбирает минимальное значение, что ограничивает отношение вероятностей в диапазоне, от $1-\varepsilon$, до $1+\varepsilon$ [41, 49, 64, 142]. В алгоритме PPO применяется *обобщённая оценка преимущества* (Generalized Advantage

Estimation, GAE) [141]. По сравнению с TRPO, на начальном этапе функционирования, алгоритму проксимальной оптимизации требуется больше опыта. Выполнив его накопление в достаточном количестве, происходит быстрый прогресс, в результате которого он обгоняет TRPO по производительности [41].

Несмотря на все преимущества градиентных алгоритмов, главным их недостатком остаётся низкая выборочная эффективность, что значительно повышает время обучения агента. С проблемой легко справляются алгоритмы с разделённой стратегией, но они не способны работать с непрерывным или очень большим пространством действий, как, например DQN. В идеале, необходим такой алгоритм, который способен обучаться с разделённой стратегией, но при этом работать с многомерными пространствами действий и находить устойчивые политики. Первой идеей, реализующей вышеописанную цель, успешно применённой к простым задачам, была реализация детерминированного исполнителя-критика, которая получила название детерминированный градиент стратегии (Deterministic Policy Gradient, DPG) [41]. В контексте алгоритма реализуется обучение детерминированной стратегии, аппроксимирующей операцию argmax, которая решает проблему вычисления максимума значения ценности по всему пространству действий [41]. Используются две стратегии, поведенческая, которая порождает траектории и отвечает за исследование пространства состояний и целевая, обучаемая на полученном опыте. В практических приложениях поведенческая политика представлена детерминированной с добавленным шумом. Обновление основывается на *детерминированном градиенте стратегии с разделённой стратегией*, оценивающем градиент относительно детерминированной политики [41]. В результате исполнитель представляет собой детерминированную стратегию, а критик – поведенческую. К сожалению, при использовании глубоких нейронных сетей, алгоритм становится неустойчивым и не способен к

обучению, поскольку возникают проблемы, описанные ранее при рассмотрении внедрения глубоких сетей в ценностные алгоритмы [18, 41, 44, 49, 64].

Первым удачным детерминированным алгоритмом, использующим нейронные сети для обучения исполнителя и критика, стал глубокий детерминированный градиент политики (Deep Deterministic Policy Gradient, DDPG) [41, 49, 115]. Он строится на двух основных идеях. Первая, заимствованная у алгоритма DQN – это буфер воспроизведения, в котором хранится вся история взаимодействия агента со средой. Подобно DQN, DDPG формирует мини-пакеты для обучения, на основе сгенерированного опыта, находящегося в этом буфере. Вторая идея заключается в использовании целевых сетей исполнителя и критика, которые подвергаются мягкому обновлению (частичному) на каждом шаге на основе параметров поведенческих сетей. Такой приём повышает устойчивость, но замедляет обучение. Алгоритм DDPG позволяет производить обновление сетей исполнителя и критика на каждом шаге взаимодействия, в результате чего нет необходимости в ожидании достаточного наполнения буфера. Обучение производится следующим образом. Происходит взаимодействие поведенческой стратегии со средой – генерация опыта. Производится обучение исполнителя и критика. Каждый шаг агента сопряжён с обновлением критика, путём минимизации функции потерь, представленной в виде среднеквадратического отклонения ценностей онлайнового критика и ценностей целевого, рассчитанных на базе целевой стратегии. Производится мягкое обновление целевой сети. Согласно полученным практическим результатам, алгоритм DDPG является выборочно эффективным, но крайне не устойчив, чувствителен к гиперпараметрам и требует их тщательной настройки [18, 41, 49, 64]. В попытке ликвидации возникших проблем был предложен алгоритм двойного глубокого детерминированного градиента стратегии с задержкой (Twin Delayed Deep Deterministic Policy Gradient,

TD3), основанный на DDPG [18, 41, 49, 64, 91]. В контексте алгоритма, для решения проблемы завышения оценки было применёно обрезанное двойное Q-обучение, смысл которого заключается в том, что вычисление целевой ценности производится на основе двух нейронных сетей (критиков), путём выбора минимального из рассчитанных ими значений. Ликвидация высокой дисперсии обеспечивается путём применения регуляризации целевой сети и приёму отложенного обновления [41]. Обновление откладывается до того момента, при котором ошибка критика будет незначительной. На практике это обеспечивается путём эмпирического подбора параметра задержки – количества шагов, после которого производится обновление. В алгоритме TD3 реализована сглаживающая регуляризация, представленная в виде добавления обрезанного шума в окрестности целевого действия [41]. Согласно полученным экспериментальным данным, алгоритм TD3 значительно превосходит DDPG по производительности и устойчивости [41]. Поскольку метод TD3 основан на DDPG, то сохраняется зависимость от качества настройки гиперпараметров.

Данные алгоритмы, как ранее упоминалось, используют вспомогательный шум, добавляемый в векторы действий для обеспечения элемента случайности, поскольку в их основе заложена детерминированная стратегия. Для упрощения подхода был предложен алгоритм мягкого исполнителя-критика (Soft Actor-Critic, SAC), схожий с DDPG и TD3, но в основе своей содержащий стохастическую стратегию [49, 95]. Его особенностью является включение энтропии стохастической политики в функцию ценности, максимизируемую агентом, что позволяет стимулировать разнообразное поведение при сохранении максимизации ожидаемого дохода. [49, 95]. Функция ценности в SAC формируется по аналогии с TD3. В практической реализации алгоритма используются две нейронные сети для аппроксимации Q-функции. Формирование конечной ценности осуществляется через операцию поиска минимума из предсказанных ими

значений. В отличие от TD3, SAC отдельно оптимизирует каждую сеть критика и внедряет энтропию к целевым значениям, альфа коэффициент которой автоматически регулируется в процессе функционирования агента, путём применения оптимизации альфа-канала, базирующейся на градиенте в направлении ожидаемой эвристической энтропии [49, 95]. Целевая энтропия обладает зависимостью от размерности пространства альтернатив. Практическая реализация SAC на сложной среде продемонстрировала хорошие показатели, но при этом, агент не смог достигнуть максимально возможного вознаграждения за эпизод [49].

Всем алгоритмам ОП свойственно длительное обучение. Для решения этой проблемы было предложено использовать параллелизацию, представленную в двух видах: синхронную и асинхронную. Принцип заключается в наличии множества действующих сетей, которые взаимодействуют в различных средах и накапливают опыт, обеспечивая несвязность и разнообразие данных. Для случая синхронной параллелизации, глобальная сеть перед обновлением ожидает завершения формирования пакета от всех действующих сетей. Для асинхронной, обновление происходит сразу при получении данных от какой либо из действующих сетей. Такой подход позволяет обеспечить стабильность работы алгоритма и устойчивость по отношению к шумам [18]. Синхронная параллелизация часто используется при реализации метода PPO. В работе [122] рассматривается алгоритм асинхронного исполнителя-критика с функцией преимущества (Asynchronous Methods for Deep Reinforcement Learning, A3C), представляющий собой применение асинхронной версии параллелизации к алгоритму исполнитель-критик. В процессе функционирования, действующие сети производят расчёт собственных градиентов, помимо формирования опыта, которые впоследствии передаются глобальной сети. Алгоритм A3C продемонстрировал лучшие на сегодняшний день результаты при применении его к играм Atari [18, 122]. К сожалению, отсутствие

блокировки при параллельном обновлении приводит к возникновению конфликтов [18]. Возникший эффект может быть минимизирован при введении допущения о том, что задача оптимизации является разреженной, то есть при обновлении параметризованной нейронной сетью функции будет преобразован лишь небольшой поднабор параметров [18]. В результате данный подход к параллелизации был именован Hogwild, жертвующий небольшим количеством конфликтов для ускорения обучения агента. Его основным недостатком является то, что допущение разреженности актуально только при малом количестве параллельных операций обновления, что свойственно небольшому количеству действующих сетей. Если их много, то количество конфликтов возрастает, делая допущение о разреженности не актуальным [18].

Подводя итог анализа существующих глубоких алгоритмов обучения с подкреплением, следует, что на текущий момент существует множество актуальных проблем, требующих рассмотрения. Во-первых, скорость обучения. Несмотря на внедрение параллелизации, агенту требуется большой временной интервал для формирования функциональной политики. Асинхронная параллелизация, с ростом количества действующих сетей, порождает конфликты обновления глобальной стратегии, в результате чего падает производительность агента. Остаётся проблема стабильности работы алгоритма, связанная с наличием дисперсии и смещения. В попытках её решения, методы ОП становились более «громоздкими», повышая требования к вычислительным ресурсам и производительности вычислительных систем. Большинство прикладных задач робототехники, включая локомоцию шагающих механизмов, строятся в условиях ограниченных ресурсов. Поэтому построение ресурсосберегающего алгоритма, является актуальной задачей в этой области. По своей природе, роботы являются системами реального времени, для которых, важным критерием является время отклика при принятии решений. С этой точки

зрения, необходимо наличие быстро реагирующего online-алгоритма, способного в достаточно короткий срок сформировать сигнал управления в ответ на изменения окружающей среды. Важным недостатком, свойственным глубоким алгоритмам ОП, является его точность. Как ранее отмечалось, табличные алгоритмы обучения с подкреплением сходятся к оптимальной стратегии, что не возможно по причине несовершенства аппроксимации нейронными сетями для глубоких методов [56]. В результате, происходит приближение к локальному оптимуму, который не всегда в полной мере способен обеспечить качественное управление.

1.4. Особенности реализации многофункционального управления мобильным шагающим роботом

Одной из важнейших задач, которые требуется решить при управлении мобильным шагающим роботом как частного случая киберфизической системы, является возможность решения им нескольких задач (Multitask/Multi-skill). Число таких задач может возрастать (например, при переходе из одной среды на другую), меняться, что заставляет предъявлять к разрабатываемой системе управления требования, связанные с обеспечением возможности многофункционального поведения агента. В контексте применения глубокого обучения с подкреплением к управлению, множественность форм поведения обеспечивает *информационная ёмкость агента*. В данном контексте, под агентом понимается глубокая нейронная сеть, аппроксимирующая стратегию. К сожалению, он способен обучаться выполнению только одной операции [128]. При смене задачи, агент перестраивает весовые коэффициента под новые условия, что приводит к эффекту «забывания». Возникает проблема повышения репрезентативной способности глубокой сети. В попытках её решения, были предложены

разнообразные подходы к формированию архитектуры агента. Среди них выделяются:

1. Иерархический [74, 89, 99, 101, 147] – стратегии представляют собой отдельные обученные модули, выполнение которых осуществляется посредством сети-менеджера, которая в зависимости от условий осуществляет выбор необходимого в конкретный момент модуля. Использование отдельных модулей-стратегий позволяет решить проблему «забывания» и обеспечить многофункциональность. Данная структура агента имеет множество недостатков. Архитектурная сложность приводит к трудностям в обучении и формировании вознаграждения. Поскольку имеется множество уровней, то необходимо обучить каждую стратегию и сеть-менеджер. В результате, обученный агент не гарантирует высокое качество функционирования.

2. Смесь экспертов (Mixture of Experts, MoE) [151] – позволяет разделить вычисления между экспертами, представляющими собой отдельные нейронные сети. Выбор экспертов-обработчиков осуществляется на основе токенов входных данных при помощи сети-маршрутизатора. Данная архитектура, в контексте решения задачи управления локомоцией, имеет те же преимущества и недостатки, что и иерархическая архитектура, но дополнительно нуждается в необходимости выполнения регуляризации, поскольку возникает вероятность возникновения «пустых» экспертов, вероятность выбора которых близка или равна нулю.

3. Дистилляция [96, 136] – подход строится на обучении одной студент-стратегии на заранее обученных стратегиях. Основным преимуществом является объединение нескольких стратегий в контексте одной. Подход имеет множество недостатков: крупный размер результирующей стратегии, необходимость в хранилище значительного объёма обучающих данных, которые соответствуют каждой из стратегий-учителей. К основному недостатку относится потеря важных нюансов,

характерных для каждой решаемой задачи, в результате чего ухудшается качество функционирования системы управления.

4. Прогрессивные нейронные сети (Progressive Neural Networks) [137] строятся на специализированной архитектуре нейронной сети, в которой каждому локомоторному паттерну выделяется собственный столбец-сеть, после обучения которого, производится заморозка его весовых коэффициентов. К его недостаткам относится возрастание количества параметров с ростом числа решаемых задач и отсутствие метода селекции столбцов, поиск которого на сегодняшний день является открытой проблемой.

Решение проблем «забывания» и обеспечения агента многофункциональностью в большинстве подходов осуществляется путём выделения отдельной нейронной сети для каждой необходимой стратегии и использования сложной архитектуры системы управления. В результате чего возникает множество проблем, связанных с настройкой и селекцией стратегий. Несмотря на существующие функционирующие подходы, базовая причина низкой репрезентативной способности глубокой сети не решается.

1.5. Особенности переноса обученного в симуляции агента на реального робота

Использование алгоритмов глубокого обучения с подкреплением в прикладных задачах робототехники, помимо наследования недостатков применяемых методов, сопряжено с дополнительными трудностями, связанными с реализацией агента в контексте реальной системы. До настоящего момента, успешные примеры политики обучения «напрямую», непосредственно на физических роботах, были продемонстрированы исключительно в относительно ограниченных областях [130]. В процессе функционирования, агент может генерировать действия, способные

повредить дорогостоящие механизмы робота или полностью вывести его из строя, поэтому большинство работ основано на подходе предварительного обучения в имитационных средах [77, 84, 92, 99, 101, 128, 130, 139, 140, 145, 154]. При переносе обученного агента на реального робота, зачастую наблюдается нестабильная работа, связанная с неточностью используемых моделей окружающих сред и, как следствие, наличием неучтённых помех реального мира [92, 128, 130, 139, 145]. Для решения соответствующих проблем были предложены такие методы, как: моделирование приводов [99], внедрение модульного подхода к обучению [167], рандомизация предметной области [130, 155], усиление обобщения политики [82] и добавление шума к наблюдениям и действиям в среде обучения. Несмотря на хорошие результаты, демонстрируемые применением предварительной симуляции, всё ещё остаётся актуальным поиск подхода, способного обеспечить стабильное и безопасное обучение робота. Одной из перспективных идей в данном направлении является внедрение ограничений [71, 92, 152].

В контексте шагающей робототехники важным является определение баланса, которое тесно связано с обеспечением безопасности функционирования и обучения агента. Существуют следующие подходы определения устойчивости:

– проекция точек нулевого момента и центра масс внутри опорного многоугольника, образованного точками соприкосновения конечностей с землёй (ZMP + CoM);

– положение точки захвата с использованием упрощённой модели инвертированного маятника (CP + LIPM);

– метод, основанный на модели (Model Base);

– модель прогнозирующего управления (MPC).

К недостаткам первого подхода относится необходимость в точной модели. Даже при её наличии не учитываются удары и проскальзывания робота во время локомоции. Подход демонстрирует плохие результаты в

динамике. Применение упрощённой модели инвертированного маятника в совокупности с точкой захвата не учитывает сложную механику движения. В результате чего возникают коллизии. Подход, основанный на модели, ограничен контекстом и не обладает требуемым уровнем адаптации. Для прогнозирующего управления также требуется модель. Подход сопряжён с вычислительной сложностью, что критично для области робототехники. В результате следует, что поиск оптимального подхода к определению сбалансированности движения в процессе локомоции является актуальной проблемой.

1.6. Цели и задачи диссертационного исследования

На основании анализа, приведенного в данной главе, можно сделать следующие выводы.

1) существующие подходы (в первую очередь, классические, основанные на математических моделях) не в полной мере гарантируют стабильность функционирования мобильным шагающим роботом;

2) при переносе агента из симуляции на реальные киберфизические системы возникает сложность с обеспечением одновременного решения базовых задач;

3) существующие методы глубокого обучения и архитектуры нейронной сети не в полной мере решают задачи многофункциональности агента, возникающие при масштабировании;

4) в связи с этим, существующие системы управления либо демонстрируют ограниченную обобщающую способность, либо страдают от деградации при переносе обученного агента из симуляции.

Анализ литературных источников позволил подчеркнуть актуальность проводимого исследования в области применения глубокого обучения с подкреплением к решению проблемы формирования управления локомоцией

шагающего робота. В результате были определены следующие актуальные для рассмотрения проблемы:

1. Необходимость в формировании управления шагающим механизмом, обеспечивающем стабильное передвижение в пространстве, не зависящее от вида и формы рельефа;
2. Необходимость в формировании методологии формирования управления шагающим роботом универсально применимой для различных типов конфигурации шагающих систем;
3. Формирование подхода, позволяющего обеспечить сходимость алгоритма к наилучшему локальному оптимуму или же к оптимальной стратегии;
4. Решение проблемы «блуждающей мишени» (нестационарности) глубоких алгоритмов обучения с подкреплением;
5. Построение ресурсосберегающего алгоритма глубокого обучения, обеспечивающего устойчивое функционирование;
6. Реализация ускорения реакции глубокого алгоритма на изменения среды;
7. Поиск решения задачи нахождения баланса между отсутствием смещения и выборочной эффективностью;
8. Поиск подхода, способного обеспечить повышение стабильности и безопасности обучения и функционирования робота;
9. Формирование методологии построения функции вознаграждения без использования эталонных траекторий и дополнительных шагов обработки;
10. Поиск возможности объединения множества локомоторных операций в одном агенте.

В контексте настоящего исследования основной акцент направлен на решение проблем 1, 2, 8 и 10. Поскольку при применении обучения с подкреплением, стороной принимающей решения является

интеллектуальный агент, то для обеспечения контроля принимаемых решений требуется система управления соответствующим агентом. Исходя из этого, и на основе анализа литературных источников была сформулирована основная цель диссертационного исследования.

Целью работы является разработка адаптивной системы управления интеллектуальным агентом на основе обучения с подкреплением, обеспечивающей обучение множеству стратегий с минимизацией коллизий на опыте, полученном при непосредственном взаимодействии киберфизической системы с окружающей средой и автономную адаптацию под изменения среды на примере решения задачи управления локомоцией шагающего робота.

В соответствии с выделенными проблемами были определены следующие задачи, которые будут рассмотрены в дальнейших частях настоящего диссертационного исследования:

1. Разработать математическое обеспечение системы управления интеллектуальным агентом, обеспечивающее повышение уровня безопасности и устойчивости функционирования управляемой киберфизической системы.

2. Разработать специальное алгоритмическое обеспечение системы управления интеллектуальным агентом, обеспечивающее ускорение обучения и переноса агента на реальную систему за счёт увеличения времени устойчивого функционирования и сокращения числа нештатных режимов (коллизий).

3. Разработать архитектуру глубокой нейронной сети, обеспечивающей агента, основанного на глубоком обучении с подкреплением, возможностью обучения нескольким стратегиям в контексте одной глубокой нейронной сети и ликвидирующей эффект «забывания» и потерю деталей каждой из стратегий.

4. Разработать структуру системы управления интеллектуальным

агентом, обеспечивающую автономное адаптивное безопасное обучение множеству стратегий с последующим переносом на реальную киберфизическую систему и безопасной настройкой на примере формирования локомоции шагающего робота.

5. Произвести экспериментальное исследование эффективности разработанного математического и алгоритмического обеспечения, архитектуры многофункционального агента и структуры системы управления.

На основании сформулированных цели и задач был реализован следующий дизайн диссертационного исследования, представленный на рисунке 1.1.

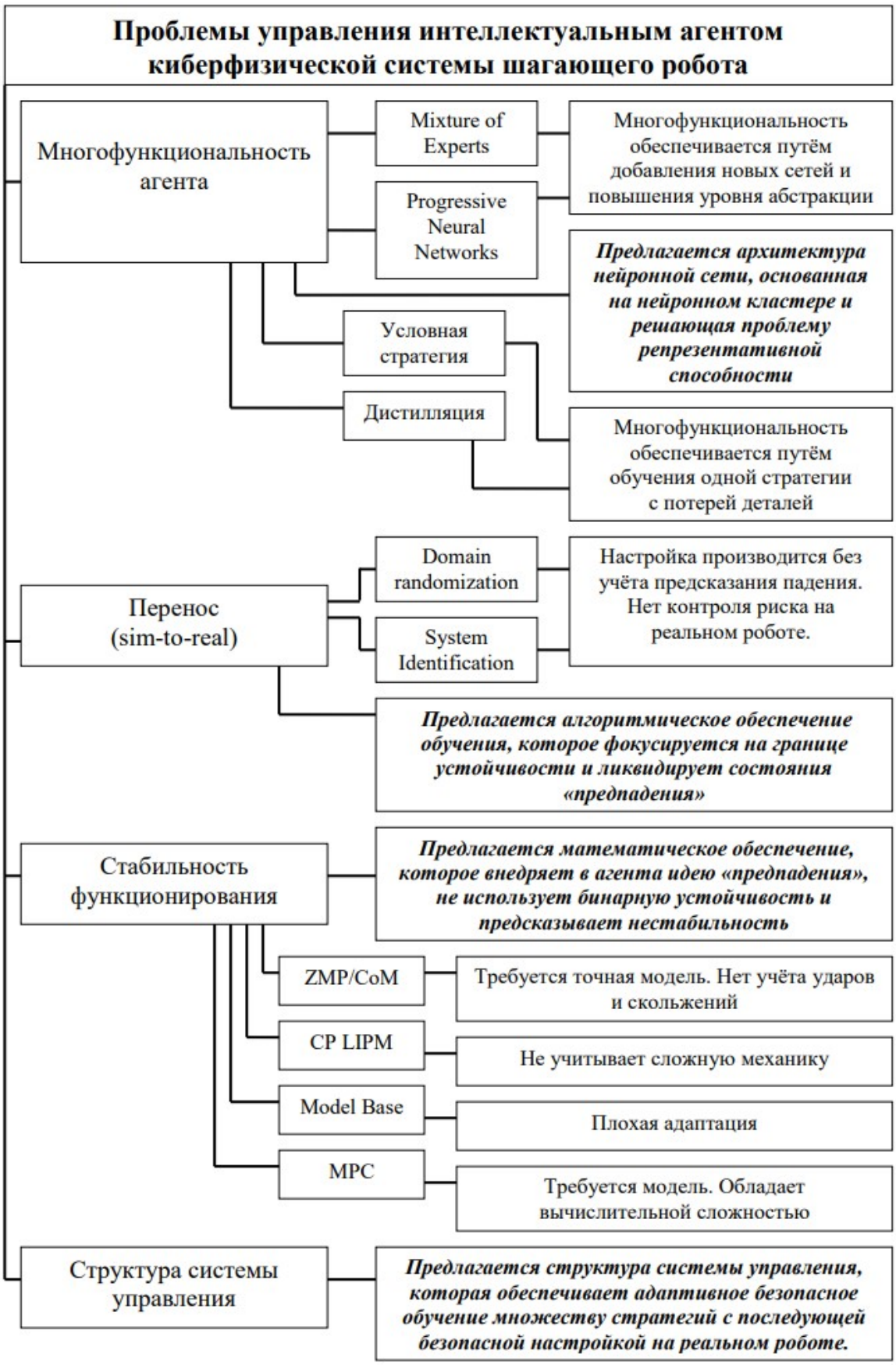


Рисунок 1.1 – Дизайн диссертационного исследования

ГЛАВА 2. МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ИНТЕЛЛЕКТУАЛЬНЫМ АГЕНТОМ КИБЕРФИЗИЧЕСКОЙ СИСТЕМЫ МОБИЛЬНОГО ШАГАЮЩЕГО РОБОТА

В рамках настоящей главы рассматривается детализация задачи управления локомоцией мобильного шагающего робота. Осуществляется конкретизация характеристик объекта управления и окружающей среды. Формируется математическое обеспечение системы управления с точки зрения применения глубокого обучения с подкреплением. Вводится концепция критических состояний – множества состояний, представляющего границу между устойчивыми состояниями и неустойчивыми. Производится внедрение концепции в аппарат обучения с подкреплением.

В начале рассматриваются основные требования к разрабатываемой системе управления. Реализуется постановка и детализация задачи. Вводятся допущения. Формируется интерфейс системы управления на уровне чёрного ящика. Производится описание динамики и построение математического обеспечения системы управления с точки зрения марковских процессов принятия решений. Определяются и рассматриваются критические состояния. Вводится функция риска, демонстрирующая удалённость текущего состояния от области неустойчивых состояний. Рассматривается её внедрение в процесс обучения. Далее производится рассмотрение динамика шагающего робота в соответствии с использованием глубокого обучения с подкреплением. Определяются системы координат робота. Вводится универсальный математический аппарат решения прямой задачи кинематики. Рассматриваются контрольные характеристики, необходимые для определения сбалансированности движения. На их основе формируются критерии устойчивости, которые модифицируются с использованием введённой в контексте исследования концепции критических состояний.

Рассматривается построение универсальных ограничений для каждого типа шагающих платформ, отличающихся количеством действующих педипуляторов. На их основе производится формализация управления локомоцией мобильного шагающего робота в форме задачи оптимизации с учётом наличия критических состояний. В конце приводятся выводы к соответствующей главе.

2.1. Постановка и детализация задачи управления интеллектуальным агентом, реализующим локомоцию мобильного шагающего робота

Исходя из цели исследования, приведенной в предыдущей главе, разрабатываемая система управления должна быть способна обучаться множеству двигательных программ на опыте, полученном при непосредственном взаимодействии механизма с окружающей средой и автономно подстраиваться под неровности рельефа местности. Из формулировки следует глобальный перечень требований к разрабатываемой системе управления:

- адаптивность – система управления должна обладать способностью коррекции поведения управляемого объекта в соответствии с изменениями окружающей среды;

- самообучаемость – система управления должна обладать способностью самостоятельного формирования стратегии генерации управляющих сигналов на основе полученного опыта от взаимодействия объекта управления с окружающей средой;

- универсальность – система управления должна обладать математическим и алгоритмическим обеспечением, позволяющим полностью или частично устранить зависимость от физических характеристик объекта управления при генерации управляющих сигналов;

- многофункциональность – система управления должна обеспечивать возможность формирования множества паттернов поведения с учётом форм рельефа местности;

- оптимальность – система управления должна обладать простым и понятным математическим аппаратом и оптимально использовать вычислительные ресурсы в процессе генерации управляющих сигналов.

На начальном этапе решения задачи управления интеллектуальным агентом, реализующим локомоцию мобильного шагающего робота, следует определиться с её условием. Пусть некоторая робототехническая шагающая платформа, обладающая двумя или более конечностями (педипуляторами) шарнирного типа, помещена в заранее не детерминированную среду, которую в определённый момент времени можно охарактеризовать кривизной и шероховатостью рельефа, вязкостью грунта. Возможно наличие препятствий разного размера. Перед устройством ставится задача самообучения выполнению указанной локомоторной программы из множества допустимых на опыте непосредственного взаимодействия со средой и адаптации полученной стратегии под разнообразные типы поверхностей [30, 108].

Базовыми единицами в теории автоматического управления являются объект управления и управляющее устройство. Тесное взаимодействие данных компонент формирует единую систему, обеспечивающую решение требуемых задач. Качество функционирования такой системы напрямую зависит от оптимальности решений, принимаемых управляющим устройством. В качестве объекта управления рассматривается электромеханическая система перемещения киберфизической системы робота, обладающая двумя и более педипуляторами. Исходя из характеристик управляемого объекта и окружающей среды, устройство управления должно состоять из двух связанных блоков: блока принятия решений, производящего выбор последовательности высокоуровневых

действий, и генератора сигналов управления, преобразующего действия в низкоуровневые команды управления приводами педипуляторов. В основе цифровых систем управления роботами используются микропроцессорные системы, являющиеся по своей природе дискретными устройствами. В соответствии с этим, разрабатываемая система управления обладает характеристикой дискретности во времени. Поскольку модель окружающей среды, в которой будет функционировать механизм, заранее неизвестна, и отсутствует полная наблюдаемость, то возникает вероятностная компонента при принятии решений, что характеризует систему управления как открытую и стохастическую. Для обеспечения выполнения требований самообучаемости и адаптивности в основе разрабатываемой системы управления будет использован гибкий аппарат глубокого обучения с подкреплением, который обеспечивает формирование стратегии принятия решений на основе фактического опыта, полученного методом проб и ошибок.

Для упрощения решаемой задачи в контексте настоящего исследования будет рассмотрена одна локомоторная программа – «движение прямо», реализованная для двух типов поверхности: ровная и песчаная. Классификация окружающей среды и определение подобия стратегий представляют собой задачи высокоуровневой подсистемы управления, генерирующей выбор локомоторной программы, которая не рассматривается в рамках настоящей работы. На уровень ниже располагается основная подсистема, формирующая стратегии, которой посвящено настоящее исследование. В результате классификация сред и рандомизация стратегий не рассматриваются в контексте настоящей работы, что означает, что каждой возможной локомоции будет соответствовать собственная стратегия принятия решений (даже если генерируемые ими паттерны схожи между собой). Для простоты реализации, программа движения задаётся оператором вручную.

Все предложенные идеи к реализации системы управления интеллектуальным агентом, реализующим локомоцию шагающего робота, базирующегося на глубоком обучении с подкреплением, будут продемонстрированы и протестированы на примере киберфизической системы шагающего робота с двумя конечностями, обладающими пятью степенями свободы каждая. В результате, тестовый стенд представляет собой устройство с десятью степенями свободы. Тестовый робот схематически представлен на рисунке 2.1.

Основным минимальным набором сенсоров, необходимым для построения системы управления, является гироскоп-акселерометр и датчики касания поверхности земли. Тестовый стенд реализован на основе сервоприводов, с диапазоном вращения от 0 до 180 градусов. В большинстве случаев, для обеспечения точности управления и позиционирования конечностей в пространстве применяются энкодеры, осуществляющие считывание фактического угла, соответствующего датчику сочленения. Чтобы облегчить реализацию механизма, в контексте настоящей работы делается предположение о том, что задаваемый угол сочленения соответствует фактическому, что позволяет отказаться от датчиков, а текущие значения хранить в специализированном списке.

Используя идеи, заложенные в реализуемой системе управления, становится возможным расширение её функционала для других локомоторных программ и форм рельефа местности, а также других робототехнических шагающих платформ, обладающих иным количеством педипуляторов.

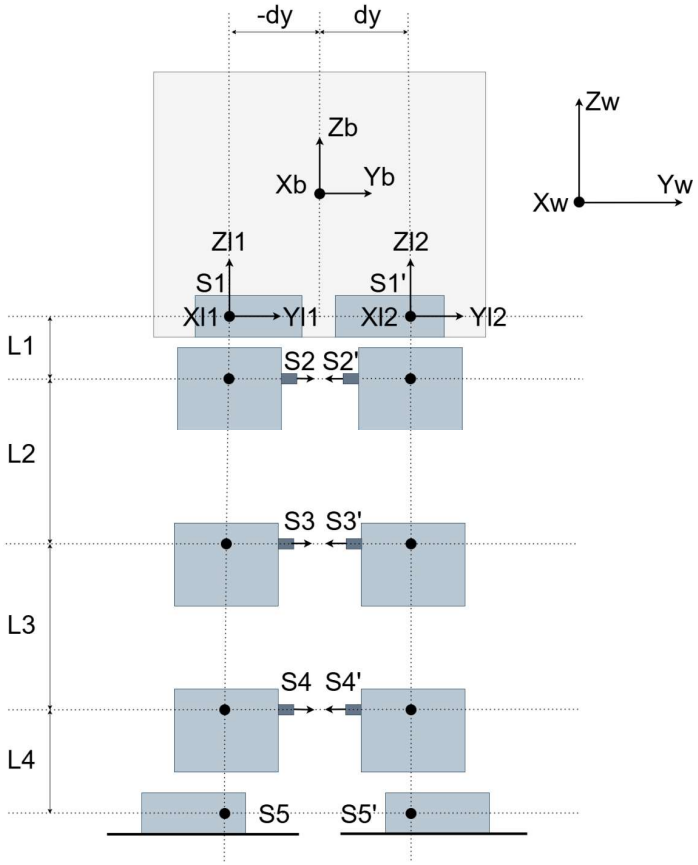


Рисунок 2.1 – Тестовый стенд

Ещё одним допущением в контексте настоящего исследования является скорость робота. В работе производится построение системы управления локомоцией при низких скоростях перемещения. Высокоскоростные паттерны имеют определённые нюансы, которые в контексте настоящего исследования не рассматриваются. В дальнейшем они могут быть добавлены путём расширения существующей системы управления.

2.2. Интерфейс системы управления интеллектуальным агентом, реализующим локомоцию киберфизической системы мобильного шагающего робота

Определим интерфейс системы управления интеллектуальным агентом, реализующим локомоцию киберфизической системы мобильного шагающего

робота на уровне чёрного ящика. Конструкция механизма состоит из подвижных конечностей рычажного типа, каждая из которых содержит несколько сочленений, характеризуемых величиной угла отклонения θ_i, указанного в радианах. Пусть общее число сочленений робота равно N. Тогда в момент времени t состояние механизма представимо в виде кортежа, содержащего значения углов сочленений следующего вида [30, 108]:

$$d_t = (\theta_1, \theta_2, \theta_3, ..., \theta_N), \tag{2.1}$$

где N – количество сочленений робота, θ_i – угол (в радианах) i- ого сочленения, d_t – состояние робота в момент времени t.

Исходя из этого, для реализации управления перемещением механизма необходимо осуществлять поочерёдное чередование кортежей в каждый дискретный момент времени t. Следовательно, вектор d представляет собой выходную шину управляющего устройства.

Также агенту следует указать предыдущее состояние d_{t-1}. Для обеспечения демпфирования в процессе движения необходимо указать угловые скорости приводов, которые рассчитываются на основе разности углов с последующей фильтрацией:

$$\upsilon_{filtered} = \alpha \upsilon_{servo}(t) + (1-\alpha)\upsilon_{servo}(t-1), \tag{2.2}$$

где α – коэффициент фильтрации, $\upsilon_{servo}(t) = \frac{\theta(t) - \theta(t-1)}{\Delta t}$ – скорость в момент времени t, $\upsilon_{servo}(t-1)$ – скорость в предыдущий момент времени.

Механизм фильтрации и подбора коэффициента α детально рассмотрены в Приложении С.

Осуществление локомоции подразумевает реализацию перемещения робота в пространстве с одновременным удержанием устойчивого

равновесия конструкции. Для выполнения соответствующих операций механизму необходимо анализировать собственные показатели ориентации в пространстве и скорости для определения корректности направления движения и факта удержания баланса. Информацию о пространственном позиционировании механизма можно получить при помощи датчика гироскопа-акселерометра, который позволяет измерить углы отклонения робота и ускорения относительно нормальной системы координат. В результате, сенсор на выходе предоставляет следующую информацию: угол тангажа – η, угол крена – μ, угол рысканья – σ и ускорения a_x, a_y, a_z. Скорость робота эквивалентна скорости точки центра масс. Поэтому во входной вектор необходимо добавить проекции вектора скорости υ_x^{CoM} и υ_y^{CoM}. Каждой конечности соответствует датчик касания с поверхностью земли. В результате, получим кортеж касания $T = (T_1, T_2, ..., T_N)$, состоящий из фиксации факта соприкосновения с поверхностью земли, где T_i соответствует i-ому педипулятору и принимает значения $[0,1]$.

Поскольку для обеспечения адаптивности в контексте работы будет использовано обучение с подкреплением, то для обозначения факта завершения эпизода взаимодействия используется бинарный флаг f, принимающий значения из множества $F : \{0,1\}$.

В результате, интерфейс системы управления локомоцией мобильного шагающего робота, на уровне чёрного ящика имеет следующий вид:

$$\begin{gathered} OUT : (d_{t+1}) \\ IN : \left(d_t, d_{t-1}, \dot{d}_t, \upsilon^t_{CoM_x}, \upsilon^t_{CoM_y}, a_x, a_y, a_z, \eta, \mu, \sigma, T_t, f\right) \end{gathered} \tag{2.3}$$

2.3. Формализация общей динамики системы управления

Взаимодействие агента с окружающей средой в процессе обучения с подкреплением производится по средствам действий, каждое из которых оценивается средой положительным или отрицательным вознаграждением [56]. Вознаграждение аккумулируется на протяжении эпизода и используется для оптимизации функции ценности или стратегии управления. Задача формирования управления локомоцией шагающего робота представляет собой открытую динамическую систему, обладающую дискретностью во времени и наличием аддитивной функции вознаграждения. По своей структуре такие системы в большей степени являются детерминированными. Ранее обсуждалось, что робот не способен полностью определить текущее состояние среды, а оперирует с небольшим подмножеством его параметров, представленных в виде наблюдения, что приводит к частичной наблюдаемости. Она, в свою очередь, порождает стохастичность – вероятностный элемент в управлении.

Пусть пространство реальных состояний системы представлено как $S \subseteq R^J$, где $s_t \in S$ – состояние среды в момент времени t. Тогда пространство наблюдений имеет вид $X \subseteq R^I$ и является подпространством множества реальных состояний S, где $x_t \in X$ – наблюдение среды в момент времени t. Множество всех доступных агенту действий задано, как $A \subseteq R^K$, где $a_t \in A$ – действие агента в момент времени t. Величины $I, J, K \in N$ соответствующие мощности множеств R^I, R^J, R^K и $t = 1,...,\infty$.

Следовательно, динамику системы можно определить в следующем виде:

$$\begin{aligned} s_{t+1} &= g(s_t, a_t) \\ x_t &= h(s_t) \end{aligned} \tag{2.4}$$

где $g: R^J \times R^K \to R^J$ – функция перехода в контексте реальных состояний системы, $h: R^J \to R^I$ – функция отображения реального состояния на наблюдение.

В (2.4) представлена функция перехода из текущего состояния s_t в новое состояние s_{t+1}, под влиянием выбранного действия a_t, с реализацией последующего преобразования его в наблюдение x_t. Поскольку киберфизическая система робота ограничена небольшим множеством сенсоров, то в результате получается вектор x_t, представляющий ограниченный «слепок» реального состояния s_t.

Пусть система является полностью наблюдаемой. Тогда функция вознаграждения, ставящая в соответствие некоторое число r_t, представляющее собой награду за посещение состояния s_t состоянию s_t будет представлена в следующем виде:

$$R: S \to R \tag{2.5}$$

Поскольку робот оперирует «слепком» x_t реального состояния, являющимся упрощённым его аналогом, то функция вознаграждения в данном случае примет вид:

$$R: X \to R \tag{2.6}$$

В практических задачах зачастую применяется дисконтированная функция вознаграждения (с обесцениванием):

$$R = \sum_{t=1}^{\infty} \gamma^{t-1} r_t \,, \tag{2.7}$$

где $\gamma-$ коэффициент обесценивания, принимающий значения в диапазоне [0,1], r_t- сигнал вознаграждения, полученный в момент времени t.

2.4. Формализация управления в рамках марковского процесса принятия решений (MDP)

Рассмотрим разрабатываемую систему управления в рамках обучения с подкреплением. Формальная постановка задачи через марковский процесс принятия решений имеет вид:

$$M = (S, A, P, R, \gamma), \tag{2.8}$$

где $s_t \in S$ – полное состояние среды, $a_t \in A$ – действие, $P(s_{t+1} \mid s_t, a_t)$ – динамика, $R(s_t, a_t)$ – вознаграждение, γ – коэффициент обесценивания.

Переходы $P(s_{t+1} \mid s_t, a_t)$ задаются в следующем виде:

$$s_{t+1} = f(s_t, a_t, \xi_t), \tag{2.9}$$

где f – неявная динамика робота, ξ_e – шум (контакт, проскальзывание, неточности).

В результате марковский процесс имеет следующий вид [37]:

$$\begin{aligned} & s_t = (d_t, \dot{d}_t, \omega_t, \nu_t, c_t) \\ & a_t = \Delta d_t \\ & s_{t+1} \sim P(s_t, a_t) \\ & R_t = R(s_t, a_t) \\ & \pi = \arg\max \mathrm{E}\left[\sum \gamma^t R_t\right] \end{aligned}, \tag{2.10}$$

где s_t – текущее состояние среды, d_t – углы сочленений (текущее состояние робота), $\dot{d}_t$ – угловые скорости сочленений, ω_t – угловая скорость робота, v_t – линейное ускорение, c_t – контакты конечностей с поверхностью земли, a_t – действие, π – стратегия.

Робот оперирует с наблюдениями $x_t \in X$. Переход между состояниями осуществляется посредствам действий $a_t \in A$ и сопровождается оценкой качества перехода в виде сигнала вознаграждения $r_t \in R$, поступающего со стороны окружающей среды в качестве обратной связи [56]. Поскольку выбор действия осуществляется на основе неполной информации о состоянии окружающей среды, возникает частичная наблюдаемость, привносящая стохастичность в управление. Из интерфейса системы управления, представленного в (2.3) и уравнений динамики (2.4) следует, что наблюдение, в контексте разрабатываемой системы управления имеет следующий вид:

$$x_t = \left(d_t, d_{t-1}, \dot{d}_t, \upsilon^t_{CoM_x}, \upsilon^t_{CoM_y}, a_x, a_y, a_z, \eta, \mu, \sigma, T_t, f\right), \tag{2.11}$$

Совокупность, всех возможных x_t образуют множество наблюдений окружающей среды X. В результате, стратегия строится и функционирует исключительно на основе множества X, в то время как динамика строится на основе реальных состояний s_t.

2.5. Концепция критических состояний

Кортеж s_t представляет собой состояние робота в момент времени t. Объединив все возможные кортежи вида s_t, получим общее множество состояний робота S. Среди них выделим *устойчивые состояния* –

подмножество S_{safe} и S_{fail} – *неустойчивые*, приводящие к повреждению механизма или его падению, такие, что:

$$S = S_{safe} \cup S_{fail} \tag{2.12}$$

Применение алгоритмов обучения с подкреплением в контексте управления робототехническими системами обладает очевидными преимуществами, но в то же время сопряжено с множеством проблем. Наиболее значимой из них является формирование действий, приводящих к повреждению механических узлов в результате падения робота или переход в конфигурацию, конструктивно не поддерживаемую механизмом. Решение задачи формирования управления шагающим роботом является непрерывным процессом генерации действий в соответствии с наблюдениями окружающей среды. В таких случаях в теории обучения с подкреплением применяется величина горизонта T, представляющая искусственную границу разделения процесса взаимодействия на эпизоды. При достижении соответствующей временной границы эпизод считается завершённым, что приводит к обнулению счётчика временных интервалов на следующем шаге взаимодействия. С учётом контекста решаемой задачи для качественного обучения необходимо учитывать факт падения или повреждения механизма робота за счёт генерации недопустимых конфигураций конечностей, что так же считается завершением текущего эпизода взаимодействия. Ранее в (2.12) было отмечено, что среди существующих состояний робота выделяются *неустойчивые состояния*, приводящие к повреждению механизма или его падению. Поскольку возникновение состояния из множества S_{fail} гарантированно приводит к завершению эпизода, для случая робототехники важным является возможность последующего восстановления взаимодействия механизма со средой и способа этого восстановления в процессе обучения. В случае

падения зачастую требуется участие человека. Поскольку конструкция механизма представляет собой дорогостоящее оборудование, то нецелесообразно допускать повреждение его узлов. Если же предотвратить данное событие не удалось, то робот на длительный срок выходит из строя до восстановления механической части. Всё это негативно влияет, в первую очередь, на скорость и качество обучения. Агент, на котором строится управление киберфизической системой робота, должен обучаться посредством взаимодействия со средой и не допускать возникновения состояний из множества S_{fail} или обеспечить минимизацию их возникновения. Для этого в контексте настоящей работы вводится понятие «*критическое состояние*». ***Критическое состояние*** представляет собой такое состояние, при котором робот ещё устойчив, но находится на грани потери баланса или близок к механическим повреждениям узлов [32]**.** Другими словами, критические состояния образуют подмножество пограничных состояний $S_{critical}$, расположенных на границе между гарантированно устойчивыми состояниями и неустойчивыми, как представлено на рисунке 2.2.

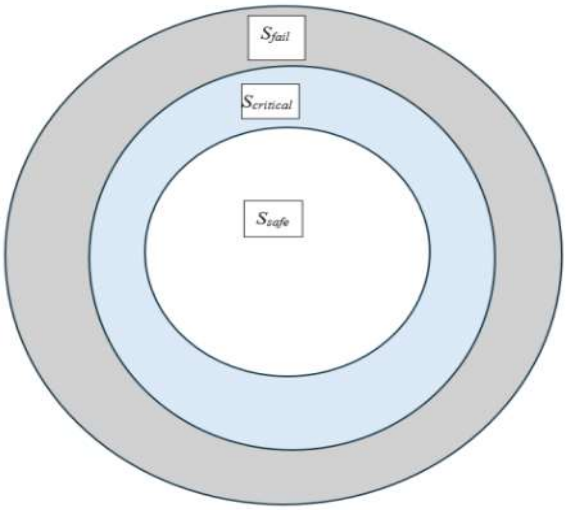


Рисунок 2.2 – Диаграмма состояний системы с введением подмножества критических состояний

Исходя из определения, можно выделить *несбалансированные критические состояния* – близкие к падению, и *травмирующие* – близкие к повреждению механизма, но не приводящие к выходу из строя. Концепция строится на том, что вводится «предпадение» вместо простого штрафа. В результате осуществляется переход от набора порогов к формальному множеству критических состояний. Введение множества критических состояний реализуется в виде аппроксимации границы устойчивости системы, применяемой для ограничения поведения стратегии. Производится явное моделирование границы устойчивости, используемое в обучении агента.

Рассмотрим шагающего робота, смоделированного в виде Марковского процесса принятия решений:

$$(S, A, P, R, \gamma), \tag{2.13}$$

где $s_t \in S$ – состояние, $a_t \in A$ – действие, $s_{t+1} \approx P(s_t, a_t)$ – динамика перехода, $R(s_t, a_t)$ – функция вознаграждения, γ – коэффициент обесценивания (дисконтирования).

Стратегия π_θ оптимизируется посредством обучения с подкреплением, причём $a_t \approx \pi_\theta$. Произведём разделение пространства состояний на три отдельные части:

$$S = S_{safe} \cup S_{critical} \cup S_{fail}, \tag{2.14}$$

где S_{safe} – множество устойчивых состояний, S_{fail} – неустойчивые состояния, приводящие к падению или повреждению, $S_{critical}$ – пограничные состояния, близкие к неустойчивым.

Определим скалярную функцию, измеряющую расстояние до множества неустойчивых состояний:

$$C : S \to R_{\geq 0}, \tag{2.15}$$

где S – множество состояний, $R_{\geq 0}$ – некоторое множество неотрицательных действительных чисел.

Функция $C(s)$ принимает значения из диапазона $[0,1]$ и характеризует меру близости к потере устойчивости. Чем меньше значение, тем безопаснее состояние.

Множество критических состояний определяется, как:

$$S_{critical} = \{s \in S_{safe} \mid C(s) \geq C_{crit}\}, \tag{2.16}$$

где $C(s)$ – функция близости к неустойчивости, C_{crit} – порог критичности, показывающий расстояние до неустойчивых состояний, относительно которого состояние считается принадлежащим множеству критических.

Другими словами, критическое состояние можно определить также как:

$$S_{critical} = \partial S_{safe} \tag{2.17}$$

Функция близости к неустойчивости может быть задана эвристически на основании показаний имеющихся сенсоров, либо аппроксимироваться посредствам нейронной сети. Рассмотрим эвристическое определение функции $C(s)$:

$$C(s) = \sum_{i=0}^{N} \omega_i C_i(s), \tag{2.18}$$

где $C_i(s)$ – источник риска с индексом i, ω_i – весовой коэффициент важности i-ого риска, N – количество рисков.

Под источником риска понимается некоторая скалярная функция C, коррелирующая с риском потери устойчивости или повреждения.

На основе данных датчика гироскопа-акселерометра и датчиков касания конечностей земли и соударения, она примет вид:

$$C(s) = \omega_1 \cdot C_{imu}(s) + \omega_2 \cdot C_{contact}(s) + \omega_3 \cdot C_{collision}(s), \tag{2.19}$$

где $\omega_1, \omega_2, \omega_3$ – весовые коэффициенты, задающие важность источника риска, C_{imu} – риск потери устойчивости, $C_{contact}$ – риск потери контакта конечностей с поверхностью земли, $C_{collision}$ – риск самосоударения (обратное расстояние между звеньями).

Риск потери устойчивости C_{imu}, определяется, как:

$$C_{imu}(s) = |\theta_{roll}| + |\theta_{pitch}| + \alpha|\omega|, \tag{2.20}$$

где θ_{roll} – угол крена (наклон вбок), θ_{pitch} – угол тангажа (наклон вперёд), ω – угловая скорость (рад/с), α – коэффициент влияния угловой скорости.

В риске C_{imu} углы θ демонстрируют наклон, а ω показывает, насколько быстро происходит падение. В основном существуют два случая: робот наклонён, но стоит и робот быстро падает. Второй вариант является критичным. Коэффициент α предназначен усилить значение величины скорости на риск. Риск потери контакта конечностей с поверхностью земли $C_{contact}$, определяется как:

$$C_{contact} = \sum \left\| \upsilon_{foot} \right\| \cdot 1_{contact}, \tag{2.21}$$

где υ_{foot} – скорость стопы или конца конечности.

Физический смысл заключается в том, что если конечность касается поверхности земли и при этом движется, то это означает наличие проскальзывания, что повышает вероятность потери устойчивости.

Риск самосоударения $C_{collision}$ определяется в виде:

$$C_{collision} = \sum \frac{1}{d_{ij}}, \tag{2.22}$$

где $d_{ij} = \left\| p_i - p_j \right\|$, p_i, p_j – позиции звеньев.

В результате C_{imu} фиксирует наклон и падение, $C_{contact}$ – скольжение, $C_{collision}$ – самосоударения.

Базовая задача обучения с подкреплением имеет вид:

$$E_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right] \to \max \tag{2.23}$$

Исходя из концепции критических состояний и функции близости к неустойчивости, задача примет следующий вид:

$$\begin{aligned} &E_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right] \to \max \\ &E_{\pi_\theta}\left[C(s_t)\right] \le \delta \end{aligned}, \tag{2.24}$$

где δ – порог риска (задаётся экспериментально).

Поскольку алгоритмы обучения с подкреплением не способны напрямую работать с ограничениями, а только максимизируют

вознаграждение, следует преобразовать задачу условной оптимизации в безусловную, путём создания Лагранжиана:

$$L(\theta,\lambda)=E_{\pi_\theta}\left[\sum_{t=0}^{T}\gamma^t r(s_t,a_t)\right]-\lambda\left(E_{\pi_\theta}[C(s_t)]-\delta\right) \tag{2.25}$$

Параметр λ – множитель Лагранжа, варьирует размер штрафа, при этом возрастает в случае нарушения ограничения и снижается при соблюдении. При формировании вознаграждение, данный штраф учитывается следующим образом:

$$r_t^{'}=r_t-\lambda C(s_t) \tag{2.26}$$

Множитель Лагранжа является настраиваемым параметром. Его обновление производится следующим образом:

$$\lambda=\max\left(0,\lambda+\beta\left(E[C(s)]-\delta\right)\right), \tag{2.27}$$

где β – скорость обучения, $\max(0,\bullet)$ – операция максимума, необходимая для обеспечения условия $\lambda\geq 0$.

Формула (2.27) получается следующим образом. Оптимизационная задача такова, что требуется максимизация суммарного дохода за эпизод при минимизации рисков возникновения критических состояний, то есть:

$$\max_{\theta}\min_{\lambda\geq 0}L(\theta,\lambda) \tag{2.28}$$

Из формулы (2.25) получается градиент:

$$\frac{\partial L}{\partial \lambda} = -\left(E[C(s)] - \delta\right) \tag{2.29}$$

Поскольку минимизация производится по параметру λ, то получим:

$$\lambda = \lambda - \beta \frac{\partial L}{\partial \lambda} \tag{2.30}$$

В результат, подставив значение градиента из (2.29) в формулу (2.30) получатся формула (2.27). Поскольку в обучении с подкреплением нет точного определения математического ожидания, необходимо воспользоваться оценкой $E[C(s)]$. Её практически можно получить по траектории (средний риск за эпизод):

$$\hat{C} = \frac{1}{T} \sum_{t=1}^{T} C(s_t), \tag{2.31}$$

где T – длина эпизода, s_t – состояние внутри одной траектории, или по набору траекторий:

$$\hat{C} = \frac{1}{N} \sum_{i=1}^{N} C(s_i), \tag{2.32}$$

где N – общее число состояний в наборе траекторий.

Сигнал завершения эпизода соответствует следующей функции:

$$end = \begin{cases} s_t \in S_{fail} \\ C(s_t) > C_{\max} \end{cases}, \tag{2.33}$$

где $C_{\max}$ – жёсткий порог критичности.

Параметр $C_{\max} \in [0.1]$ и выбирается опытным путём, например определением значения $C(s)$ перед падением для случайной политики или на ранней версии агента.

В результате, на основании функции $C(s)$ можно концептуально представить состояния следующим образом:

$$\begin{aligned} &S_{safe} : C(s) < C_{crit} \\ &S_{critical} : C_{crit} \le C(s) < C_{\max} \\ &S_{fail} : C(s) \ge C_{\max} \end{aligned} \tag{2.34}$$

Предлагаемая концепция не предназначена для устранения сбоев, но направлена на значительное снижение их частоты, что позволяет агенту предвидеть и избегать критических состояний.

2.6. Динамика многозвённого шагающего робота

Динамика шагающего робота, так же как и любого многозвённого механизма может быть описана стандартным уравнением движения Лагранжа:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{2.35}$$

где $q-$ вектор обобщённых координат (положение корпуса и положения суставов робота), $\dot{q}$ – вектор скоростей, $\ddot{q}$ – вектор ускорений, $M(q)-$ матрица инерции системы (масс узлов робота), $C(q,\dot{q})\dot{q}$ – кориолисовы и центробежные силы, $G(q)-$ гравитационные силы, $\tau-$ вектор управляющих моментов в суставах.

Для шагающих роботов добавляются контактные силы, поскольку при контакте конечностей с землёй появляются силы реакции опоры. В результате уравнение динамики примет следующий вид:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau + J^T(q)F \tag{2.36}$$

где $q-$ вектор обобщённых координат (положение корпуса и положения суставов робота), $\dot{q}$ – вектор скоростей, $\ddot{q}$ – вектор ускорений, $M(q)-$ матрица инерции системы (масс узлов робота), $C(q,\dot{q})\dot{q}$ – кориолисовы и центробежные силы, $G(q)-$ гравитационные силы, $\tau-$ вектор управляющих моментов в суставах, $J(q)$ – якобиан точки контакта конечности, F – вектор сил реакции опоры.

На основании уравнений динамики (2.35) и (2.36) можно сделать вывод, что движение считается сбалансированным при наличии управления $\tau(t)$, гарантирующего расположения траектории состояния в допустимой области фазового пространства.

Поскольку в контексте данной работы используется безмодельное глубокое обучение с подкреплением, которое осуществляет принятие решений о выполняемых действиях на основании опыта, модель динамики (2.35) и (2.36) не применяется для реализации управления, а формируется на основании реальных опытных данных. Она требуется для случая обучения с подкреплением, основанного на модели или при реализации механизма планирования, при котором каждое действие сопоставляется с моделью, в результате чего определяется качество выбора действия, до его фактического исполнения. Блок планирования можно рассматривать в качестве дополнительного модуля, который может быть добавлен в дальнейшем к существующей системе управления. В контексте настоящей работы он не рассматривается. Стоит отметить используемые скоростные режимы движения робота. В исследовании акцент делается в большей степени на

решении проблемы многофункциональности агента и повышении «безопасности» обучении робота. В связи с этим, высокоскоростные паттерны не будут рассмотрены.

Важным аспектом является удержание баланса и введённая в контексте исследования концепция критических состояний, обеспечивающих «безопасное» обучение. При стандартном подходе использования глубокого обучения с подкреплением баланс не определяется, а приобретается в процессе настройки агента, что сопряжено с травматизацией и частыми падениями в процессе обучения «напрямую» в роботе. Учёт концепции критических состояний в процессе управления обеспечивает обучение балансу, но без потери равновесия и травматизации. Для определения сбалансированных критических состояний требуется наличие специального математического аппарата. На практике физический баланс определяется относительно опорного многоугольника, представляющего поверхность, построенную из точек фактического соприкосновения конечностей робота с поверхностью земли. Для определения сбалансированности движения и сбалансированных критических состояний требуются:

– знание прямой кинематики конечностей;

– определение факта непосредственного контакта конечности с поверхностью земли;

– показания датчика гироскопа-акселерометра (imu);

– построение опорного многоугольника;

– формирование критериев устойчивости.

Инерциальные параметры робота, такие как ориентация корпуса, угловые скорости и линейные ускорения определяются посредствам датчика гироскопа-акселерометра. Для определения фактических значений величин углов сочленений требуются энкодеры. В контексте настоящей работы делается допущение о том, что приводы реализуют перемещение без погрешности, в результате чего угол фактического отклонения будет

соответствовать задаче управления. Это позволяет представить величины углов в виде вектора, без использования дополнительных датчиков. Необходимо фиксировать факт касания конечности с землёй, что обеспечивается системой микропереключателей, установленных на ступнях механизма.

Для построения математического обеспечения контроля баланса необходимо определить:

– системы координат;

– прямую задачу кинематики для получения координат опорных конечностей;

– определить способ построения опорной плоскости;

– определить контрольные характеристики, необходимые для определения баланса и сформировать на их основе критерии устойчивости;

– переформулировать критерии устойчивости в соответствии с концепцией критических состояний;

В контексте киберфизической системы мобильного шагающего робота рассматривается следующая общая структура координатных систем:

– F_W (World Frame) – мировая система координат (фиксированная в окружающем мире);

– F_B (Base Frame) – система координат корпуса робота (движется совместно с корпусом робота);

– F_{L_i} (Leg Root Frame) – система координат i-ой конечности робота (в точке крепления);

2.7. Определение системы координат и критериев устойчивости движения

Для формирования математического обеспечения определим:

- глобальную систему координат;

- базовую систему координат относительно корпуса робота;
- локальную систему координат конечности;
- прямую кинематику шагающего робота;
-построение опорной плоскости;
- контрольные характеристики, необходимые для определения баланса; критерии устойчивости движения.

Решим последовательно каждую из этих задач.

Глобальная система координат F_W представляет собой некоторую неподвижную инерциальную координатную систему, у которой ось Z направлена вверх, а оси X и Y лежат в плоскости земли (правая) [118]. В процессе формирования управления роботом она необходима, поскольку только относительно неё определяется отклонение центра масс и наклон корпуса. Баланс определяется относительно опорной плоскости, которая рассматривается в плоскости земли, которая, в свою очередь, определяется только в контексте глобальной системы F_W. Глобальная система координат выбирается произвольным образом. Возможны следующие варианты:

– зафиксировать начало координат системы F_W в начальном положении корпуса;
– использовать датчик гироскопа-акселерометра;
– зафиксировать начало в точке опоры первой поставленной ноги.

Другими словами, F_W начинается в некоторой точке-якоре. Она необходима не для «позиционирования в мире» (координаты GPS), а для реализации следующих целей:

– определение направления гравитации;
– вычисление точки нулевого момента (zmp);
– построение опорного многоугольника;
– оценка наклона корпуса робота (тангаж, крен, рыскание).

Глобальная система координат F_W – это математическая поверхность, относительно которой измеряется устойчивость.

Базовая (локальная) система координат F_B (Base Frame) представляет собой координатную систему, жёстко связанную с корпусом робота, являющуюся базой для его кинематики [118]. Она не зависит от GPS и перемещается совместно с роботом (поступательное и вращательное движение). Её начало обычно расположено в геометрическом центре корпуса. Это может быть середина туловища, центр прямоугольника корпуса или фиксированная точка конструкции механизма. Оси направлены следующим образом: ось X_B обозначает движение вперёд, ось Z_B направлена вверх, а Y_B выбирается таким, образом, чтобы система координат была правой (удовлетворяла правилу правой руки). В результате, система F_B фиксируется на корпусе робота, конечности описываются относительно её центра, баланс рассматривается на основе показаний датчиков касания и IMU. Глобальная система координат используется исключительно для реализации навигации.

Для определения локальной системы координат конечности для каждой конечности задаётся собственная система координат, начало которой выбирается в точке механического крепления конечности к корпусу [118]. Оси располагаются следующим образом: ось Z_{L_i} совпадает с осью первого сустава, ось X_{L_i} направлена «вниз по ноге» в нулевой конфигурации, а ось Y_{L_i} выбирается таким образом, чтобы система была правой (удовлетворяла правилу правой руки). Во многих теоретических описаниях $F_{L_i} = F_{H_i}$. Данная система координат вводится для обеспечения модели модульностью, поскольку становится возможным рассмотрение конечности как атомарной единицы.

В контексте настоящей работы используются следующие условия взаимодействия координатных систем:

– глобальная система координат рассматривается относительно показаний датчика гироскопа-акселерометра (imu);

– корпус робота представляет собой свободное тело;

– все конечности рассматриваются относительно локальной системы координат F_B (локальная кинематика);

– баланс определяется в глобальной системе координат F_W на основе показаний датчика гироскопа-акселерометра (IMU), гравитационного вектора и позиции стоп.

Каждая система координат рассматривается относительно системы более высокого уровня, переход между которыми представляет собой фиксированное преобразование T [118]. Для построения опорного многоугольника необходимо выполнить определение координат точек соприкосновения педипуляторов с поверхностью земли. Поверхность земли относится к глобальной системе координат.

Для каждого сустава существует преобразование, зависящее от угла поворота θ:

$$T_{parent,child}(\theta) \tag{2.37}$$

Если звено имеет родителя p, то его положение в мировой системе рекурсивно вычисляется, как:

$$T_{W_i} = T_{Wp} T_{pi}(\theta) \tag{2.38}$$

В современной робототехнике используют группу преобразований SE(3) или Special Euclidean Grope in 3D – это специальная евклидова группа в трёх измерениях, представляющая множество всех возможных жёстких преобразований в трёхмерном пространстве [118]. Под жёсткими преобразованиями понимаются такие, которые сохраняют расстояния, углы и

не деформируют объект движения. К ним относятся вращение, перенос или их комбинация. SE(3) представляет собой пространство всех возможных поз робота. Любая поза включает позицию и ориентацию в пространстве. Элементом SE(3) группы является матрица, составленная из матрицы вращения R и вектора перемещения p:

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}, \tag{2.39}$$

где R – матрица вращения, p – вектор переноса.

При решении задач кинематики часто можно встретить использование классического подхода с применением параметров Денавита-Хартенберга (DH). Его основными недостатками являются привязанность к последовательной структуре, неудобство в применении для манипуляторов со сложной геометрией и плохая масштабируемость. В контексте данной работы, главным критерием построения системы управления является универсальность, которую сложно обеспечить вышеупомянутым методом. В настоящее время существует современный способ описания кинематики робототехнических систем, который не зависит от количества суставов и конфигурации робота, идеально подходит для n-ногих систем и легко расширяется на динамику. Это подход «Произведение экспонент» (Product of Exponentials, PoE) [118].

Основной идеей PoE является представление любой последовательности суставов в виде последовательности винтовых движений. Движение твёрдого тела описывается дифференциальным уравнением $\dot{T} = [S]T$, решением которого является $T = e^{[S]\theta}$. Экспонента представляет собой интеграл мгновенного движения – позу робота. Каждый сустав создаёт элемент SE(3) $e^{[S_i]\theta_i}$, а вся кинематика конечности представляется в виде произведения соответствующих экспонент:

$$T(q) = e^{[S_1]\theta_1} e^{[S_2]\theta_2} \ldots e^{[S_n]\theta_n} M \,, \tag{2.40}$$

где S_i – винтовая ось i-ого сустава (Twist S), θ_i – угол i-ого сустава, M – поза конечности при нулевых углах (положение стопы когда все углы равны 0).

Формула (2.40) описывает кинематику педипулятора относительно его локальной системы координат T_{L_i}. Пусть T_{WE}^i – положение стопы i-ой конечности в мировой системе координат, T_{WB} – положение базовой системы координат относительно мировой, T_{BL}^i – положение системы координат i-ой конечности в базовой системе робота, то кинематика для i-ого педипулятора имеет следующий вид:

$$T_{WE}^i = T_{WB} \cdot T_{BL}^i \cdot T_{L_i} \tag{2.41}$$

Подробнее ознакомиться с деталями использования группы SE(3) и подхода «Произведение экспонент» можно в Приложении D.

Опорный многоугольник можно сформировать при помощи простого и надёжного алгоритма монотонной цепи Эндрю (Andrew’s Monotone Chain) [72], реализующего построение выпуклой оболочки множества точек на плоскости. Определение конечностей, контактирующих с поверхностью земли возможно при помощи микропереключателей, расположенных на стопах. Координаты точек соприкосновения определяются на основе прямой кинематики, рассмотренной в предыдущем разделе.

Среди формальных критериев удержания баланса роботом выделяются:

– расположение проекции центра масс (Center of Mass, CoM) на плоскости опоры;

– проекция точки нулевого момента (Zero Moment Point, ZMP) на плоскости опоры;

– точка захвата (Capture Point, CP).

Для квазистатических режимов движения (медленных) используется подход, на основе проекции центра тяжести. Применение ZMP и точки CP актуальны для динамических режимов передвижения, причём CP позволяет определять баланс при движении на высоких скоростях. Детали определения координат и скорости центра, координат точки нулевого момента и точки захвата представлены в Приложении Е.

Точка центра масс применяется для определения статической устойчивости. Баланс в динамике определяется при помощи точек ZMP и CP. Точка нулевого момента характеризует текущее состояние робота, в то время как точка захвата предсказывает баланс «в будущем». Если ZMP – это текущее положение точки опоры, то CP – это положение, в котором она должна находиться на следующем шаге. ZMP применяется при медленной ходьбе, реализации переноса веса между ногами. При больших скоростях, например, при беге, динамической ходьбе или прыжках используется точка CP. Её применение необходимо для формирования реакции на сторонние толчки. Для определения сбалансированности движения точки ZMP и CP незаменимы и используются совместно. На основании точек центра масс, нулевого момента и захвата определяются следующие критерии устойчивости:

– **движение является устойчивым, если проекция точки центра масс и точка zmp находятся внутри плоскости опоры;**

– **движение является устойчивым, если существует возможность на следующем шаге установить конечность так, чтобы точка захвата располагалась внутри области достижимых шагов.**

Если применить данные критерии в качестве жёстких ограничений, то робот не сможет реализовывать динамическую ходьбу (бег, прыжки),

поскольку будет «слишком осторожным», ухудшится обучение. Это связано с тем фактом, что динамически устойчивые движения способны нарушать эти условия. Поэтому, для учёта критериев устойчивости их следует включить в функцию функция близости к неустойчивости:

$$C(s) = \omega_1 \cdot C_{imu}(s) + \omega_2 \cdot C_{contact}(s) + \omega_3 \cdot C_{collision}(s) + \omega_4 C_{balance}, \quad (2.42)$$

где $\omega_1, \omega_2, \omega_3, \omega_4$ – весовые коэффициенты, задающие важность источника риска, C_{imu} – риск потери устойчивости, $C_{contact}$ – риск потери контакта конечностей с поверхностью земли, $C_{collision}$ – риск самосоударения (обратное расстояние между звеньями), $C_{collision}$ – риск выхода точек ZMP и CoM из области опорного многоугольника.

Риск $C_{collision}$ можно определить через расстояние d_{edge} от соответствующих точек p_{CoM} и p_{ZMP} до границы опорного многоугольника P:

$$C_{collision} = \begin{cases} 0, & p \in \mathrm{P}, d_{edge} >> d_{\max} \\ 1 - \dfrac{d_{edge}}{d_{\max}}, & p \in \mathrm{P} \\ 1, & p \notin \mathrm{P} \end{cases} \quad (2.43)$$

где $d_{\max}$ – максимальное расстояние до границы опорного многоугольника.

2.8. Математическая формализация задачи управления интеллектуальным агентом, реализующим локомоцию шагающего робота средствами оптимизации

Теория обучения с подкреплением строится на принципе гедонизма, в результате чего агент обучается выбору действий, приводящих к получению максимального значения суммарного вознаграждения за эпизод

взаимодействия со средой [56]. Главная цель, стоящая перед агентом, заключается в поиске оптимальной политики управления, обеспечивающей максимизацию будущего вознаграждения. Большинство современных приложений, использующих обучение с подкреплением, включая настоящую работу, используют аппарат глубоких нейронных сетей для обеспечения аппроксимации функции ценности (для реализации ценностных алгоритмов, например DQN) или стратегии принятия решений (для градиентных алгоритмов). В результате агент представляет собой глубокую нейронную сеть, а для обеспечения максимизации суммарного дохода за эпизод происходит постоянная настройка его весовых коэффициентов θ. От качества этой настройки зависит качество принятия решений. Отсюда следует, что вектор весовых коэффициентов θ является вектором параметров управления. Для задач робототехники, наиболее предпочтительными алгоритмами глубокого обучения с подкреплением выступают градиентные методы, обеспечивающие аппроксимацию стратегии π_θ, а не ценностей состояний [18, 41, 56]. Каждый эпизод взаимодействия можно охарактеризовать траекторией принятия решений τ, состоящей из всех посещённых состояний, полученных вознаграждений и выбранных в каждый момент времени действий. Каждая траектория напрямую зависит от значений весовых коэффициентов θ. В результате, целевая функция агента имеет следующий вид:

$$J(\theta) = E_{\tau \sim \pi_\theta}\left[R(\tau)\right] \to \max, \tag{2.44}$$

где τ – траектория, порождённая агентом при следовании параметрической стратегии π_θ, θ – вектор параметров стратегии π_θ, R – функция вознаграждения, $E_{\tau \sim \pi_\theta}$ – математическое ожидание случайной величины суммарного дохода за эпизод.

Приводы, применяемые в робототехнике, имеют конструктивные ограничения по диапазону углов отклонения и характеризуются величиной минимально возможного шага. Например, среднестатистический сервопривод работает в диапазоне углов $[0^o, 180^o]$ с шагом 1^o. В процессе функционирования робот реализует локомоцию, используя лишь область допустимых значений углов. Всё, что выходит за её рамки, способно конструктивно привести к выходу из строя механизма. В результате возникает совокупность ограничений для каждого сочленения по диапазону рабочих значений. Пусть Θ_i представляет собой диапазон возможных значений i–ого сочленения, а ε_{θ_i} – минимально допустимое значение угла i –ого сочленения и ε'_{θ_i} – максимально допустимое значение угла i–ого сочленения. Тогда совокупность рабочих диапазонов приводов можно определить в следующем виде:

$$\varepsilon_{\theta_i} \le \Theta_i \le \varepsilon'_{\theta_i} \tag{2.45}$$

При выполнении локомоции помимо удержания равновесия требуется учитывать следование по определённой траектории. Это возможно обеспечить путём анализа углов тангажа – η, крена – μ и рысканья – σ, задающих по трём осям положение робота относительно центра его инерции. Для них также характерны рабочие диапазоны, к тому же зависящие от паттерна локомоции. Пусть L – множество всех локомоторных команд, выполняемых роботом. Тогда η'_l и η''_l – нижняя и верхняя границы рабочего диапазона углов тангажа для локомоторной программы l, μ'_l и μ''_l – крена, σ'_l и σ''_l – рысканья, такие, что:

$$\begin{aligned} &\eta_l' \le \eta \le \eta_l'' \\ &\mu_l' \le \mu \le \mu_l'' \\ &\sigma_l' \le \sigma \le \sigma_l'' \\ &l \in L \end{aligned} \tag{2.46}$$

Концепция критических состояний вносит ограничение в задачу обучения с подкреплением:

$$E_{\pi_\theta}\left[C(s_t)\right] \le \delta \tag{2.47}$$

С учётом всех возможных ограничений, включая критические состояния (2.47), и основной оптимизационной задачи, представленной в (2.44), решаемая задача управления сводится к оптимизации с ограничениями следующего вида [37]:

$$\begin{aligned} &J(\theta) = E_{\tau \sim \pi_\theta}\left[R(\tau)\right] \to \max \\ &E_{\pi_\theta}\left[C(s_t)\right] \le \delta \\ &\varepsilon_{\theta_i} \le \Theta_i \le \varepsilon_{\theta_i}' \\ &\eta_l' \le \eta \le \eta_l'' \\ &\mu_l' \le \mu \le \mu_l'' \\ &\sigma_l' \le \sigma \le \sigma_l'' \\ &l \in L \end{aligned} \tag{2.48}$$

Таким образом, было получено математическое обеспечение системы управления интеллектуальным агентом, отличающееся внедрением в агента возможности предсказания риска возникновения нестабильных состояний, основанной на выделенном множестве критических состояний, и обеспечивающее повышение безопасности обучения и стабилизацию функционирования управляемой киберфизической системы за счет внедрения в агента возможности предсказания риска возникновения

нестабильных состояний, основанной на выделенном множестве критических состояний.

2.9. Выводы

В рамках данного раздела была выполнена детализация поставленной задачи исследования. Реализована конкретизация характеристик объекта управления и окружающей среды. Сформировано математическое обеспечение системы управления, с точки зрения применения глубокого обучения с подкреплением, основанное на предложенной концепции критических состояний, обеспечивающее минимизацию рисков падения или повреждения механизма в процессе обучения и взаимодействия со средой. Рассмотрены основные требования к разрабатываемой системе управления. Введены требуемые допущения. Сформирован интерфейс системы управления на уровне чёрного ящика. Произведено описание динамики системы управления с точки зрения Марковских процессов принятия решений. Реализована формализация критических состояний. Введена функция риска, демонстрирующая удалённость текущего состояния от области критических состояний. Рассмотрено её внедрение в процесс обучения. Рассмотрена динамика шагающего робота в соответствии с использованием глубокого обучения с подкреплением. Определены системы координат робота. Введён универсальный математический аппарат решения прямой задачи кинематики. Рассмотрены контрольные характеристики, необходимые для определения сбалансированности движения. На их основе сформированы критерии устойчивости, которые были модифицированы с использованием введённой в контексте исследования концепции критических состояний. Рассмотрено построение универсальных ограничений для каждого типа шагающих платформ, отличающихся количеством действующих педипуляторов. На их основе произведена

формализация задачи управления интеллектуальным агентом, реализующим локомоцию киберфизической системы мобильного шагающего робота в форме задачи оптимизации с учётом наличия критических состояний.

ГЛАВА 3. АЛГОРИТМИЧЕСКОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ИНТЕЛЛЕКТУАЛЬНЫМ АГЕНТОМ. АРХИТЕКТУРА ГЛУБОКОЙ НЕЙРОННОЙ СЕТИ ИНТЕЛЛЕКТУАЛЬНОГО АГЕНТА

В рамках данной главы рассматривается алгоритмическое обеспечение, лежащее в основе разрабатываемой системы управления интеллектуальным агентом, реализующим локомоцию киберфизической системы мобильного шагающего робота, и представлена инновационная архитектура нейронной сети агента, позволяющая обеспечить его возможностью обучаться и выполнять несколько локомоторных паттернов. В начале главы рассматривается обобщённый алгоритм управления киберфизической системой мобильного шагающего робота с точки зрения обучения с подкреплением. Производится структурный анализ и структуризация системы управления на основе парадигмы объектно-ориентированного проектирования. Осуществляется разделение на основные элементы и блоки. Предлагается алгоритм ликвидации критических состояний, обеспечивающий фокус обучения на критической области. Далее, рассматривается предлагаемый подход к решению проблемы многофункциональности агента за счёт повышения репрезентативной способности его глубокой нейронной сети. Производится описание архитектуры глубокой сети и механизмов её функционирования. Рассматривается применение предлагаемой архитектуры к прогрессивным нейронным сетям. Демонстрируется структура агента и модифицированный алгоритм предсказания выхода сетью. Даются рекомендации по применению архитектуры. В заключении приводятся выводы к главе.

3.1. Алгоритмическое обеспечение системы управления интеллектуальным агентом, реализующим локомоцию киберфизической системы мобильного шагающего робота

В данном разделе главы рассматриваются разработанные алгоритмы, обеспечивающие функционирование системы управления интеллектуальным агентом, реализующим локомоцию киберфизической системы мобильного шагающего робота. Сначала обсуждается управление в общих чертах с учётом применения алгоритмов глубокого обучения с подкреплением. Далее в контекст управления добавляется введённая в предыдущей главе концепция критических состояний.

3.1.1. Обобщённый алгоритм управления интеллектуальным агентом

Взаимодействие робота с окружающей средой представляет собой непрерывный процесс. Агент осуществляет выбор действия из заранее определённого множества, на основе наблюдения среды в текущий момент времени, реализует его и получает обратную связь в виде сигнала вознаграждения и перехода среды в новое состояние [56]. В предыдущей главе было сказано, что любой непрерывный процесс в теории обучения с подкреплением сводится к эпизодическому путём введения горизонта T, представляющего собой искусственную границу, при достижении которой начинается следующий эпизод. Завершением эпизода также служит падение робота или любое другое его состояние, требующее стороннего вмешательства для возобновления процесса взаимодействия. В процессе задействованы два основных участника [56]: агент – система, основанная на алгоритме обучения с подкреплением, которая осуществляет принятие решения о выборе действия в конкретный момент времени и окружающая

среда – всё, что расположено вне агента. Выбор действия основывается на данных наблюдения состояния окружающей среды. Среда, в свою очередь, генерирует вознаграждение и изменяет своё состояние на новое. Этот процесс происходит до тех пор, пока робот не выйдет из строя или не будет отключён оператором. В большинстве симуляторов, работающих с задачами, построенными на алгоритмах обучения с подкреплением, существует сформированная модель окружающей среды в виде некоторой функции, зависящей от действия агента. В ответ на действие функция возвращает кортеж, содержащий следующее состояние, вознаграждение и сигнал завершения эпизода. Так же каждой среде характерна функция инициализации, которая переводит среду в некоторое начальное устойчивое состояние. Рассмотрим типовой алгоритм функционирования агента, основанного на глубоком обучении с подкреплением, применяемый в средах имитационного моделирования, представленный на рисунке 3.1 [31].

В самом начале создаётся объект среды, в которой будет функционировать агент. Заранее определяется некоторое число, характеризующее количество эпох взаимодействия со средой. Внутри каждой эпохи происходит считывание наблюдения состояния среды, принимается решение о действии и осуществляется получение награды и перехода среды в новое состояние с последующим накоплением опыта до тех пор, пока эпизод не будет завершён. После накопленный опыт используется для настройки агента, по результатам которой формируется новая итерация. Из рисунка 3.1 становится понятно, что агентом в данном алгоритме выполняются две основные операция – выбор действия и обучение. Всё остальное, включая считывание наблюдений, относится к окружающей среде.

Если имитационная среда обладает моделью окружающей среды, то есть её реакциями на воздействие с последующей генерацией награды, то в реальных условиях она полностью отсутствует и перед разработчиком агента стоит выбор: либо формировать её на основе полученного опыта с

последующим использованием, либо настраивать стратегию без модели. Для реальных приложений, к которым относится управление роботом, построение модели среды является дорогостоящей операцией.

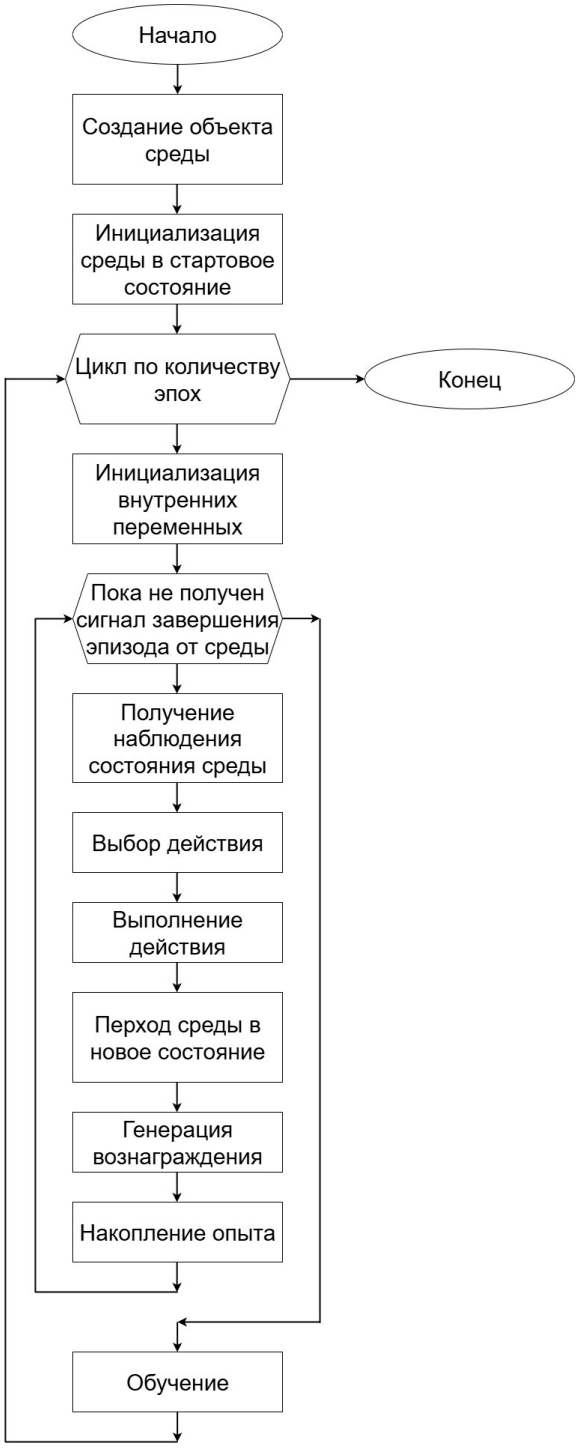


Рисунок 3.1 – Типовой алгоритм функционирования интеллектуального агента для среды имитационного моделирования

Поэтому оптимальным становится формирование стратегии выбора без модели. Несмотря на это, агент, в любом случае, должен получать наблюдения для реализации выбора и быть проинформированным о завершении эпизода, что необходимо для начала процесса обучения. Произведём структурное разделение системы управления на два основных блока: *агента* – блок принятия решений и *среду* – блок, отвечающий за считывание текущего наблюдения, генерацию сигнала завершения эпизода и формирование вознаграждения. В свою очередь, среду разделим на следующие блоки: *инициализации, генерации награды, считывания наблюдения, определения завершения эпизода, накопления опыта и выполнения действия*. Структурная схема системы управления, представлена на рисунке 3.2.

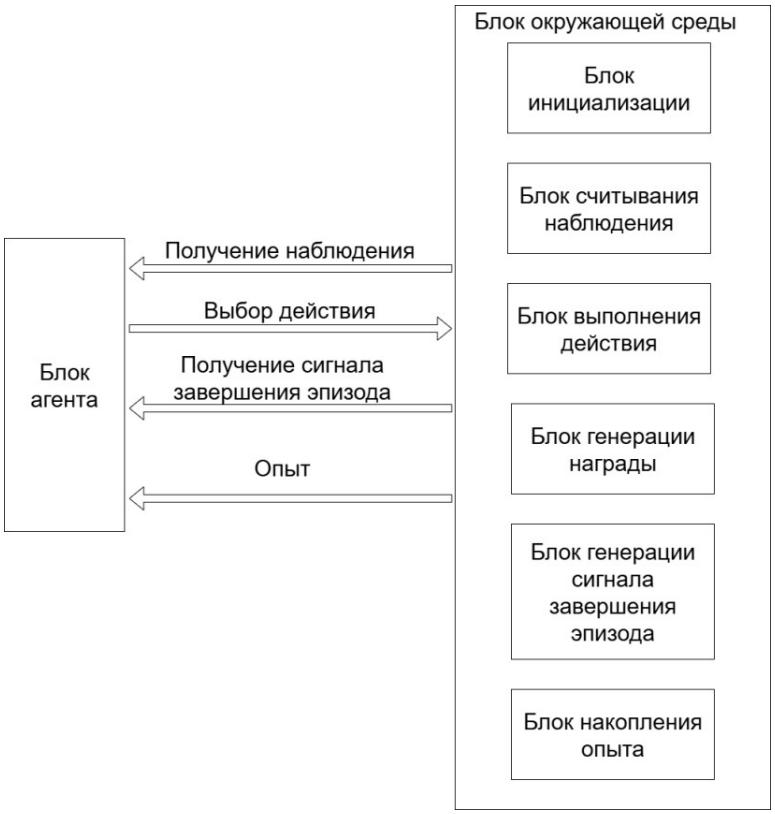


Рисунок 3.2 – Структурная схема системы управления интеллектуальным агентом

Рассмотрим каждый блок с точки зрения парадигмы объектно-ориентированного проектирования, что предполагает представление блока в виде некоторого объекта реального мира, обладающего функционалом с демонстрацией имеющихся связей. Для удобства, на высоком уровне будет использован привычный интерфейс взаимодействия, свойственный имитационным средам, при котором объект среды содержит два метода: инициализации, с возвратом стартового состояния (наблюдения) и выполнения шага взаимодействия, где в качестве возвращаемого значения будет передаваться следующее состояние (наблюдение), вознаграждение и сигнал завершения эпизода. Объект агента так же будет оснащён двумя методами: предсказания действия на основании поступающего на вход наблюдения и обучения, куда передаётся весь полученный на протяжении эпизода взаимодействия опыт.

Рассмотрим подробно блок считывания наблюдения. В его функциональные обязанности входит определение показаний сенсоров, имеющихся на борту робота и организация их в необходимый формат. Качество применяемых датчиков является критически важным, поскольку от их точности напрямую зависит процесс принятия решений и функционирование киберфизической системы робота в целом. В результате, данный блок представляет собой программно-аппаратный комплекс, реализующий низкоуровневую механику считывания сигналов сенсоров с последующей программной обработкой и форматированием их значений. Блок выполнения действия на вход получает текущее состояние (наблюдение) и выбранное действие. В результате, основной функционал заключается в низкоуровневой реализации перехода системы в новое состояние. Другими словами, данный блок представляет собой программно-аппаратный комплекс, который фактически выполняет перемещение робота в пространстве. Осуществляется единичный переход по графу состояний. Блок генерации вознаграждения как и блок выполнения действия принимает

текущее состояние (наблюдение) и новое, в которое был осуществлён переход. В результате обработки входных данных на выходе генерируется некоторое число, соответствующее ценности перехода. Внутри блока инициализации определяется некоторое устойчивое состояние механизма, которое устанавливается в результате стартовой активации. Блок инициализации является программно-аппаратным комплексом, определяющим устойчивое состояние механизма и устанавливающий в него конечности робота в момент включения питания. В блоке накопления опыта (буфер) формируется сохранение данных переходов из состояния в состояние в определённом формате, полученных на протяжении эпизода взаимодействия. После обучения буфер очищается и заполняется новыми данными. Блок генерации сигнала завершения эпизода на входе получает состояние (наблюдение), в которое был осуществлён переход робота, на основании которого производится многофакторный анализ позиционирования механизма в пространстве. В результате, на выходе генерируется требуемый бинарный сигнал, где 0 – эпизод продолжается и 1 – эпизод завершён. На рисунке 3.3 представлена схема взаимодействия блоков, входящих в состав блока окружающей среды.

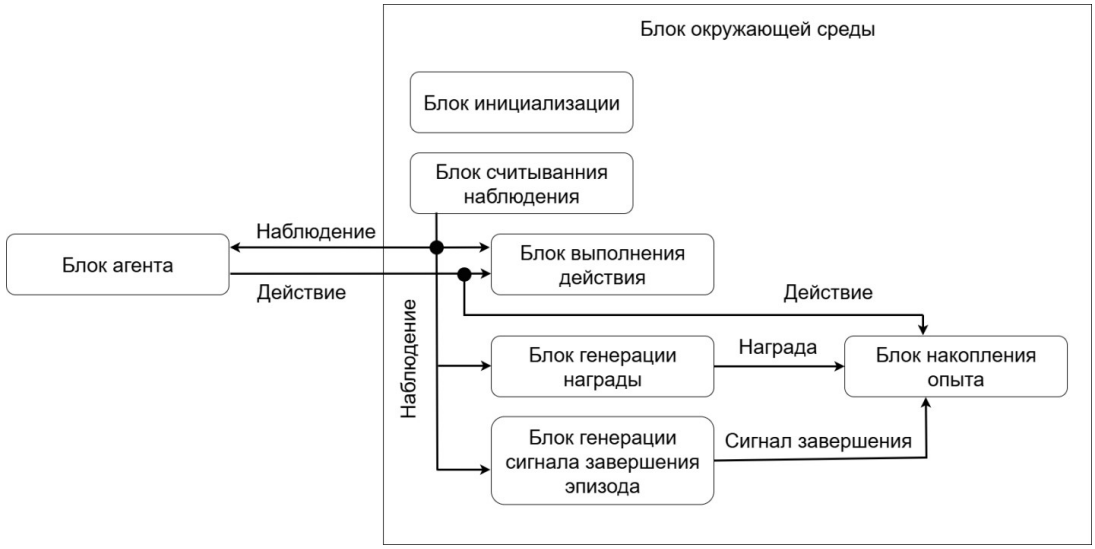


Рисунок 3.3 – Схема взаимодействия блоков окружающей среды

3.1.2. Алгоритм ликвидации критических состояний

Главной проблемой применения обучения с подкреплением для управления интеллектуальным агентом, реализующим локомоцию мобильного шагающего робота непосредственно на основе физического механизма киберфизической системы является генерация состояний, способных вывести из строя или опрокинуть дорогостоящий механизм. По этой причине наибольшее предпочтение среди разработчиков подобного типа систем отдаётся предварительной настройке агента в имитационной среде. В предыдущей главе была введена концепция «критических» состояний, при которых робот ещё сохраняет устойчивость, но находится на грани падения или близок к травматизации узлов и подразделяющихся на несбалансированные и травмирующие состояния [32]. Их применение предназначено для обеспечения минимизации неустойчивых состояний киберфизической системы, как в процессе обучения, так и при непосредственном функционировании.

В биологических системах любая травма ассоциируется с болевым ощущением, величина которого характеризует масштаб полученного ущерба. С другой стороны, сигнал боли информирует организм о недопустимости выполняемого действия, тем самым ограничивая диапазон возможных движений и положений конечностей. Для обеспечения робота подобным механизмом тактильной обратной связи достаточно фиксировать непосредственный факт касания конечностей или их частей в процессе локомоции. Техническая реализация определения несбалансированных и травмирующих критических состояний представлена в следующей главе настоящей работы.

На основании предлагаемой концепции был разработан алгоритм ликвидации критических состояний. Основная идея данного алгоритма заключается **в специфической реакции на возникновение критического**

состояния, которая заключается в «откате» после завершения эпизода в новое стартовое состояние, расположенное на удалении k шагов назад, следуя по траектории принятия решений предыдущего эпизода, начиная с обнаруженного критического состояния. Следующий эпизод начинается в соответствующем стартовом состоянии, в результате чего агент делает акцент на критической области, детально производя её изучение. Разработанный алгоритм ликвидации критических состояний имеет следующий вид:

Начало алгоритма

Инициализировать стратегию π

Инициализировать value-функцию V

Инициализировать модель риска C_ϕ

Инициализировать множитель λ

Задать параметры $\alpha, \beta, \delta, C_{crit}, C_{\max}, k$

Пока не выполнен критерий остановки обучения:

Начало эпизода

Если есть сохранённое критическое состояние:

Установить начальное состояние s *из буфера (откат на* k *шагов по траектории* τ*)*

Иначе:

Инициализировать s *стандартным образом*

Сбросить флаг $critical\_flag$ *= ЛОЖЬ*

Для каждого шага эпизода:

Выбрать действие a *согласно стратегии* π *в состоянии* s

Выполнить действие a

Получить новое состояние s_{next} *и награду* r

Вычислить предсказанный риск C_{pred} *с помощью модели* C_ϕ

Вычислить модифицированную награду:

$$r_{\text{mod}} = r - \lambda \cdot C_{pred}$$

Если произошло попадание в неустойчивое состояние или $C_{pred} \geq C_{\max}$ *:*

Установить флаг окончания эпизода $done$ *в ИСТИНА*

Иначе:

Установить флаг окончания эпизода $done$ *в ЛОЖЬ*

Если $C \geq C_{crit}$ *:*

Установить флаг наличия критического состояния $critical\_flag$ *в ИСТИНА*

Сохранить переход $(s, a, r_{\text{mod}}, C_{pred}, done)$ *в буфер*

Если флаг окончания эпизода $done$ *в значении ИСТИНА:*

Прервать цикл шагов

Обновить состояние:

$s = s_{next}$

Конец цикла шагов

Начало обновления моделей

Вычислить преимущества $advantage$ *и обесцененные доходы* $return$ *по буферу*

Обновить параметры стратегии π *и value-функции* V *(алгоритм PPO)*

Для каждого сохранённого состояния:

Определить метку y *:*

$y = 1$, *если в течение следующих* k *шагов было попадание в неустойчивое состояние*

$y = 0$ *иначе*

Обновить параметры модели риска $C\phi$ *по ошибке предсказания*

Вычислить средний риск C_{mean} *по эпизоду*

Обновить множитель λ *:*

$\lambda = \max(0, \lambda + \beta \cdot (C_{mean} - \delta))$

Конец обновления моделей

Начало процедуры сброса

Если $critical\_flag$ *= ИСТИНА:*

Выбрать состояние из буфера с откатом на k *шагов*

Использовать его как начальное для следующего эпизода

Иначе:

Использовать стандартную инициализацию

Очистить буфер

Конец процедуры сброса

Конец цикла обучения

Конец алгоритма

В сравнении с существующими аналогами, где обучение основано на поиске стратегий, не приводящих к падению или травматизации, алгоритм ликвидации критических состояний обучается избеганию состояний, способных привести к падению или травме. Другими словами, агент обучается не просто не падать, а избегать ситуаций, приводящих к падению или повреждению.

Когда предсказанное значение функции больше верхней границы области критических состояний, то есть $C_{pred} \geq C_{\max}$, производится установка флага конца эпизода, обучение на полученной траектории, «откат» на $t-k$ шагов назад. В качестве нового стартового состояния выбирается s_{t-k}. В результате происходит фокус на границе устойчивости. Обучение чаще происходит на состояниях $s \approx \partial S_{safe}$. Вместо длинных эпизодов получается много «почти плохих» коротких траекторий, что является более информативным.

Параметр k является центральным элементом алгоритма ликвидации критических состояний и отличительной особенностью в сравнении с

обычным алгоритмом обучения с подкреплением, который учится только на факте уже произошедшего падения. Параметр k задаёт насколько далеко назад во времени расположена причина обнаруженной опасности или падения. Обычный алгоритм обучения с подкреплением видит факт попадания в неустойчивое состояние, но не знает где, в какой момент времени, произошла ошибка. Особенность алгоритма ликвидации критических состояний заключается в том, что если падение или критическое состояние произошло в момент времени t, то проблема началась раньше. Именно для этого и вводится окно длиной k. Рассмотрим пример. Пусть имеется следующий эпизод: $t=0$, $t=1$ и $t=2$ – робот корректно перемещается, $t=3$ – наблюдается небольшая потеря равновесия, $t=4$ – ситуация ухудшается, $t=5$ – робот почти упал, $t=6$ – фактическое падение. Обычный алгоритм глубокого обучения с подкреплением видит, что падение произошло в момент $t=6$, но не знает, где началась ошибка. Возможно, настоящая причина была ещё на моменте $t=3$. Пусть для алгоритма ликвидации критических состояний задано $k=3$, тогда при падении в момент $t=6$, опасными состояниями считаются $t=3$, $t=4$, $t=5$ и $t=6$. При обнаружении критического состояния останавливается эпизод и производится «откат» на k шагов назад, поскольку текущее состояние уже «плохое». Если произвести перезапуск обучения из него, то агент сразу окажется в критической области. Поэтому берётся состояние на k шагов раньше и из него продолжается обучение. Концептуально, обычный алгоритм обучения с подкреплением учится на последствиях, в то время как алгоритм ликвидации критических состояний пытается определить область зарождения ошибки, а не только её финальное проявление. Параметр k – это не физическое время, а временной масштаб причинно-следственной связи между состоянием и будущим падением или повреждением.

Важным является определение параметра k и механизм «отката». Если задать параметр некорректно, то возможно переобучение на границе

устойчивости. В результате чего агент хорошо функционирует в области критических состояний и плохо в обычных состояниях. Возможна потеря разнообразия, которая заключается в отсутствии «естественных траекторий». Для ликвидации негативных последствий рекомендуется выбирать небольшое значение k. При слишком малом значении агент не успеет отреагировать и в скором времени упадёт или будет повреждён. Например, если $k = 1$, то тогда агент «смотрит» только на состояние прямо перед падением. При слишком большом значении теряется основной смысл алгоритма и введённого параметра. Например, при $k = 200$, почти вся траектория станет «опасной». В результате, получится слишком много ложных тревог, при которых рискованными будут считаться даже безопасные состояния. Для обеспечения разнообразия рекомендуется ввести вероятность реализации «отката» p_{reset}. Если случайное число меньше вероятности p_{reset}, то «откат» производится. В противном случае – нет. Так же рекомендуется смешивать обычное обучение и обучение с использованием алгоритма ликвидации критических состояний. В результате предлагается стратегия сброса, ориентированная на границу устойчивости, которая перезапускает эпизоды с состояниями, предшествующими критическим переходам, что позволяет эффективно осваивать стратегии восстановления. В отличие от стандартных подходов, основанных на прекращении обучения, предлагаемый алгоритм предполагает повторную выборку околокритических состояний для повышения устойчивости политики в области границы стабильности.

Таким образом, было получено алгоритмическое обеспечение, позволяющее агенту безопасно обучаться на взаимодействии с окружающей средой посредством ликвидации критических состояний и обеспечивать более безопасный перенос агента с последующей безопасной настройкой в реальной среде.

3.2. Построение многофункционального агента

Практическое применение глубокого обучения с подкреплением в контексте построения систем искусственного интеллекта и интеллектуальной робототехники связано с проблемой обеспечения множественности форм поведения в контексте одного агента. Агент, строящийся на алгоритме глубокого обучения с подкреплением, представляет собой глубокую нейронную сеть, способную аппроксимировать только одну локомоторную программу [128]. В случае смены программы сеть демонстрирует эффект забывания в попытке перестроиться под новые условия задачи. Это связано с её низкой репрезентативной способностью. Данный раздел главы посвящен решению проблемы расширения репрезентативной способности глубокой нейронной сети интеллектуального агента, основанного на глубоком обучении с подкреплением, обеспечивающей запоминание и реализацию нескольких локомоторных стратегий в контексте одной глубокой нейронной сети. Для реализации построения новой архитектуры, обеспечивающей многофункциональность, вводятся новые понятия нейронного кластера и нейронной группы [27, 33]. На основе последних достижений в области нейробиологии, были разработаны и использованы, в контексте предлагаемой архитектуры, переключающие нейроны, являющиеся искусственной аналогией нейронов базального ганглия, обеспечивающие переключение стратегий в рамках одной сети.

3.2.1. Архитектура глубокой нейронной сети интеллектуального агента

Рассматривая проблематику обеспечения агента возможностью запоминания нескольких локомоторных программ, становится очевидным, что этого невозможно достичь без изменения архитектуры полносвязной

нейронной сети. Обзор литературных источников по данной проблематике позволил заключить, что проблема забывания может быть решена путём выделения отдельной нейронной сети под каждую конкретную локомоторную задачу. Множество независимых модулей порождает следующую проблему – способ и механизм выбора модуля и связанная с этим настройка сети-модуля. Обращая внимание на примеры организации нервной системы биологических организмов, можно сделать вывод, что она составлена из фиксированного числа нервных клеток. При этом, несмотря на относительно небольшое количество нейронов, свойственных нервной системе насекомых или, к примеру, червей, она способна выполнять множество различных операций, включая локомоторные программы, с возможностью адаптации их паттернов под разные условия окружающей среды. Исходя из этого, становится очевидным существование способа организации нервной системы, способной при ограниченном количестве нервных клеток, связанных между собой, реализовать множество различных операций.

Введём определение нейронного кластера [27, 33]. Пусть задана некоторая совокупность нейронов, полностью связанных между собой. Данная совокупность называется *нейронным кластером*. Условно разделим кластер на несколько частей. Каждая часть представляет собой *нейронную группу*. Количество нейронных групп внутри нейронного кластера характеризует его *программную ёмкость* W. Под программной ёмкостью понимается количество программ, которое способен аппроксимировать нейронный кластер. Каждая группа ассоциируется с конкретной решаемой задачей, тем самым обеспечивая выполнение принципа «одна задача – одна сеть», исключающего проблему забывания. Особенность такого подхода заключается в том, что под кластером понимается одна полносвязная нейронная сеть, в контексте которой производится условное разделение обязанностей по решению задач между различными её частям, без

физического их разделения. Это аналогично функционированию головного мозга, где при выполнении разных операций происходит активация соответствующих им отделов. Активация нейронных групп производится поочерёдно, в соответствии с условиями решаемой задачи, как представлено на рисунке 3.4.

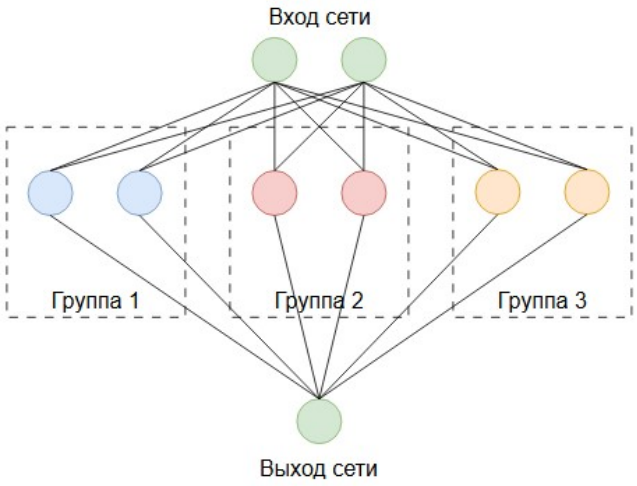


Рисунок 3.4 – Архитектура нейронной сети-кластера с двумя входами и одним выходом с программной ёмкостью $W = 3$

Использование нейронных групп решает задачу забывания, но порождает новую проблему, связанную с механикой управления выбором необходимой группы в конкретный момент времени. В свою очередь, данная проблематика может быть поделена на два основных вопроса: поиск способа взаимно однозначного кодирования нейронных групп и поиск способа селекции групп в соответствии с кодом. Исходя из того, что нейронный кластер характеризуется программной ёмкостью, которая является фиксированной величиной, для токенизации групп можно воспользоваться бинарным кодированием, при котором уникальный бинарный код p ставится в соответствие нейронной группе w. Например, для кластера с программной ёмкостью $W = 16$, можно использовать совокупность кодов, состоящих из

четырёх разрядов, для каждой из возможных команд. Управляющая команда селектора программы может быть подана как с вышестоящего уровня, так и напрямую, с пульта управления оператора.

Предлагаемая архитектура основывается на принципах функционирования биологических скоплений нейронов, в которых выполнение операций стимулируется посредством последовательных активаций и деактиваций различных участков одной и той же нейронной сети. Поступление кода некоторой группы стимулирует активацию её нейронов и производит деактивацию нейронов других групп. В контексте предлагаемой архитектуры данный механизм реализуется на принципе, близком к Dropout регуляризации – процедуры, случайным образом отключающей некоторые нейроны сети в процессе её обучения. В отличие от неё, предлагаемый способ активации и деактивации базируется на определении детерминированных активационных масок m, которые, аналогично кодам активации p, ставятся в соответствие каждой группе нейронов. Каждый разряд маски m соответствует конкретному нейрону слоя. Пусть задана некоторая функция $\Psi(p)$, ставящая в соответствие коду стратегии p некоторую битовую маску m, активирующую нейроны выбранной группы и деактивирующая все остальные. На рисунке 3.5 представлена архитектура однослойного нейронного кластера, использующего активационную маску m, в соответствии с функцией $\Psi(p)$.

Реализация двигательной программы сопряжена с выполнением множества операций головным мозгом. За их оркестрацию отвечает древняя, глубоко расположенная его часть – базальные ядра [87, 132]. В [87, 132] представлены результаты исследовательской работы, связанной с изучением генерации управляющих сигналов, посылаемых в двигательные центры головного мозга. В исследовании [87] отмечается, что функционирование базальных ганглиев сопоставимо с отлаженной системой светофоров, где каждый из них, в соответствии с запланированным действием, осуществляет

активацию или деактивацию выполнения. Такая форма поведения, свойственная базальным ганглиям, указывает на особую, основанную на движениях, систему кодирования, обладающую высокой детализацией [87, 132].

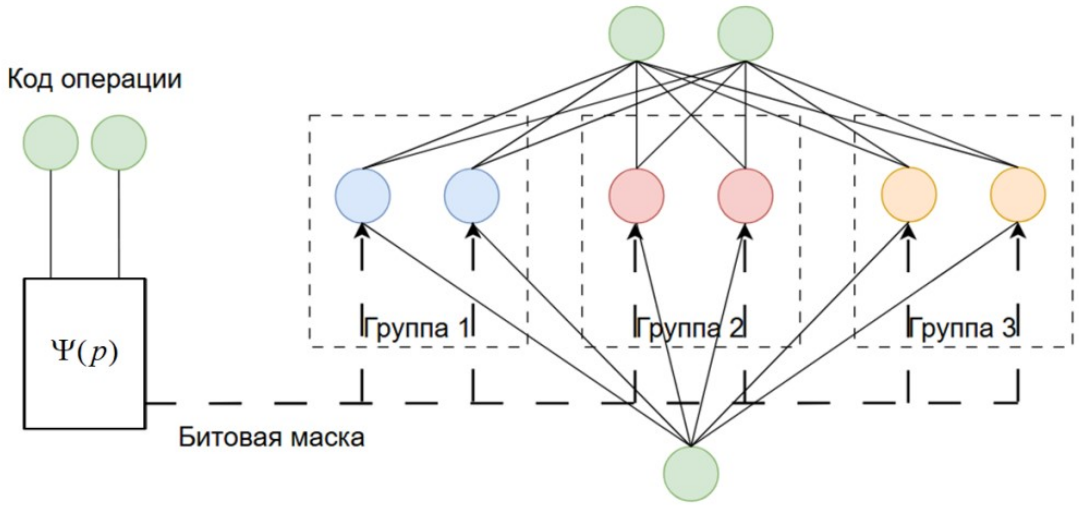


Рисунок 3.5 – Архитектура однослойного нейронного кластера с двумя входами и одним выходом, использующего генератор регулирующей маски

Рассматривая функцию $\Psi(p)$, прослеживается прямая аналогия с функционированием базальных ганглиев, рассматриваемых в работе [87]. Исходя из этого, следует, что $\Psi(p)$ является переключающим нейроном – искусственным нейроном базального ганглия, ставящим в соответствие коду стратегии p (равносильно движению) некоторую активационную маску m группы внутри кластера (эквивалент кодировки активации, генерируемой нейроном базального ганглия). В результате была получена архитектура однослойного нейронного кластера, которая отличается от существующих аналогов наличием переключающего нейрона – искусственного нейрона базального ганглия, обеспечивающего расширение репрезентативной способности полносвязной нейронной сети-кластера, посредством условного

разделения её на группы (в соответствии с локомоторными программами), активация которых осуществляется на основе активационных масок [27, 33]. Использование переключающих нейронов делает возможным построение сложной многослойной архитектуры, где каждому скрытому слою соответствует собственный переключающий нейрон. В совокупности, все переключающие нейроны глубокой сети образуют искусственный базальный ганглий. Использование искусственного базального ганглия с двумя искусственными базальными нейронами, на примере двухслойной архитектуры с двумя входами и одним выходом, представлено на рисунке 3.6.

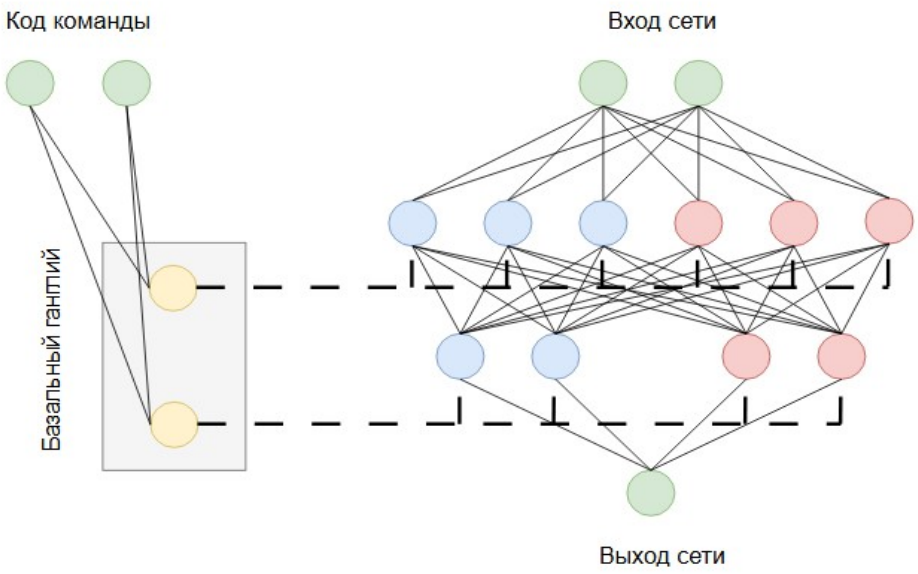


Рисунок 3.6 – Использование искусственного базального ганглия с двумя искусственными базальными нейронами на примере двухслойной архитектуры с двумя входами и одним выходом

Предложенная архитектура выполняет разделение полносвязной нейронной сети между задачами, обеспечивая их независимое запоминание без «потери памяти» при переключении агента между ними. Каждый

искусственный нейрон базального ганглия отвечает за активацию своей группы нейронов в слое, не пересекаясь с нейронами других групп. На рисунке 3.7 представлен пример функционирования искусственных нейронов базального ганглия, осуществляющих переключение между двумя независимыми группами полносвязной сети-кластера с двумя скрытыми слоями, двумя входами и одним выходом, где «0» обозначен не активный нейрон, «1» – активный, а «00» соответствует коду первой группы, «01» – коду второй группы.

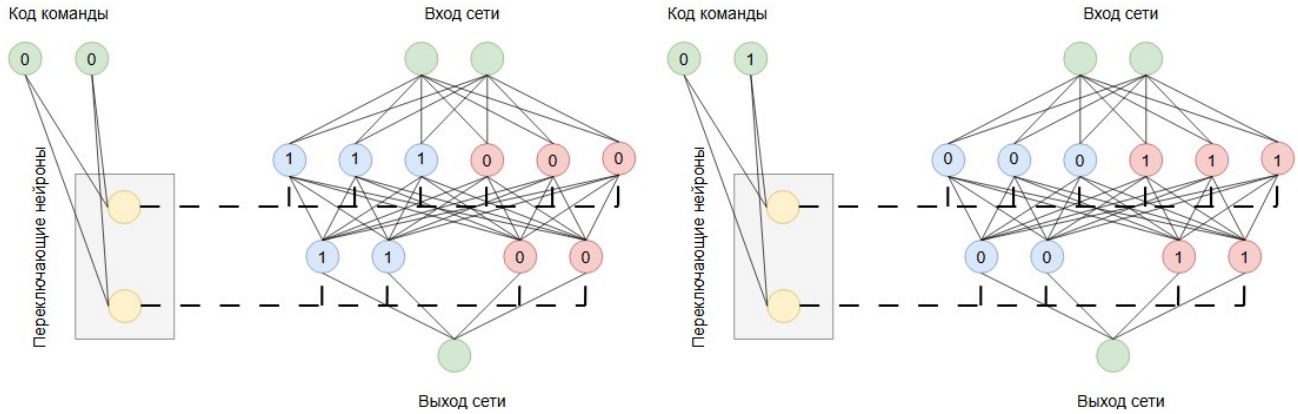


Рисунок 3.7 – Пример функционирования искусственных нейронов базального ганглия, осуществляющих переключение между двумя независимыми группами полносвязной сети-кластера с двумя скрытыми слоями

3.2.2. Прогрессивные нейронные сети

Полное отсутствие каких-либо связей между нейронными группами не всегда оказывает положительный эффект при настройке стратегий. В особенности это касается похожих действий. В работе [137], посвящённой прогрессивным нейронным сетям, отмечается положительный эффект от

применения боковых связей между несколькими отдельными нейронными сетями, называемыми «столбцами». Он заключается в лучшей сходимости, поскольку связи обеспечивают использование предыдущего опыта настроенных ранее столбцов. Решая проблему «забывания», прогрессивные нейронные сети не обладают механизмом селекции. Вопрос остаётся открытым. Данная задача может быть решена благодаря применению предлагаемой архитектуры, с использованием некоторых допущений. Без ограничения общности, рассмотрим прогрессивную нейронную сеть, состоящую из двух столбцов (программ) $k1$ и $k2$, каждый из которых представлен глубокой нейронной сетью с двумя скрытыми слоями, двумя входами и выходами, представленную на рисунке 3.8. Обозначим пунктирной линией боковые связи, объединяющие столбцы $k1$ и $k2$, в направлении от столбца $k1$ к столбцу $k2$.

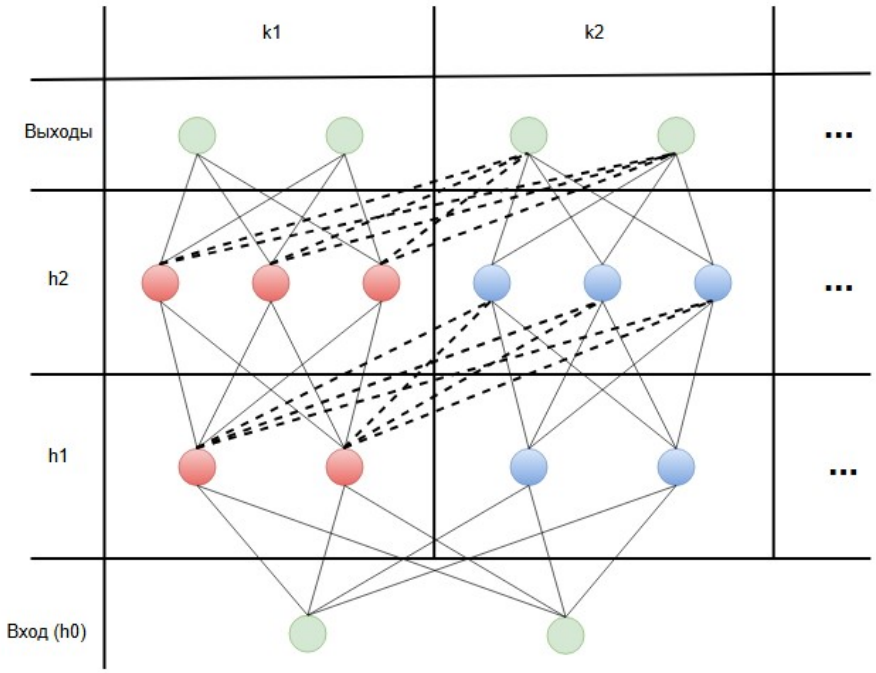


Рисунок 3.8 – Прогрессивная нейронная сеть, состоящая из двух столбцов (программ) $k1$ и $k2$ с боковыми связями

Спроектируем прогрессивную нейронную сеть-кластер, на базе предлагаемой архитектуры, с использованием переключающих нейронов, обеспечивающую запоминание множества стратегий с использованием предыдущих знаний, обладающую механизмом селекции столбцов.

Из рисунка 3.8 следует, что прогрессивная нейронная сеть является частично полносвязной, учитывающей «правые соединения» от столбца с индексом i до столбца с индексом $i+1$. Её можно получить из полносвязной сети, путём исключения «левых» связей. Ограничим соответствующую нейронную сеть, путём введения программной ёмкости W. Это означает, что сеть содержит W столбцов. Выполним добавление переключающего нейрона на каждый глубокий слой нейросети, включая выходной, как показано на рисунке 3.9.

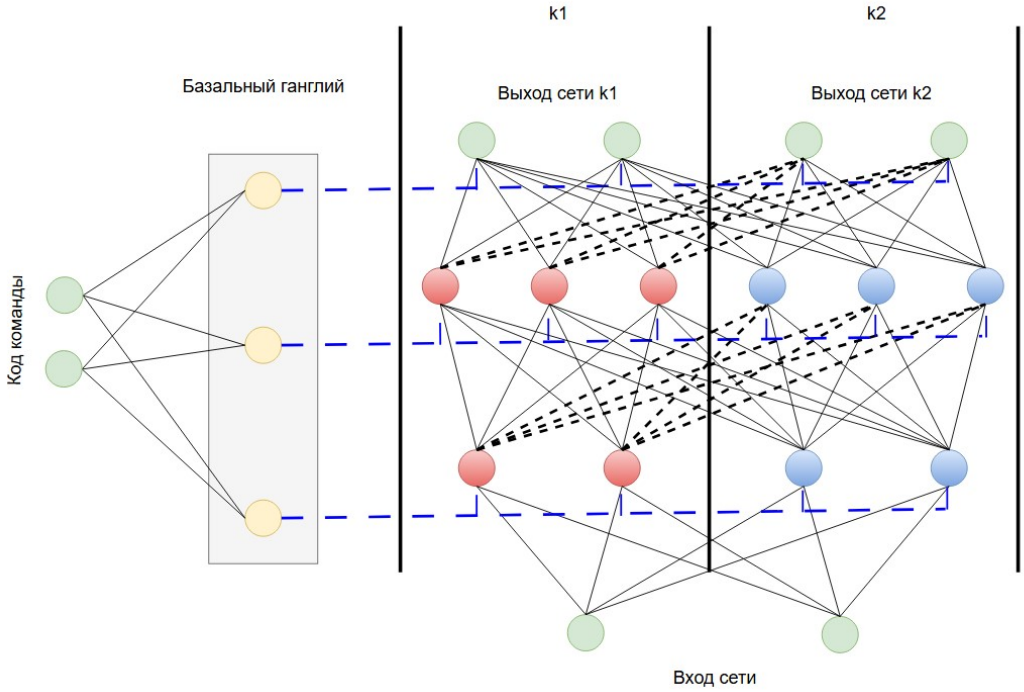


Рисунок 3.9 – Прогрессивная нейронная сеть, основанная на архитектуре полносвязного нейронного кластера с использованием переключающих нейронов

Управляющий нейрон соответствующего слоя выполняет генерацию активационной маски, которая «включает» требуемую нейронную группу в некоторый момент времени, согласно управляющему коду, подаваемому на вход нейронов искусственного базального ганглия. Использование предлагаемой архитектуры к прогрессивным нейронным сетям видоизменяет процесс предсказания выхода сетью, поскольку следует брать в рассмотрение исключительно «правые» соединения. Ликвидация «левых связей» производится на основе имеющихся активационных масок управляющих нейронов. Исходя из рисунка 3.8, изменённый алгоритм предсказания, в контексте прогрессивных нейронных сетей с использованием переключающих нейронов, имеет следующий вид:

Шаг 1: Выполнить расчёт выходов y_1 *слоя* h_1.

Шаг 2: Получить маску переключающего нейрона для слоя h_2.

Шаг 3: Согласно полученной маске для слоя h_2 *вычислить выходные значения для активных нейронов (вектор значений* y_2^1*) слоя* h_2 *по всем связям с нейронами слоя* h_1.

Шаг 4: Выполнить инверсию маски, полученной от переключающего нейрона для слоя h_2.

Шаг 5: Получить маску переключающего нейрона для слоя h_1.

Шаг 6: Выполнить инверсию маски, полученной от переключающего нейрона для слоя h_1.

Шаг 7: На основе инвертированных масок, полученных на шагах 4 и 6 вычислить выходные значения для активных нейронов слоёв h_1 *и* h_2 *– вектор значений* y_2^2.

Шаг 8: Сформировать рассчитанный вектор выходов слоя h_2 *по следующей формуле:* $y_2 = concat(y_2^1, y_2^2)$*, где concat – операция слияния векторов в один.*

Шаг 9: Получить маску переключающего нейрона для слоя выходов.

Шаг 10: Согласно полученной маске для слоя h_2 *вычислить выходные значения* y_3 *для активных нейронов выходного слоя по всем связям с нейронами слоя* h_2.

Данный алгоритм представлен схематически на рисунке 3.10.

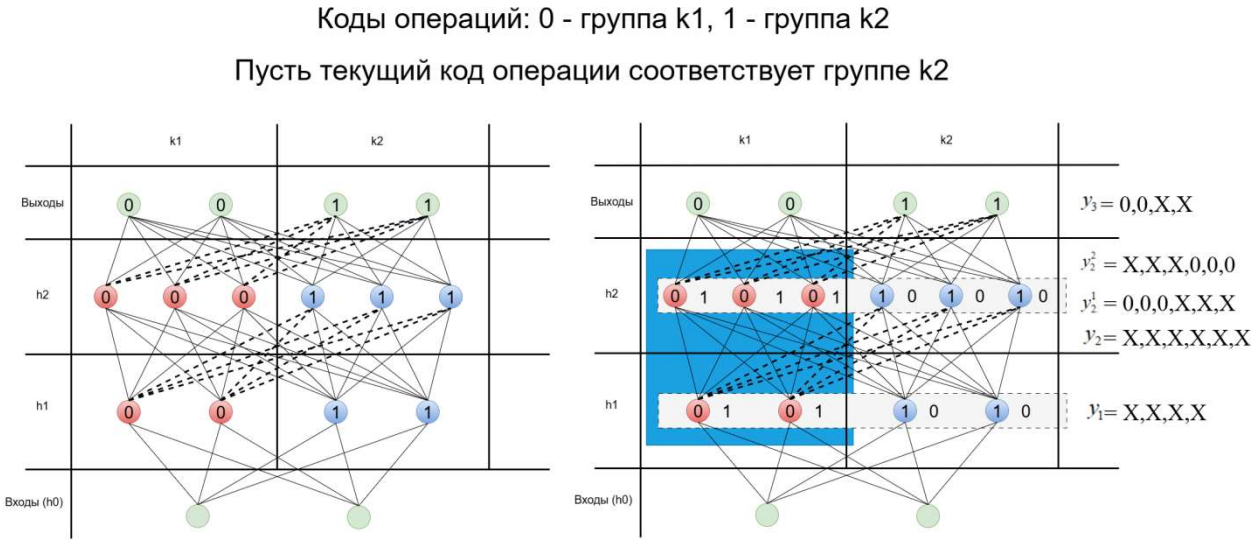


Рисунок 3.10 – Визуализация алгоритма

На рисунке слева представлены исходные данные, где активной является вторая группа k_2. Справа представлена визуализация алгоритма с указанием формата выходов соответствующих слоёв. Символ «X» обозначает некоторое значение выхода нейрона. Инвертированные значения масок указаны рядом с соответствующим нейроном. Выделенный блок соответствует шагу 8 алгоритма.

Обратное распространение ошибки выполняется аналогично случаю обучения сети, с использованием Dropout регуляризации, учитывая сгенерированные переключающими нейронами маски. Предложенный выше

алгоритм применим к нейронным сетям с любым количеством скрытых слоёв.

Для полноценного моделирования функционала прогрессивных нейронных сетей, посредством предлагаемой архитектуры необходимо игнорировать все группы, расположенные «справа» от активной и использовать исключительно связи «левых» групп. Данная процедура осуществляется на основе активационных масок путём игнорирования нулевых разрядов, расположенных справа от последнего разряда активной группы в процессе инверсии маски. Например, если маска имеет вид $m = [0,0,0,1,1,1,0,0,0,0]$, то инвертированная маска примет вид $m = [1,1,1,0,0,0,0,0,0,0]$.

Прогрессивные нейронные сети учитывают только дополнительные связи с группами, расположенными слева от текущей. В результате отсутствует связь с «правыми» группами, которая может содержать полезную дополнительную информацию. К примеру, ранее было изучено движение прямо по ровной поверхности, но позже, через несколько столбцов, изучается движение прямо по иной форме рельефа. Возможно, новая стратегия может содержать дополнительную информацию, которая способна улучшить или дополнить родственную стратегию движения прямо, изученную ранее. Логика обучения прогрессивных нейронных сетей такова, что сначала обучается первая группа, после вторая, но уже с боковыми связями, далее третья, также с дополнительными связями и так далее. Поскольку прогрессивные сети рассматриваются в совокупности с обучением с подкреплением, то после полного формирования перечня групп, некоторые могут подвергаться дополнительной настройке и учитывать только предшествующие стратегии. Если ликвидировать данное ограничение, то получится, что активная группа будет получать вспомогательную информацию от всех ранее изученных стратегий, тем самым учитывать дополнительную информацию.

В результате была получена архитектура, основанная на полносвязном нейронном кластере, управляемом переключающими нейронами, обладающая механизмом селекции программ (групп) и наличием боковых связей, унаследованных от прогрессивных нейронных сетей, способствующих ускорению процесса обучения. Применение переключающих нейронов обеспечивает нейронный кластер двумя режимами функционирования. Первый заключается в полном условном разделении кластера на группы. Второй, распространяет концепцию на прогрессивные нейронные сети, решая проблему селекции столбцов, которые, в контексте архитектуры, представляют нейронные группы, и использует преимущества наличия боковых связей, для ускорения обучения стратегий.

3.3. Генерация активационных масок управляющих нейронов

Ранее был подробно рассмотрен механизм функционирования переключающих нейронов без обсуждения способа генерации активационных масок, которые применяются для разделения групп внутри полносвязной сети-кластера. Данный вопрос поднимает фундаментальную проблему, существующую в теории глубоких нейронных сетей, которая заключается в отсутствии аппарата, позволяющего предсказать необходимую и достаточную структурную организацию сети, в соответствии с решаемой задачей. Большинство приложений основывается на принципе эмпирического перебора возможных вариантов структур глубокой сети, с последующей селекцией на основе требуемых критериев, таких как, например, качество решения задачи. Активационная маска генерируется переключающим нейроном и содержит в себе в виде активных разрядов (равных «1») количество используемых конкретной группой нейронов конкретного слоя. Исходя из этого, с учётом проблематики прогнозирования

требуемой структуры, следует, что формирование активационной маски строится на основании эмпирических опытов, в результате которых для некоторой задачи экспериментально, варьируя количество активных нейронов, подбирается необходимая и достаточная для группы размерность каждого глубокого слоя.

Следующим актуальным вопросом является способ объединения активных нейронов слоя так как их организация может быть хаотичной. Для решения этой задачи воспользуемся концепцией столбцов из теории прогрессивных нейронных сетей, где каждый новый столбец располагается справа от всех предыдущих. То есть, каждая группа нейронов постепенно, в процессе проектирования, будет добавляться к имеющимся группам справа. Такой способ организации позволяет обеспечить упорядоченность групп и возможность обучения и подбора необходимых и достаточных структур для каждой задачи в контексте полносвязной нейронной сети, но при этом рассматривать каждую группу отдельно от остальных. Поскольку нейронный кластер характеризуется программной ёмкостью W, такой подход позволит постепенно наращивать количество групп до достижения соответствующего параметра. В результате этого кластер будет полностью сформирован, а количество нейронов каждого слоя для соответствующей группы будет необходимым и достаточным для решения поставленной задачи. Механизм проектирования простого кластера, состоящего из двух слоёв с программной ёмкостью $W = 3$, представлен на рисунке 3.11.

В данном примере предполагается, что первая группа содержит два выходных нейрона, вторая – один и третья – два. На первом шаге производится подбор необходимого и достаточного количества нейронов на каждом из слоёв для решения соответствующей группе задачи. Предположим, что после экспериментов, первой группе необходимо три нейрона первого скрытого слоя и два второго, начиная от входа.

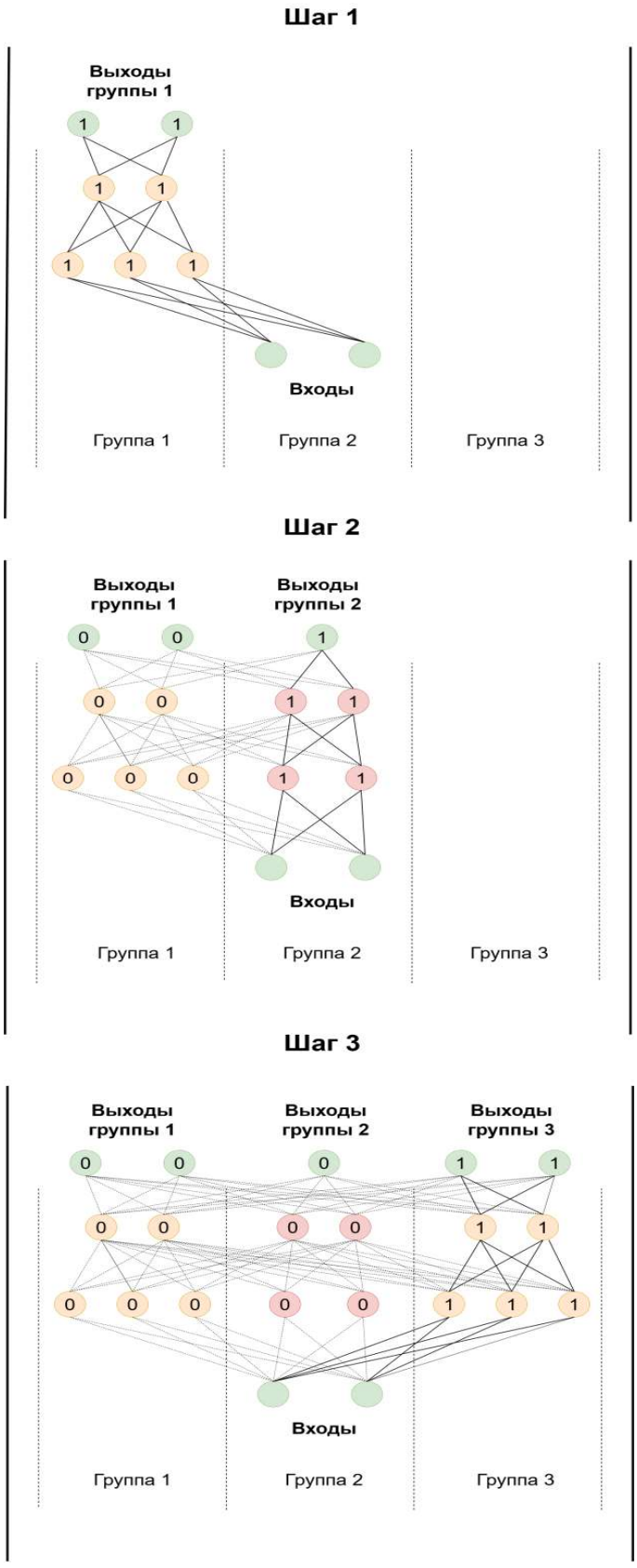


Рисунок 3.11 – Механизм проектирования простого кластера, состоящего двух слоёв с программной ёмкостью $W = 3$

На втором шаге выполним ту же процедуру, но с деактивацией нейронов первой группы. Пусть для неё необходимым и достаточным количеством нейронов является по два на каждом из глубоких слоёв. Деактивировав нейроны первой и второй групп, повторим операцию для третьей группы. Предположим, что, как и первой группе, третьей необходимо и достаточно трёх нейронов первого скрытого слоя и двух второго. Произведём суммирование полученных значений для каждой из групп по каждому слою.

В результате получим, что размерность вектора активационной маски первого скрытого слоя составляет восемь нейронов, второго – шесть, а для слоя выходов – пять. Учитывая последовательность групп и соответствующие им значения выполним формирование активационных масок, представленных в таблице 3.1.

Таблица 3.1 – Сгенерированные активационные маски для примера, представленного на рисунке 3.11.

Группа	**Слой 1**	**Слой 2**	**Слой выхода**
Группа 1	[1, 1, 1, 0, 0, 0, 0, 0,]	[1, 1, 0, 0, 0, 0,]	[1, 1, 0, 0, 0]
Группа 2	[0, 0, 0, 1, 1, 0, 0, 0]	[0, 0, 1, 1, 0, 0]	[0, 0, 1, 0, 0]
Группа 3	[0, 0, 0, 0, 0, 1, 1, 1]	[0, 0, 0, 0, 1, 1]	[0, 0, 0, 1, 1]

Следующим логичным вопросом является определение способа сопоставления активационной маски управляющему коду селектора групп. Ранее была введена некоторая функция $\Psi(p)$, ставящая в соответствие коду стратегии p некоторую битовую маску m, и представляющая математическую модель переключающего нейрона. Пусть множество P представляет собой совокупность всех команд. Его мощность равна W. Слой i управляется переключающим нейроном искусственного базального ганглия с индексом i. Множество M_i содержит всю совокупность возможных масок для конкретного слоя i. Исходя из необходимости в обеспечении взаимно

однозначного соответствия, его мощность согласована с мощностью множества всех возможных команд и равна программной ёмкости кластера W. Обозначим через $\Psi_i(p)$ функцию для переключающего нейрона, с индексом i. Очевидным способом сопоставления является аппроксимация функции $\Psi_i(p)$, соответствующей i-му искусственному нейрону базального ганглия, на основании заранее известных значений из множеств P и M_i, такая, что: $\Psi_i(p): P \to M_i$. Её применение накладывает дополнительные временные расходы на разработку кластера, связанные с обучением и тестированием. Наличие способа, обеспечивающего непрерывное обучение стратегий, без дополнительных сложных шагов, представляет наибольший практический интерес. Поскольку табличное представление функции является частным случаем аппроксимации, то исходя из малого объёма сопоставляемых данных, предлагается использовать табличную форму соответствия. В результате $\Psi_i(p)$ будет представлять собой функцию, содержащую словарь в формате ключ-значение, где в качестве ключа выступает управляющий код, а значения – соответствующая ему маска. Данный подход позволит постулировать маски внутри управляющих нейронов, тем самым ликвидируя потребность в дополнительных затратах на их обучение.

3.4. Сопряжение групп внутри кластера с разным количеством глубоких слоёв

В процессе проектирования нейронного кластера может возникнуть ситуация, при которой для решения задач требуется разное количество глубоких слоёв. В таком случае актуальным становится вопрос сопряжения групп внутри полносвязной нейронной сети. Подход с разбиением на группы, представляющий условную модульность, способен решить соответствующую задачу путём введения нейронов «повторителей». Обозначим требуемое

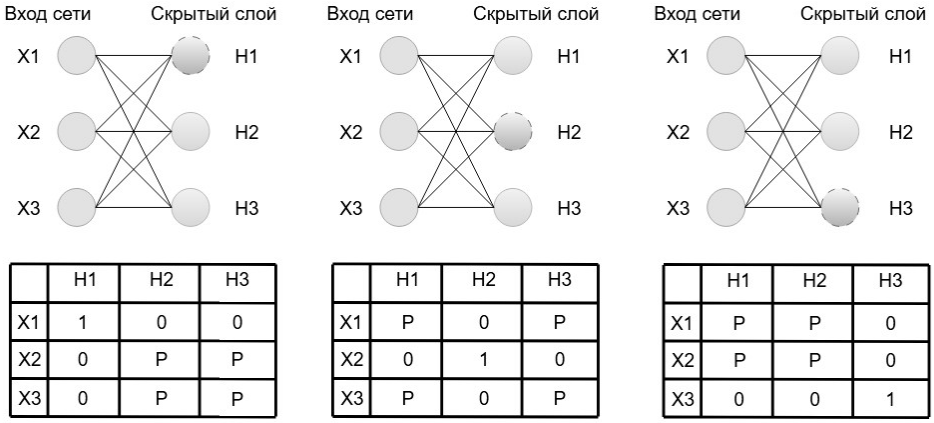
количество скрытых слоёв для некоторой группы w, как L_w, а размерность скрытого слоя l группы w, как $N_{w,l}$. Тогда нейронный кластер, составленный из W групп, будет иметь следующие структурные характеристики:

$$\begin{aligned} L &= \max(L_w) \\ N_l &= \sum_{l=1,w=1}^{L,W} N_{w,l} \end{aligned} \tag{3.1}$$

Здесь $L-$ количество скрытых слоёв обобщающей сети, N_l- количество нейронов скрытого слоя l, $l \in L$.

Исходя из того факта, что максимальное количество глубоких слоёв кластера определяется согласно критерию максимума для тех групп, у которых недостаточно слоёв, предлагается дополнить слоями-повторителями, обеспечивающими передачу значений фактического выхода группы на недостающие слои [35]. Основная задача нейронов-повторителей заключается в транспортировке входа на выход без искажения. Это необходимо для упрощения матричных операций в контексте полносвязной нейронной сети-кластера. Данная процедура предусматривает рассмотрение исключительно «прямых» связей нейронов, при которых каждый нейрон предыдущего слоя напрямую связан с соответствующим ему нейроном входа следующего слоя, без учёта сторонних связей. В результате возникает вопрос, связанный со способом «заморозки» лишних соединений за исключением прямых. Для обеспечения процедуры ликвидации сторонних связей предлагается корректировать матрицы весовых коэффициентов слоёв-повторителей. Рассмотрим рисунок 3.12, на котором представлены конфигурации участка сети с тремя входами и тремя нейронами-повторителями глубокого слоя, обеспечивающими трансляцию входа на выход для каждого из нейронов-повторителей (выделены штриховой линией). В приведённых на рисунке таблицах весов, символом P обозначен

вес некоторой связи, «1» – весовой коэффициент, реализующий передачу значения входа на выход, а «0» – «замороженные» связи. Единичный вес всегда устанавливается на пересечении индексов нейрона-повторителя и нейрона-источника. Коэффициенты «заморозки» ликвидируют сторонние связи с нейронами-соседями, обеспечивая передачу сигнала на выход только со стороны требуемого входа.



	H1	H2	H3
X1	1	0	0
X2	0	P	P
X3	0	P	P

	H1	H2	H3
X1	P	0	P
X2	0	1	0
X3	P	0	P

	H1	H2	H3
X1	P	P	0
X2	P	P	0
X3	0	0	1

Рисунок 3.12 – Конфигурации участка сети с тремя входами и тремя нейронами-повторителями глубокого слоя, обеспечивающими трансляцию входа на выход для каждого из нейронов-повторителей

Рассматривая таблицы весовых коэффициентов рисунка 3.12, становится очевидным, что для обеспечения передачи «прямого сигнала» необходимо обнулить все веса соединений, расположенных на пересечении индексов нейрона-источника и нейрона-повторителя. Определим такую операцию, как «вычёркивание». Стоит отметить, что её выполнение необходимо исключительно в процессе обучения сети, на этапах перед выполнением предсказания (после процесса стартовой инициализации), так и после проведения обновления весов. В результате, обучение нейронной сети,

с использованием нейронов-повторителей, производится по следующему алгоритму:

Шаг 1: Начальная инициализация весовых коэффициентов слоёв;

Шаг 2: Вычёркивание весов, соответствующих нейронам-повторителям;

Пока не достигнута допустимая погрешность ошибки предсказания, выполнить:

Начало цикла:

Шаг 3: Предсказание выходного вектора сети;

Шаг 4: Обновление весовых коэффициентов посредствам обратного распространения;

Шаг 5: Вычёркивание весов, соответствующих нейронам-повторителям;

Конец цикла.

Стоит отметить, что процедура вычёркивания требуется исключительно в случае совмещения групп разной длины. Для построения сети-кластера, с точки зрения минимизации вычислительных затрат, рекомендуется использовать группы одинаковой длины.

3.5. Преимущества и недостатки использования предлагаемой архитектуры глубокой нейронной сети

Главной особенностью предлагаемой архитектуры глубокой нейронной сети-кластера является то, что в отличие от остальных подходов, игнорирующих проблематику расширения репрезентативной способности глубокой сети и занимающихся решением проблемы обеспечения агента многофункциональностью за счёт добавления отдельных сетей-модулей, она

решает проблематику повышения репрезентативности одной сети и, как следствие, обеспечивает многофункциональность построенного на её основе агента. Ниже представлен перечень преимуществ использования архитектуры, основанной на нейронном кластере и переключающих нейронах:

- предоставляет простой и понятный механизм селекции стратегий;

- селекция, за счёт использования табличного хранения масок, не нуждается в дополнительных шагах обучения;

- архитектура решает проблему селекции стратегий, свойственную прогрессивным нейронным сетям;

- решает проблему «забывания» за счёт увеличения репрезентативной способности полносвязной сети-кластера;

- увеличивает репрезентативную способность одной полносвязной сети;

- в отличие от дистилляции, данная архитектура не теряет деталей и тонкостей решаемой задачи, поскольку для каждой задачи выделена собственная группа нейронов внутри кластера;

- управление селекцией групп осуществляется «сбоку», что упрощает настройку групп;

- функционирование каждой группы осуществляется в унифицированной манере, с использованием одной и той же матрицы весов сети кластера.

Предложенная архитектура обладает основным недостатком, связанным с использованием вычислительных ресурсов. Поскольку группа представляет собой часть основной сети, а обучение и предсказание осуществляются по всей сети в целом, то неизбежны дополнительные накладные расходы на «лишние» расчёты для неиспользуемых связей «замороженных» групп. Если кластер использует прогрессивные нейронные сети или же группы имеют разные длины, то это также сопряжено с

дополнительными расчётами, связанными с обратным распространением и реализацией вычёркивания соответственно. Несмотря на данный недостаток, количество преимуществ значительно перевешивает в сторону актуальности применения данной архитектуры для организации агента, основанного на глубоком обучении с подкреплением. В особенности для решения задачи управления локомоцией шагающего робота.

3.6. Выводы по разделу

В рамках данного раздела было рассмотрено алгоритмическое обеспечение системы управления интеллектуальным агентом, реализующим локомоцию киберфизической системы мобильного шагающего робота, с точки зрения обучения с подкреплением, как в общем виде, так и с учётом концепции критических состояний. С точки зрения парадигмы объектно-ориентированного проектирования произведен структурный анализ системы управления с разбиением её на основные функциональные блоки. Выполнено описание особенностей функционирования каждого из них. Реализовано определение связей между блоками, включая их тип. Представлена инновационная архитектура нейронной сети агента, позволяющая обеспечить его возможностью обучаться и выполнять несколько локомоторных паттернов. Произведено детальное описание архитектуры глубокой сети и механизмов её функционирования. Введены понятия нейронный кластер, нейронная группа и переключающий нейрон. Введено понятие программной ёмкости, являющейся количеством обрабатываемых кластером программ (стратегий) и механизм их селекции, основанный на уникальных бинарных кодах. Определены принципы функционирования переключающего нейрона, генерирующего активационные маски в соответствии с кодом селектора программ. Рассмотрено применение предлагаемой архитектуры к прогрессивным нейронным сетям. Продемонстрирована структура агента,

построенного на прогрессивных нейронных сетях, с учётом предлагаемой архитектуры, и модифицированный алгоритм реализации предсказания выхода. Даны рекомендации по способу генерации активационных масок и сопряжению групп разной длины. Произведено обоснование преимущества предлагаемой архитектуры, по сравнению с существующими вариантами.

ГЛАВА 4. СТРУКТУРА СИСТЕМЫ УПРАВЛЕНИЯ ИНТЕЛЛЕКТУАЛЬНЫМ АГЕНТОМ КИБЕРФИЗИЧЕСКОЙ СИСТЕМЫ МОБИЛЬНОГО ШАГАЮЩЕГО РОБОТА

В контексте настоящей главы производится формирование функции вознаграждения, обеспечивающей реализацию локомоторного паттерна «движение вперёд». Производится обоснование выбора используемого системой управления алгоритма глубокого обучения с подкреплением. Реализуется архитектура агента для формирования двух стратегий: движение прямо по ровной поверхности и движение прямо по песчаной поверхности. Формируется структура системы управления интеллектуальным агентом, реализующим локомоцию мобильного шагающего робота для конкретного тестового стенда. Производится постановка экспериментов, на основании которых осуществляется анализ функционирования системы, с точки зрения, предложенной в рамках исследования концепции критических состояний и многофункциональности, обеспечиваемой введенной в рамках работы новой архитектурой на базе нейронного кластера и переключающих нейронов. В конце приводятся выводы к соответствующей главе.

4.1. Определение функции вознаграждения

В контексте исследования рассматривается устойчивое прямолинейное движение вперёд. Система управления строится для агента, использующего глубокое обучение с подкреплением. Следовательно, основная цель обеспечивается путём задания функции вознаграждения. Исходя из контекста решаемой задачи, требуется выполнить движение вперёд с минимальным отклонением от прямой, с учётом устойчивого равновесия в процессе локомоции и реализацией физически адекватной походки. На основе соответствующих данных, выполним формирование функции

вознаграждения. В первую очередь требуется определить необходимый сенсорный аппарат робота. Для ориентации в пространстве используется датчик гироскоп-акселерометр. Фиксации факта соприкосновения стоп с поверхностью земли осуществляется при помощи датчиков касания. Соударение конечностей или столкновение с препятствием фиксируется посредством датчика удара, состоящего из матриц микропереключателей, имитирующих тактильные ощущения. Согласно проведённому исследованию, посвящённому формированию функции вознаграждения, она представима в виде параметрической функции следующего вида [24]:

$$R = \sum_{i=1}^{N} \alpha_i R_i \,, \tag{4.1}$$

где R – общее вознаграждение, R_i – i-ая функция вознаграждения из N имеющихся, α_i – настраиваемые коэффициент значимости.

В результате получим следующую общую структуру функции вознаграждения, согласно используемому сенсорному аппарату:

$$R = \alpha_{forward} R_{forward} + \alpha_{direction} R_{direction} + \alpha_{stability} R_{stability} + \alpha_{contact} R_{contact} + \alpha_{smooth} R_{smooth} + \alpha_{penalty} R_{penalty} \tag{4.2}$$

где $R_{forward}$ – вознаграждение за движение вперёд, $R_{direction}$ – вознаграждение за движение без отклонения от траектории, $R_{stability}$ – вознаграждение за удержание баланса, $R_{contact}$ – вознаграждение за контакт с землёй, R_{smooth} – вознаграждение за плавность движений, $R_{penalty}$ – штраф, α – соответствующий коэффициент значимости.

Основным компонентом функции вознаграждения, в данном случае, является награда за движение вперёд $R_{forward} = \frac{\Delta x}{\Delta t}$, где Δx – изменение

координаты центра масс вдоль требуемого направления, Δt – интервал времени. В данном случае не следует вознаграждать агента за позицию, поскольку стратегия может быть выбрана таким образом, что агент будет двигаться вперёд прыжками. Для реализации прямолинейного движения требуется генерировать вознаграждение за скорость. Немаловажным является ликвидация отклонений от траектории. Для этой цели предусмотрено соответствующее вознаграждение $R_{direction}$, которое можно сформировать как на основе координат центра масс $R_{direction} = e^{-k_y|y|}$ (или более мягкий вариант $R_{direction} = e^{-k_y y^2}$), так и на базе показаний гироскопа-акселерометра $R_{direction} = e^{-k_\theta|\theta|}$, где y – отклонение центра масс по оси Y, θ – угол отклонения от направления вперёд, k_y, k_θ – некоторый настраиваемый коэффициент. Данная награда необходима для осуществления прямолинейного движения, поскольку в противном случае робот будет двигаться «боком». Для обеспечения корректного чередования конечностей отвечает вознаграждение контакта с землёй $R_{contact}$. Оно позволяет не допустить прыжки в процессе перемещения. Самым простым вариантом её формирования является назначение штрафа за слишком долгий контакт конечностей или за отсутствие контакта, что характерно для прыжка.

Одной из важных функций вознаграждения является функция R_{smooth}, обеспечивающая плавность движения и ликвидирующая резкие переходы между состояниями. Это позволяет сделать движения «естественными». Данная функция имеет вид $R_{smooth} = e^{-k_a\|a_t - a_{t-1}\|}$, где a_t – текущее действие, k_a – настраиваемый коэффициент. В качестве дополнительного контроля потребления энергии при движении можно использовать функцию вознаграждения R_{energy}. Поскольку в контексте настоящего исследования не ставится задача оптимизации энергопотребления, то данная функция вознаграждения не используется.

Для ликвидации возможности травматизации механизма вводится вознаграждение (штраф) $R_{penalty}$. Оно должно быть задано достаточно большим числом, поскольку в противном случае агент будет генерировать действия, приводящие к соударениям узлов и прочим травмирующим состояниям. Функция вознаграждения $R_{stability}$ предназначена для поощрения удержания баланса во время движения.

При выполнении локомоторной задачи, важным критерием является сбалансированность движения. Баланс представляет собой не бинарное состояние, поскольку необходимо:

- оставаться в устойчивой области опорной плоскости;
- стараться не приближаться к критической области;
- обладать запасом устойчивости;
- иметь возможность восстановления равновесия.

Исходя из этого, данная функция вознаграждения (так же, как и основная) должна быть потенциальной, а не в форме простого штрафа. То есть величина вознаграждения зависит не от событий, а от качества текущего состояния.

Функция вознаграждения, отвечающая за баланс, должна давать плавный градиент, генерировать большой штраф за выход за границы допустимой области, поощрять запас устойчивости и учитывать будущее состояние (для случая использования capture point). Для определения сбалансированности используются: точка центра масс, точка нулевого момента ZMP и точка захвата движения Capture Point. Поскольку ранее упоминалось, что в контексте исследования не рассматриваются высокоскоростные паттерны, то точка CP не будет учтена при формировании вознаграждения. Пусть S – область опорного полигона, Δ_{cr} – ширина критической области, $S_{\Delta_{cr}}$ – безопасная область полигона, причём $S_{\Delta_{cr}} \subset S$,

d_{CoM}, d_{ZMP} – расстояния соответствующих точек до центра опорного полигона. Тогда

$$R_{ZMP} = \begin{cases} e^{-k_{ZMP} d_{ZMP}^2}, & d_{ZMP} \in S_{\Delta_{cr}} \\ -K_1 & d_{ZMP} \in \Delta_{cr} \\ -K_2 & d_{ZMP} \notin S \end{cases}$$

и

$$R_{CoM} = \begin{cases} e^{-k_{CoM} d_{CoM}^2}, & d_{CoM} \in S_{\Delta_{cr}} \\ -K_1 & d_{CoM} \in \Delta_{cr} \\ -K_2 & d_{CoM} \notin S \end{cases},$$

где K_1, K_2 – подбираемые значения штрафов, k_{ZMP}, k_{CoM} – настраиваемые коэффициенты.

Важно отметить, что для формирования вознаграждений все расстояния и скорости должны быть нормализованными. В ранее описанных функциях можно заметить использование экспоненты. Её применение обусловлено наличием гладкого градиента, отсутствием резких скачков и постоянным наличием сигнала.

4.2. Выбор алгоритма глубокого обучения с подкреплением и влияние архитектуры глубокой сети на функционирование агента

Основным ядром разрабатываемой системы управления является метод глубокого обучения с подкреплением, от выбора которого напрямую зависит качество генерации управляющих сигналов. Ранее было отмечено, что градиентные алгоритмы позволяют обеспечить аппроксимацию стратегии с возможностью функционирования при наличии большого или непрерывного множества действий. В работе [22] было произведено исследование зависимости метода REINFORCE от параметров среды, в результате которого данный алгоритм продемонстрировал негативную корреляцию с качеством выполняемой задачи. При дальнейшем поиске подходящего

алгоритма, для построения системы управления локомоцией шагающего робота, был рассмотрен метод исполнитель-критик. В работах [26, 28, 29] были определены факторы, влияющие на сходимость и устойчивость алгоритма, включая воздействие архитектуры используемой глубокой нейронной сети на качество функционирования агента. Согласно проведённым исследованиям с ростом количества скрытых слоёв падает производительность алгоритма. Оптимальным числом, для решения большинства задач, как показала совокупность произведённых экспериментов, являются два-три скрытых слоя. Было рассмотрено сравнение производительности и качества алгоритма при использовании разделённой сети исполнитель-критик, когда сеть исполнителя и критика представляют собой разные сети без общих скрытых слоёв, и объединённой конфигурации, как представлено на рисунке 4.1.

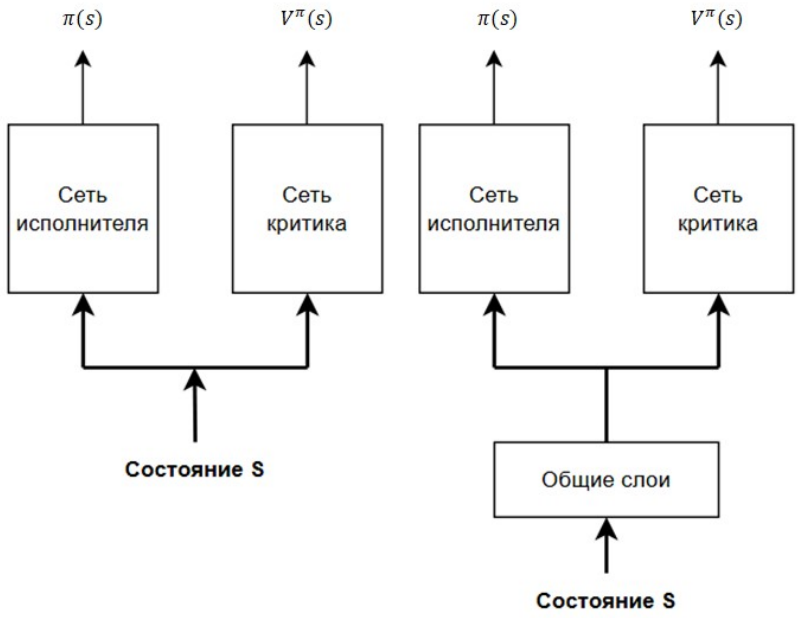


Рисунок 4.1. Варианты архитектуры сети исполнитель-критик

Согласно полученным экспериментальным данным, использование объединённой архитектуры обеспечивает лучшую сходимость алгоритма, но

делает агента нестабильным по причине распространения суммарной ошибки исполнителя и критика. Поскольку стабильность обеспечивает качество выполнения задачи агентом, для построения системы управления локомоцией шагающего робота целесообразно использовать разделённую архитектуру исполнителя и критика без общих скрытых слоёв.

Эксперименты показали, что алгоритм исполнитель-критик обладает множеством проблем, таких как высокая дисперсия, неполнота опыта и отсутствие стабильности при решении непрерывных задач, к которым относится подавляющее большинство проблематик реального мира, включая управление локомоцией. Дальнейшие поиски привели к алгоритму проксимальной оптимизации стратегии (Proximal Policy Optimization, PPO), основанному на методе исполнитель-критик и ликвидирующему вышеперечисленные недостатки. В [36] был проведён сравнительный анализ алгоритма PPO и исполнитель-критик при разных архитектурах глубоких нейронных сетей исполнителя и критика, который продемонстрировал преимущество алгоритма проксимальной стратегии над алгоритмом исполнитель-критик с точки зрения стабильности и скорости сходимости и подтвердил необходимость использования разделённой архитектуры глубокой сети. В результате в контексте настоящей работы, в качестве основного алгоритма глубокого обучения с подкреплением рассматривается алгоритм проксимальной оптимизации стратегии с разделённой архитектурой глубокой сети исполнитель-критик.

4.3. Эксперименты по построению многофункционального агента на основе нейронного кластера с переключающими нейронами с использованием библиотеки Open AI Gym

Перед проектированием многофункционального агента разрабатываемой системы управления требуется произвести апробацию

предлагаемой архитектуры в контролируемых лабораторных условиях имитационной среды [33]. Выполним построение тестового многофункционального агента, основанного на ранее предложенной архитектуре нейронной сети-кластера с переключающими нейронами, и произведём анализ его функционирования на основе имитационных сред CartPole и CliffWalking библиотеки OpenAi Gym [94]. Основная идея тестового стенда заключается в попытке запоминания одним агентом двух стратегий решения соответствующих задач, где CartPole – балансировка инвертированного маятника и CliffWalking – «блуждание возле обрыва». В результате, программная ёмкость сети кластера равна 2.

CartPole-v0 – это классическая задача управления удержанием баланса инвертированного маятника, основной целью, которой является обеспечение максимального времени удержания стержня в состоянии равновесия, путём перемещения подвижной тележки, к которой он закреплён посредством шарнира [94]. Каждый шаг, при котором обеспечивается баланс, вознаграждается сигналом, равным 1. Эпизод взаимодействия со средой считается завершённым при достижении горизонта, равного 200 или при потере баланса маятником. Множество возможных действий состоит из двух: 0 – отклонение тележки влево, а 1 – отклонение тележки вправо. Наблюдение состоит из четырёх элементов: положения тележки, её скорости, угла наклона опоры и её угловой скорости.

Сеточный мир среды CliffWalking-v0 состоит из таблицы 4×12 клеток [94]. Стартовым состоянием является состояние 36, или ячейка [3, 0], а целевым – состояние 48 или ячейка [4, 11]. Между ними расположена запрещённая область, имитирующая обрыв. Основной целью агента в данной среде является формирование стратегии достижения целевого состояний из стартового, обеспечивающего безопасное перемещение вдоль обрыва. Множество действий состоит из четырёх элементов: 0 – движение вверх, 1 – движение вправо, 2– движение вниз и 3 – движение влево. Наблюдение

представляется в виде числа, составленного из координат соответствующей ячейки по формуле: $obs = iN + j$, где i – индекс строки ячейки, начинающийся с 0, j – индекс столбца ячейки, начинающийся с 0 и N – количество столбцов. Каждый «безопасный» шаг вознаграждается в размере -1. Попадание агента в зону обрыва прерывает взаимодействие и генерирует награду в размере -100. Максимально возможное вознаграждение за эпизод для данной имитационной среды равно -13.

Данные имитационные среды обладают разными характеристиками входных (состояний) и выходных (действий) параметров. Объединение двух сред в контексте одного агента подразумевает приведение их входов и выходов к общему виду. В результате, наблюдение было представлено в виде вектора из шести значений, а вектор действий в виде вектора, размерностью 4. Исходя из того факта, что CliffWalking-v0 ставит в соответствие наблюдению число, максимальное значение которого равно 48, что в двоичной форме эквивалентно числу, состоящему из шести разрядов 110000, а среда CartPole-v0 вектор, размерностью 4, то добавив два разряда с нулевыми значениями к наблюдению среды CartPole становится возможным унификация входных данных. Среда CliffWalking обладает в два раза большим числом возможных действий по сравнению со средой маятника. Поэтому вектор действий имел длину в четыре единицы, первые два разряда которого, использовались для среды CartPole. Алгоритмом глубокого обучения с подкреплением, применяемым для тестового стенда, являлся исполнитель-критик с преимуществом (A2C). Нейронная сеть многофункционального агента базировалась на двух полносвязных сетях исполнителей и критиков. Каждая группа нейронов представляла собой двухслойную сеть, состоящую из 128 нейронов. Для реализации селекции групп были добавлены по два переключающих нейрона, как для сетей исполнителей, так и для критиков. Группе, формирующей стратегию для решения задачи среды CliffWalking-v0, был присвоен код 00, а для CartPole-

v0 – 01. Результирующая архитектура тестовых агентов представлена на рисунке 4.2.

Настройка каждого тестового агента производилась при одинаковых условиях и параметрах алгоритма глубокого обучения с подкреплением. Количество эпизодов соответствовало 2000. Скорость обучения сетей исполнителя и критика соответствовала 0.0005 . Коэффициент обесценивания был равен 0.95.

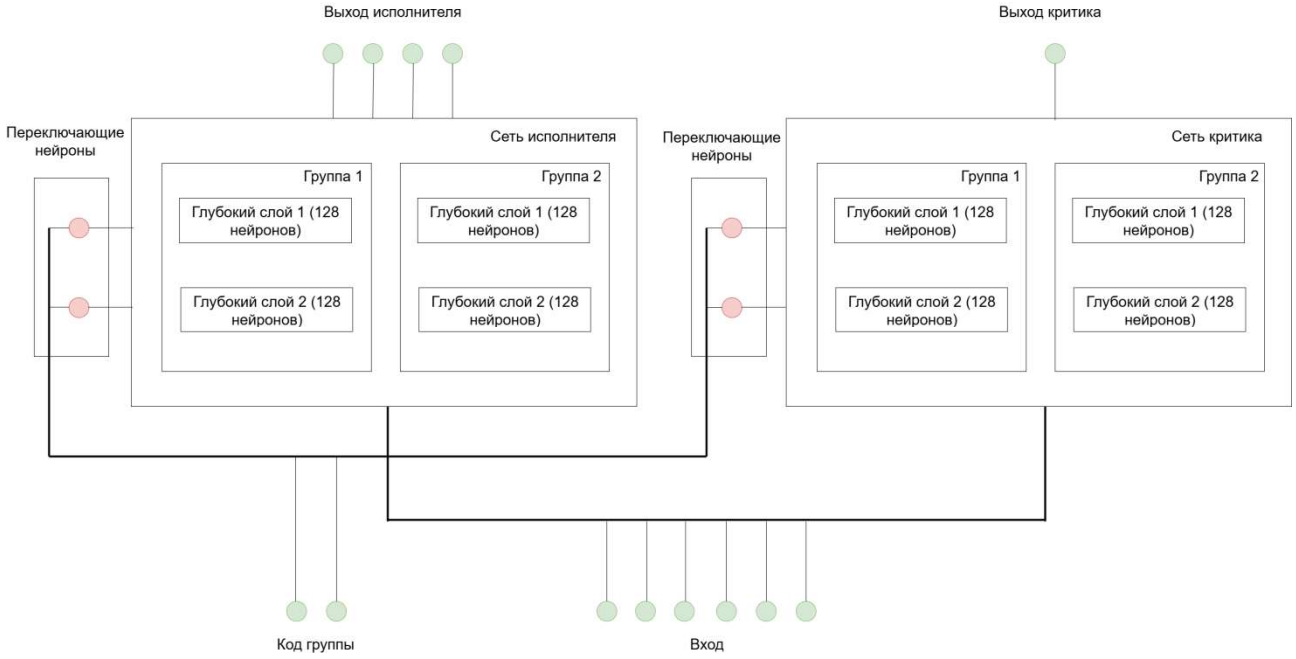


Рисунок 4.2. Архитектура тестового агента

В ходе экспериментов, на начальном этапе производилось обучение стратегий, соответствующих имитационным средам. Измерялся процент эпизодов без возникновения коллизий для каждого из случаев. Далее, стратегии менялись местами. То есть обученная стратегия CartPole помещалась в среду CliffWalking и стратегия CliffWalking в CartPole и измерялся показатель метрики. Результаты экспериментов с указанием процента эпизодов без возникновения коллизий представлены в таблице 4.1. В процессе экспериментирования было получено, что каждая нейронная

группа, в контексте одного агента, произвела полноценную аппроксимацию каждой из стратегий. Это подтверждает процент эпизодов без возникновения коллизий. Низкий процент соответствующей метрики при чередовании сред и стратегий показывает корректность механизма переключения нейронных групп, поскольку группа, обученная выполнению одной стратегии, при переключении на решение иной задачи демонстрирует резкое снижение производительности и качества выполнения.

Таблица 4.1 – Метрика процента эпизодов без возникновения коллизий

Стратегия обучения	Переход	Процент эпизодов без коллизий
CartPole	CartPole - CartPole	95
CliffWalking	CliffWalking - CliffWalking	93
CartPole	CartPole - CliffWalking	5
CliffWalking	CliffWalking - CartPole	2

В результате предлагаемая архитектура, основанная на полносвязной сети-кластере с применением переключающих нейронов, обеспечивает независимую настройку каждой из групп, ликвидирует эффект «забывания» и решает проблему ограниченной репрезентативной способности нейронной сети. Использование переключающих нейронов позволяет производить селекцию групп внутри сети-кластера в соответствии с кодом группы.

4.4. Аппроксимация функции близости к неустойчивости $C(s)$

Ранее, в главе 2 при рассмотрении предлагаемой в контексте работы концепции критических состояний подчёркивалось, что функция близости к неустойчивости может быть задана как эвристически (на основе показаний сенсоров), так и быть аппроксимирована посредствам нейронной сети.

Подробно рассмотрим аппроксимацию функции близости к неустойчивости. Ранее обсуждалось, что функция $C(s)$ показывает, насколько близко конкретное состояние к падению. Пусть задана глубокая нейронная сеть $C_\phi(s)$, которая на основании текущего состояния предсказывает риск падения в ближайшие k шагов (вероятность падения). На её выходе генерируется некоторое число из интервала $[0,1]$. Чем меньше предсказанное значение, тем безопаснее состояние.

Применение аппроксимации функции $C(s)$ расширяет смысловую нагрузку параметра k. Во-первых, обучается сеть определения риска $C_\phi(s)$, которая определяет вероятность возникновения отказа в ближайшие k шагов из текущего состояния. Во-вторых, параметр k участвует в реализации «отката», смысл которого заключается в возврате в состояние, которое возможно стало первопричиной возникновения отказа. Обычный алгоритм обучения с подкреплением учится на последствиях, в то время как предложенный в главе 2 алгоритм ликвидации критических состояний пытается определить область зарождения ошибки, а не только её финальное проявление. В результате, параметр k задаёт глубину анализа причин возникновения критического состояния. Он определяет горизонт прогнозирования функции риска и одновременно глубину отката траектории при восстановлении после критического состояния. Благодаря этому агент обучается не только избегать падений, но и распознать состояния, из которых падение становится вероятным в ближайшем будущем.

В начале обучения, $C_\phi(s)$ генерирует случайные значения. В результате возникают хаотичные падения, переходы в опасные состояния и отсутствует полноценный штраф за риск, что приводит к нестабильности обучения и опасно для реального робота. По этой причине предлагается объединить эвристическое представление с аппроксимацией следующим образом:

$$C(s) = (1 - \alpha(t)) \cdot C_{heur}(s) + \alpha(t) \cdot C_\phi(s), \tag{4.3}$$

где α – коэффициент доверия сети, t – время, $C_{heur}(s)$ – риск, сгенерированный эвристической формулой, $C_\phi(s)$ – риск, предсказанный нейронной сетью.

Коэффициент доверия сети α в начале обучения близок к 0. В процессе постепенно возрастает до тех пор, пока не будет близок к 1. Зададим α в виде линейной функции следующего вида:

$$\alpha(t) = \min\left(1, \frac{t}{T}\right), \tag{4.4}$$

где T – момент полного перехода.

Если один эпизод состоит из N шагов, а переход с эвристики на прогноз сети планируется за K эпизодов, то $T = K \cdot N$.

Данные для обучения необходимо формировать на основе полученных в процессе взаимодействия траекторий. Для каждого состояния s_t создаётся метка $y = 1$, если спустя k шагов, после его наступления произошло падение, иначе $y = 0$. В качестве функции потерь используется бинарная перекрестная энтропия:

$$Loss = -y \log C_\phi(s) - (1 - y)\log(1 - C_\phi(s)) \tag{4.5}$$

Поскольку в процессе обучения, за один эпизод безопасных состояний значительно больше, чем приводящих к падению (модель обладает малым набором данных) в результате чего возникает дисбаланс классов, приводящий к тому, что значение $C_\phi(s)$ будет равно 0 для всех состояний. Чтобы решить данную проблему, следует модифицировать формулу ошибки, добавив весовые коэффициенты ω_0, ω_1:

$$Loss = -\omega_1 y \log C_\phi(s) - \omega_0(1-y)\log(1-C_\phi(s)) \tag{4.6}$$

Коэффициенты задаются следующим образом:

$$\begin{aligned} \omega_0 &= 1 \\ \omega_1 &= \frac{N_0}{N_1}, \end{aligned} \tag{4.7}$$

где N_0 – число безопасных состояний, N_1 – число опасных состояний.

4.5. Архитектура многофункционального агента проектируемой системы управления

Выполним формирование агента разрабатываемой системы управления на примере шагающего робота с двумя педипуляторами. Каждая конечность механизма обладает пятью подвижными суставами. В общем случае количество сочленений N равно 10. Исходя из этого, получим вектор состояния робота в момент времени t:

$$(\theta_1 - \theta_5, \theta_6 - \theta_{10}), \tag{4.8}$$

где $\theta_1 - \theta_5$ соответствуют правому педипулятору, $\theta_6 - \theta_{10}$ – левому.

Важно отметить, что все углы представляются строго в радианах, а не в фактических значениях углов. Каждой конечности соответствует датчик касания с поверхностью земли. В результате, получим вектор касания, состоящий из фиксации факта соприкосновения с поверхностью земли, где T_1 соответствует правому педипулятору, а T_2 – левому:

$$(T_1, T_2), \tag{4.9}$$

Робот оснащён датчиками соударения P и удержания баланса B, каждый из которых генерирует соответствующий логический сигнал. Среда также формирует сигнал E завершения, учитывая достижение горизонта эпизода. Для фиксации завершения эпизода объединим данные сигналы операцией дизъюнкции: $End = P \vee B \vee E$. За каждый переход в новое состояние, генерируется вознаграждение R по ранее описанному принципу, но в вектор наблюдения оно не входит.

В результате, согласно общему виду интерфейса, рассмотренному в главе 2, и характеристикам выбранного тестового стенда, вектор наблюдения примет следующий вид:

$$o_t = \left(\theta^t_{1-10}, \theta^{t-1}_{1-10}, \upsilon^t_{1-10}, \upsilon^t_{CoM_x}, \upsilon^t_{CoM_y}, a_x, a_y, a_z, g_x, g_y, g_z, T^t_{1-2}, End\right) \tag{4.10}$$

Выход агента представляет собой вектор, содержащий небольшое значение коррекции величины угла в радианах для каждого сочленения:

$$(c_1 - c_5, c_6 - c_{10}), \tag{4.11}$$

Такой подход позволяет не управлять углами напрямую, а изменять их путём генерации небольшой величины коррекции. Важным является введение вектора масштабирования ψ, который содержит масштабы изменений для каждого сочленения − максимально безопасное отклонение конкретного сустава от базовой позы (в радианах). Элемент вектора ψ − это не полный диапазон сустава, а значение равное 10-50% от безопасного диапазона, которое подбирается экспериментально или представляется в виде обучаемого параметра. Имея сгенерированные агентом значения вектора c и

вектор масштабирования ψ, получается фактическое значение коррекции текущего состояния:

$$\begin{aligned} \Delta &= c\psi \\ d_{t+1} &= d_t + \Delta \end{aligned}, \tag{4.12}$$

где Δ – вектор, содержащий величины корректировки углов сочленений, c – сгенерированные агентом значения коррекции, ψ – вектор масштабирования, d_t – состояние в момент времени t, d_{t+1} – новое состояние в момент времени $t+1$.

Зачастую алгоритм проксимальной оптимизации политики PPO, используемый в исследовании, возвращает действия не «напрямую», а посредствам аппроксимации параметров нормального распределения μ – среднего и отклонения σ, что является стандартной практикой при наличии очень большого или непрерывного множества действий [18, 41, 44, 49, 56]. Для формирования действия на основании сгенерированных сетью исполнителя параметров μ и σ строится нормальное распределение, из которого случайным образом выбирается некоторое значение. На основании полученного значения определяется вероятность действия, необходимая для функционирования алгоритма PPO и формируется вектор c, характеризующий отклонения, путём применения функции гиперболического тангенса. Это необходимо, поскольку нормальное распределение рассматривается в диапазоне $(-\infty,+\infty)$. Функция tanh обеспечивает преобразование в диапазон $[-1,1]$. В результате алгоритм получения вектора действий c имеет следующий вид [18, 41, 44, 49, 56]:

Шаг 1: Предсказание нейронной сетью исполнителя параметров μ и σ;

Шаг 2: Формирование нормального распределения на основе полученных параметров μ и σ;

Шаг 3: Получение случайным образом значения из сгенерированного распределения (сэмплирование);

Шаг 4: Получение вероятностей на основе полученного значения;

Шаг 5:Получение фактического вектора действия c, путём обработки полученного значения функцией tanh.

Такой подход к генерации действий, основанный на нормальном распределении, позволяет решить проблематику исследования-использования, свойственную алгоритмам обучения с подкреплением. В результате обучения, параметр σ значительно уменьшается, так что действие становится равным μ.

Поскольку основной акцент данного исследования сфокусирован на обеспечении увеличения репрезентативной способности глубокой нейронной сети агента и безопасном обучении, то для формирования архитектуры и последующего тестирования в реальных условиях воспользуемся двумя паттернами: движение прямо по ровной поверхности и движение прямо по песчаной (вязкой) поверхности. В результате программная ёмкость проектируемого нейронного кластера будет равна $W = 2$.

Определим архитектуру глубокой нейронной сети агента, с учётом обеспечения многофункциональности. Исходя из работ [26, 29] следует, что для реализации качественной аппроксимации стратегии глубокая сеть должна состоять максимум из трёх скрытых слоёв. Оптимальным числом является два скрытых слоя. Это связано с тем, что слишком глубокие сети приводят к нестабильному обучению. Поскольку основной проблемой глубоких сетей является затухание градиентов, то при наличии большой глубины сети сигнал вознаграждения будет зашумлённым. Велика вероятность переобучения.

В работе [26] было рассмотрено влияние функции активации, в результате которого сделаны выводы, что гиперболический тангенс Tanh имеет качественные преимущества перед функцией ReLU. Так же было рассмотрено влияние количества нейронов скрытого слоя на производительность алгоритма глубокого обучения с подкреплением, в результате которого было определено, что оптимальным количеством является 128 нейронов в слое. Исходя из этого, глубокая нейронная сеть-кластер агента разрабатываемой системы управления представляет собой многослойный перцептрон и характеризуется программной ёмкостью $W = 2$, двумя скрытыми слоями по 256 нейронов каждый с функцией активации tanh. То есть на каждую группу выделяются по 128 нейронов на каждом слое. Поскольку нейронный кластер состоит из двух групп по два слоя каждая, причём выход и вход которых идентичен, то для управления переключением между группами необходимо сформировать искусственный базальный ганглий, состоящий из двух переключающих нейронов и поставить в соответствие каждой группе бинарный код, где 0 – для первой группы, 1 – для второй. Используемый в исследовании алгоритм проксимальной оптимизации стратегии PPO, основан на методе исполнитель-критик. Исходя из работы [29] для устранения эффектов совместной ошибки, сети исполнителя и критика будут полностью разделены. Отсюда так же следует использование двух нейронных кластеров для сетей исполнителя и сетей критика. Разрабатываемый агент учитывает критические состояния. Для этого предусмотрен дополнительный нейронный кластер, аппроксимирующий функцию C_ϕ, аналогичный кластерам исполнителя и критика.

На вход каждой нейронной сети-кластера подаётся наблюдение и код соответствующей группы. На выходе сеть-кластер исполнителя агента генерирует параметры нормального распределения (сеть исполнителя), которые впоследствии преобразуются функцией гиперболического тангенса

в вектор действия, согласно ранее описанному алгоритму и значение функции ценности (сеть критика) для обеспечения корректировки выбора. Сеть-кластер критика продуцирует значение функции ценности. На выходе третьего кластера формируется предсказание C_ϕ.

В итоге, архитектура агента разрабатываемой системы управления для реализации двух стратегий, представлена на рисунке 4.3.

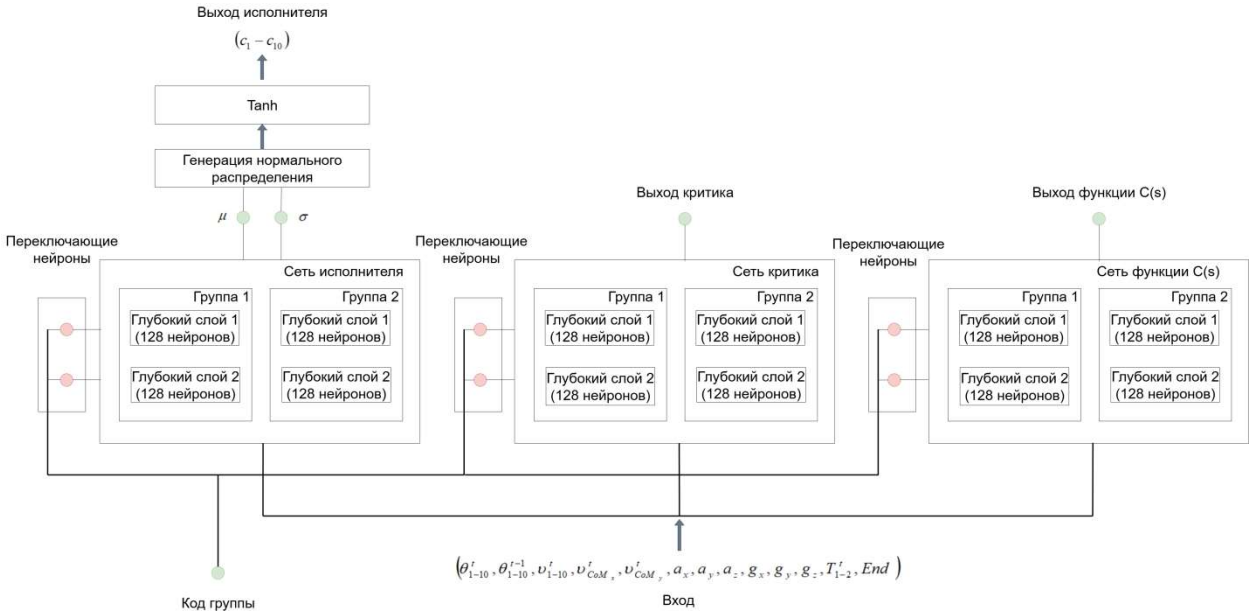


Рисунок 4.3 – Архитектура агента разрабатываемой системы управления

4.6. Структура системы управления

На основе полученных результатов экспериментов по обеспечению многофункциональностью на имитационных средах, архитектура системы управления агентом, отвечающим за формирование локомоции мобильного шагающего робота, для реализации прямолинейного движения представлена на рисунке 4.4.

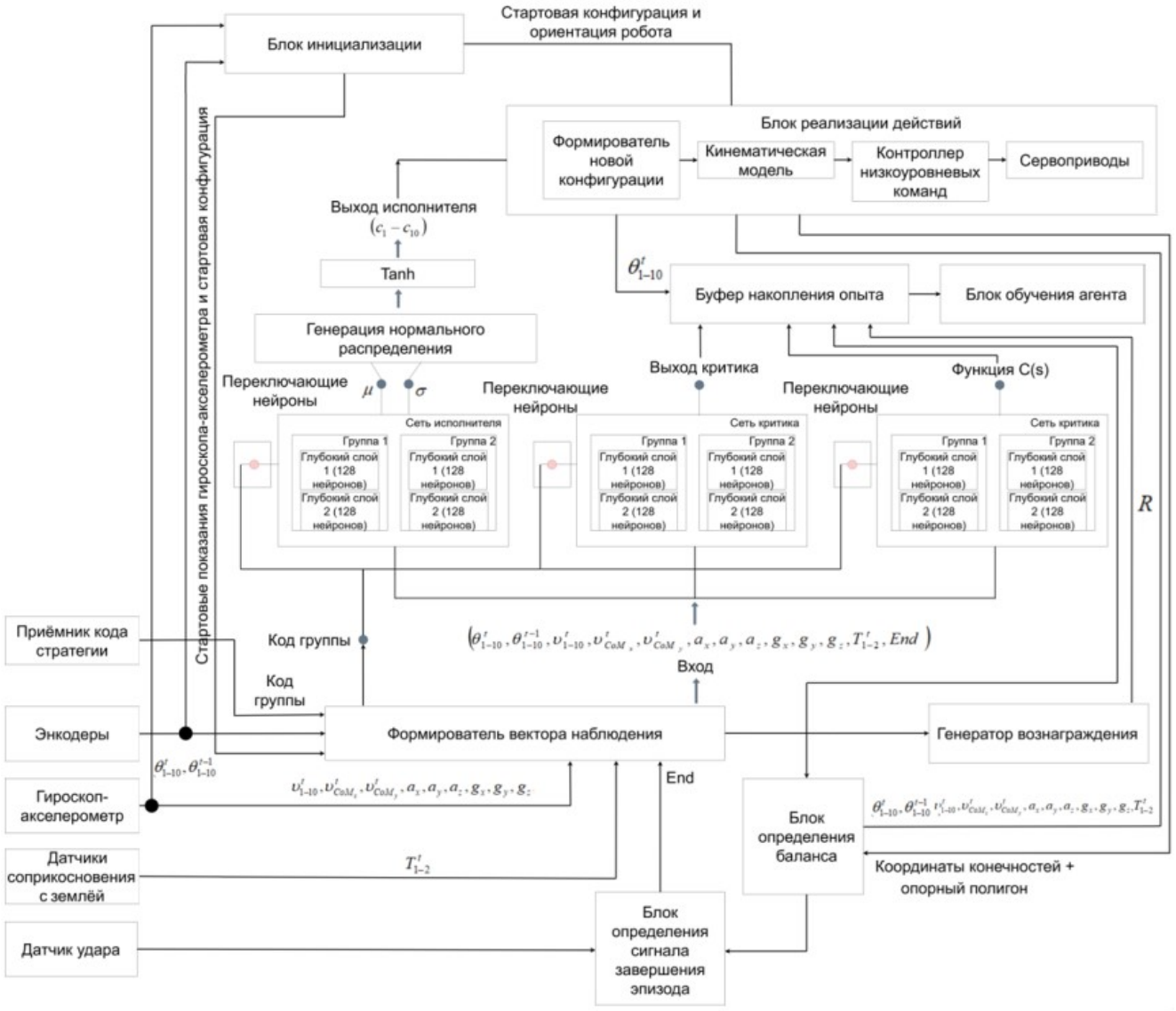


Рисунок 4.4 – Структура разработанной системы управления агентом, обеспечивающим управление локомоцией мобильного шагающего робота для двух стратегий

При включении питания робота, активируется блок инициализации. Производится стартовая инициализация механизма, в результате которой робот принимает начальную конфигурацию (позу). Вектор углов сочленений инициализируется начальными значениями, соответствующими матрицам M для каждой из конечностей (Приложение D, Приложение F). Производится первичный расчёт отклонения показаний гироскопа $bias\_gyro$ и

формирование первоначального кватерниона положения, на основе которого формируется стартовая матрица вращения элемента преобразования T_{WB} для мировой системы координат. Устанавливается начальное значение для вектора p и формируется стартовое преобразование T_{WB}.

Показания сенсоров, полученные в текущий момент времени t, поступают в формирователь вектора наблюдения, где происходит нормирование данных и структуризация вектора наблюдения. Сенсорные данные используются для определения баланса, которое, совместно с показателями датчика столкновения формируют сигнал завершения эпизода взаимодействия со средой и инициализируют процесс обучения агента на полученном за время функционирования опыте. Блок определения сбалансированности взаимодействует с кинематической моделью, на основании которой производится получение текущих координат конечностей. Они используются для построения опорного многоугольника, относительно которого производится анализ расположения точек CoM и ZMP. Каждое выбранное действие агентом поступает на блок реализации, где на основании спрогнозированных корректировок, формируется новая конфигурация робота, которая обрабатывается кинематической моделью и реализуется исполнительным механизмом, на основе низкоуровневых команд соответствующего контроллера. Каждое изменение конфигурации порождает обратную связь в виде сигнала вознаграждения, который формируется генератором на основании текущих показаний сенсоров с учётом критических состояний. Весь полученный за эпизод взаимодействия опыт накапливается в буфере, который используется в блоке обучения агента, при генерации сигнала завершения эпизода. После процесса настройки агента производится очистка буфера опыта. В результате обучения, происходит обновление весовых коэффициентов нейронов сети-кластера, соответствующих выбранной группе. Цикл взаимодействия производится до отключения питания механизма.

4.7. Результаты экспериментов для алгоритма ликвидации критических состояний

Апробация предложенного в контексте исследования алгоритма ликвидации критических состояний производилась в сравнении с алгоритмом, который не учитывает их в контексте функционирования и считается базовым [38]. Сравнение производилось на базе ранее описанной робототехнической платформы, состоящей из двух педипуляторов и небольшого торса. Выбор конструкции обусловлен наибольшей нестабильностью, в сравнении с другими разновидностями шагающих платформ с большим количеством конечностей. Основной задачей робота являлась реализация прямолинейного движения по ровной поверхности. Экспериментальные данные были получены на 6000 эпизодах. Длина эпизода соответствовала 300 шагов. Параметр скорости обучения сетей исполнителя и критика соответствовал 0,0001. Количество эпох на итерацию равнялось 4. В качестве метрик качества для реализации сравнения были выбраны: *процент эпизодов без фактического падения, процент полностью завершённых эпизодов, максимальная длина эпизода, количество шагов до падения и доля случаев, когда робот был в падении, но восстановился*. В результате экспериментов были получены следующие результаты, представленные в таблице 4.2.

Таблица 4.2 – Полученные показатели метрик качества

Метрика	Классический	Предлагаемый
Процент эпизодов без фактического падения	65%	85%
Процент полностью завершённых эпизодов	65%	80%
Максимальная длина эпизода	300	300
Количество шагов до падения	300	1000
Доля случаев, когда робот был в падении, но восстановился	5%	55%

В результате было получено, что процент эпизодов без падения вырос, по сравнению с применением базового алгоритма примерно на двадцать процентов. Наблюдается увеличение количества шагов до падения более чем в три раза, по сравнению с базовым алгоритмом. Заметен рост значения метрики доли случаев, когда робот был в падении, но восстановился. Алгоритм ликвидации критических состояний обеспечил прирост более чем на одиннадцать процентов по сравнению с базовым алгоритмом. Максимальная длина эпизода одинакова для обоих случаев.

Метрика процента полностью завершённых эпизодов показывает процент эпизодов, в которых агент достигает горизонта. В результате наблюдается прирост на двадцать процентов метрики при использовании алгоритма ликвидации, по сравнению с базовым алгоритмом. Метрика процента полностью завершённых эпизодов для алгоритма ликвидации на пять процентов меньше, по сравнению с метрикой процента эпизодов без падения для того же алгоритма, поскольку вторая метрика учитывает завершения эпизода по достижении $C_{\max}$.

Из полученных экспериментальных данных следует, что предлагаемый алгоритм ликвидации критических состояний является эффективным средством, обеспечивающим минимизацию коллизий, возникающих в процессе управления функционированием агента, реализующего локомоцию шагающего робота. Фокусирование обучения на области критических состояний, реализуемое путём установки стартового состояния в расположенное на $-k$ шагов от терминального, позволяет подробно исследовать критическую область. Это способствует лучшему предсказанию вероятности падения при текущем состоянии среды, и учитывать это в процессе реализации управления.

Данные эффекты наблюдаются на графиках распределения случайной величины суммарного дохода за эпизод, полученные путём усреднения по десяти запускам и представленные на рисунке 4.5.

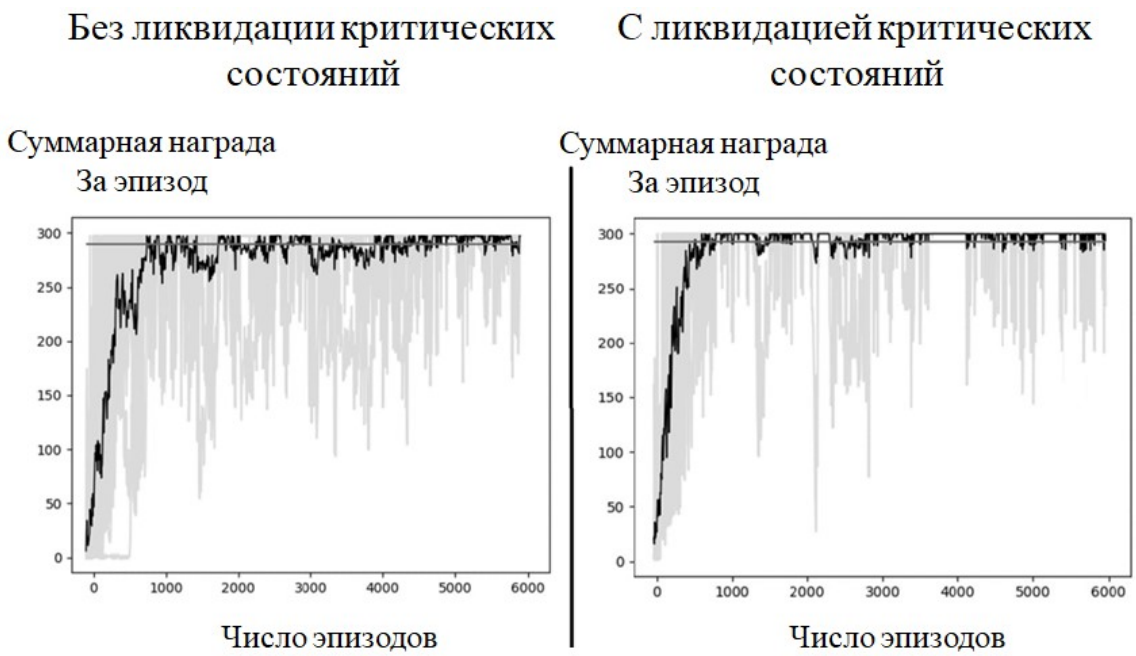


Рисунок 4.5 – Графики распределения случайной величины суммарного дохода за эпизод. Слева для базового алгоритма, справа с учётом алгоритма ликвидации критических состояний

4.8. Результаты экспериментов по применению нейронной сети-кластера и переключающих нейронов для обеспечения многофункциональности агента

Ранее обсуждалось, что в контексте настоящего исследования рассматривается локомоторный паттерн «движение прямо» на двух типах поверхностей: ровной и песчаной. Разделение на две разные стратегии обусловлено предположением о различии реализации соответствующего движения на разнообразных типах рельефа местности. Для подтверждения работоспособности переключения между стратегиями посредством переключающих нейронов внутри нейронного кластера необходимо произвести сравнение функционирования робота на не соответствующих выбранной программе условиях. Если стратегия движения по ровной поверхности, при переносе робота на песчаную поверхность,

продемонстрирует заметное изменение стратегии, а с возвратом в первоначальные условия полное восстановление, аналогично со стратегией движения по песку, то это означает то, что механика переключения выполняется корректно.

Сначала обучалась стратегия движения по ровной поверхности, с указанием соответствующего данной программе кода. Далее, изменив код операции, производилось обучение второй стратегии перемещения по песчаной поверхности. Рассчитывалась базовая метрика для каждого случая, в качестве которой был использован *процент эпизодов без падения*. На этом подготовительный этап был завершён. На втором этапе экспериментов робот с указанием кода стратегии для движения по ровной поверхности был перемещён на песчаную поверхность. Аналогичные действия были предприняты для движения по песчаной поверхности, где агент с кодом движения по песчаной поверхности был помещён на ровную поверхность. Для каждого случая был произведён расчёт метрики. Полученные результаты проведённых экспериментов представлены в таблице 4.3.

Таблица 4.3 – Метрика процента эпизодов без падения

Стратегия обучения	Переход	Процент эпизодов без падения
Прямо по ровной поверхности	Ровная - Ровная	85
Прямо по песчаной поверхности	Песчаная - Песчаная	80
Прямо по ровной поверхности	Ровная - Песчаная	40
Прямо по песчаной поверхности	Песчаная - Ровная	85

Из полученных результатов видно, что при переходе обученной стратегии движения по ровной поверхности на песчаную, заметно значительное снижение количества эпизодов без падений (более чем в 2 раза). Это связано с тем, что данная стратегия не учитывает проскальзывание и делает агрессивный шаг. В отличие от неё, стратегия обученная движению по песку более осторожна и учитывает вязкость. Поэтому при переходе на

ровную поверхность заметно возрастание процента эпизодов без падения. Таким образом, каждая группа корректно аппроксимирует соответствующую ей стратегию. Выполняется корректная селекция стратегий.

В результате, данная серия экспериментов свидетельствует о фактической реализации переключения между стратегиями и подтверждает корректность предлагаемых в контексте исследования механизма переключения посредством переключающих нейронов и концепции групп внутри нейронного кластера.

4.9. Выводы

1. Для управления агентом, реализующим формирование локомоции киберфизической системы мобильного шагающего робота, требуется специализированная система управления, способная генерировать управляющие сигналы, обеспечивающие многофункциональность, корректную реализацию каждой локомоторной программы и безопасную адаптацию механизма к условиям окружающей среды без дополнительных настроек и конфигурации.

2. Разработанное математическое и алгоритмическое обеспечение, основанные на введенной в контексте исследования концепции критических состояний, обеспечивают безопасное обучение и функционирование агента, с минимизацией возникновения коллизий.

3. Разработанная архитектура агента, состоящая из нейронного кластера с переключающими нейронами, обеспечивает решение проблематики низкой репрезентативной способности глубокой нейронной сети и реализует возможность множественной обработки разнообразных стратегий в контексте одного агента, на котором базируется разработанная система управления.

4. Разработанная структура системы управления агентом, реализующим формирование локомоции мобильного шагающего робота, обеспечивает многофункциональность и безопасное выполнение локомоторных стратегий с минимизацией возникновения коллизий за счёт предсказания возможности падения.

5. Проведённые эксперименты, на примере шагающего робота с двумя педипуляторами, демонстрируют эффективность разработанной системы управления с точки зрения безопасного обучения и обеспечения множественности запоминания и выполнения разнообразных стратегий с селекцией, основанной на переключающих нейронах.

6. Произведена регистрация элементов разработанной системы управления в ФИПС [17].

ЗАКЛЮЧЕНИЕ

В процессе выполнения диссертационного исследования были получены следующие основные результаты:

1. Выполнен анализ существующих подходов к построению систем управления интеллектуальным агентом на основе глубокого обучения с подкреплением, включая проблематику управления локомоцией шагающих роботов, как частного случая киберфизических систем, в контексте поиска наиболее подходящего для реализации цели построения адаптивного автономного управления.

2. Разработано специальное математическое обеспечение системы управления интеллектуальным агентом, обеспечивающее повышение уровня безопасности и устойчивости функционирования управляемой киберфизической системы.

3. Разработано специальное алгоритмическое обеспечение системы управления интеллектуальным агентом, обеспечивающее ускорение обучения и переноса агента на реальную систему за счёт увеличения времени устойчивого функционирования и сокращения числа нештатных режимов (коллизий).

4. Разработана архитектура глубокой нейронной сети, обеспечивающая агента, основанного на глубоком обучении с подкреплением, возможностью обучения нескольким стратегиям в контексте одной глубокой нейронной сети и ликвидирующей эффект «забывания» и потерю деталей каждой из стратегий.

5. Разработана структура системы управления интеллектуальным агентом, обеспечивающая автономное адаптивное безопасное обучение множеству стратегий с последующим переносом на реальную киберфизическую систему и безопасную настройку на примере формирования локомоции шагающего робота.

6. Разработанные элементы программного обеспечения системы

управления локомоцией мобильного шагающего робота зарегистрированы в ФИПС.

7. Выполнен цикл экспериментов по определению эффективности разработанного математического и алгоритмического обеспечения безопасного обучения агента и предложенной архитектуры глубокой нейронной сети на примере решения задачи управления локомоцией киберфизической системы мобильного шагающего робота. Результаты экспериментов продемонстрировали увеличение устойчивости агента на 20%, при использовании алгоритма ликвидации критических состояний в сравнении с базовым алгоритмом, не учитывающим соответствующие состояния. Так же наблюдался значительный прирост значения метрики «количество шагов до падения» и «доля случаев, когда робот был в падении, но восстановился».

Рекомендации и перспективы дальнейшей разработки темы

1. Результаты исследования целесообразно применять при проектировании адаптивных автономных систем управления, использующих алгоритмы обучения с подкреплением, основанные на глубоких нейронных сетях. Предложенная архитектура многофункционального агента и концепция критических состояний применимы для решения задач управления в областях интеллектуальной робототехники и построения систем искусственного интеллекта.

2. Дальнейшую разработку темы следует направить на изучение применения концепции нейронного кластера к другим типам глубоких сетей, таким как рекуррентные нейронные сети и сети с памятью (LSTM, GRM). Также следует рассмотреть задачу классификации окружающей среды для построения высокоуровнего генератора кодов селектора стратегий.

СПИСОК ЛИТЕРАТУРЫ

1. Аветисян, Т. В. Алгоритмы моделирования для оценки рисков в киберфизических системах / Т. В. Аветисян, Я. Е. Львович, А. П. Преображенский // Телекоммуникации. — 2023. — № 2. — С. 9–15.

2. Аветисян, Т. В. Анализ возможностей построения рациональной структуры киберфизической системы / Т. В. Аветисян, Я. Е. Львович, А. П. Преображенский // Моделирование, оптимизация и информационные технологии. — 2023. — Т. 11, № 1 (40). — С. 18–19.

3. Аветисян, Т. В. Исследование возможностей оптимизации процессов управления киберфизическими системами / Т. В. Аветисян, Я. Е. Львович, А. П. Преображенский, Ю. П. Преображенский // Информационные технологии и вычислительные системы. — 2023. — № 2. — С. 96–105.

4. Аветисян, Т. В. Моделирование и оптимизация модулей и информационных массивов в киберфизической системе / Т. В. Аветисян, Я. Е. Львович, А. П. Преображенский, Ю. П. Преображенский // Известия Кабардино Балкарского научного центра РАН. — 2023. — № 6 (116). — С. 116–124.

5. Аветисян, Т. В. Моделирование киберфизических систем при их развитии / Т. В. Аветисян, Я. Е. Львович, А. П. Преображенский // Системы управления и информационные технологии. — 2023. — № 1 (91). — С. 23–27.

6. Аветисян, Т. В. Оптимизация процессов в киберфизических системах / Т. В. Аветисян, Я. Е. Львович, А. П. Преображенский // Известия высших учебных заведений. Приборостроение. — 2023. — Т. 66, № 5. — С. 389–398.

7. Аветисян, Т. В. Проблемы управления киберфизической системой при наличии в ней неисправностей / Т. В. Аветисян, А. П. Преображенский, Ю. П. Преображенский // International Journal of Advanced Studies. — 2023. — Т. 13, № 4. — С. 7–21.

8. Акимов, А. А. Использование методов Монте Карло в обучении с подкреплением / А. А. Акимов, Е. С. Малагина // Научное обозрение. — 2022. — № 6. — С. 5–11.

9. Алексеев, Р. А. Алгоритмы управления движением шагающего робота / Р. А. Алексеев, И. В. Мирошник // Научно технический вестник информационных технологий, механики и оптики. — 2005. — № 19. — С. 67–75.

10. Анципорович, П. П. Структура механизмов : методическое пособие к лабораторным работам по дисциплине «Теория механизмов, машин и манипуляторов» / П. П. Анципорович, В. К. Акулич, Е. М. Дубовская. — 3 е изд. — Минск : БНТУ, 2017. — 33 с. — ISBN 978-985-583-044-4.

11. Бжихатлов, И. А. Исследование колебаний платформы двуногого шагающего робота / И. А. Бжихатлов, В. С. Громов // Известия высших учебных заведений. Приборостроение. — 2020. — Т. 63, № 3. — С. 278–285.

12. Бобырь, М. В. Автоматизированная система определения расстояния до объекта / М. В. Бобырь, А. А. Дородных, А. Ю. Алтухов // Промышленные АСУ и контроллеры. — 2023. — № 10. — С. 3–11.

13. Бобырь, М. В. Исследование свойств волнового алгоритма для нахождения маршрута передвижения роботов / М. В. Бобырь, А. Г. Крюков // Промышленные АСУ и контроллеры. — 2023. — № 9. — С. 21–29.

14. Борисов, О. И. Методы управления робототехническими приложениями : учебное пособие / О. И. Борисов, В. С. Громов, А. А. Пыркин. — Санкт Петербург : Университет ИТМО, 2016. — 108 с.

15. Вольф, Д. А. Интерактивный четвероногий робот класса фелидов с блоком нейронной обработки / Д. А. Вольф, Р. В. Мещеряков, А. О. Исхакова // Мехатроника, автоматизация, управление. — 2023. — Т. 24, № 10. — С. 542–550.

16. Гафаров, Ф. М. Искусственные нейронные сети и приложения : учебное пособие / Ф. М. Гафаров, А. Ф. Галимянов. — Казань : Изд-во Казан. ун-та, 2018. — 121 с.

17. Гребенникова Н. И., Олейникова С. А., Кашко В. В. Интеллектуальная система поддержки принятия решений на базе аппарата искусственных нейронных сетей : свидетельство о регистрации программы для ЭВМ № 2023665763 / Федеральный институт промышленной собственности. — Москва, 2023. — Дата регистрации: 02.07.2023

18. Грессер, Л. Глубокое обучение с подкреплением: теория и практика на языке Python / Л. Грессер, Ван Лун Кенг. — Санкт Петербург : Питер, 2022. — 416 с. : ил. — (Серия «Библиотека программиста»). — ISBN 978-5-4461-1699-7.

19. Деменева, С. Б. Решение прямой задачи кинематики для структуры типа гексапод / С. Б. Деменева, О. Р. Ачкасов // Актуальные проблемы авиации и космонавтики. — 2018. — Т. 1, № 14. — С. 24–26.

20. Дубенко, Ю. В. Алгоритм обучения с подкреплением для децентрализованных многоагентных систем, основанный на обмене опытом и обучении агентов случайному взаимодействию / Ю. В. Дубенко, Н. А. Рудешко // Вестник Воронежского государственного технического университета. — 2022. — Т. 18, № 4. — С. 30–36.

21. Зенкевич, С. Л. Основы управления манипуляционными роботами : учебник для вузов / С. Л. Зенкевич, А. С. Ющенко. — Москва : Изд во МГТУ им. Баумана, 2004. — 480 с.

22. Кашко, В. В. Анализ качества решения задачи управления интеллектуальным агентом на основе алгоритма градиента стратегии REINFORCE с точки зрения особенностей окружающей среды / В. В. Кашко // Управление программным инжинирингом : труды открытой Международной научно практической конференции. — 2025. — С. 44–49.

23. Кашко, В. В. Анализ методов обучения с подкреплением для управления роботизированными системами / В. В. Кашко, С. А. Олейникова // Инновационные технологии: теория, инструменты, практика. — 2024. — Т. 1. — С. 133–140.

24. Кашко, В. В. Анализ применения параметрических функций вознаграждения в алгоритмах обучения с подкреплением / В. В. Кашко, С. А. Олейникова, Ю. С. Акинина // Вестник Воронежского государственного технического университета. — 2024. — Т. 20, № 3. — С. 37–43.

25. Кашко, В. В. Анализ проблем обучения нейронных сетей и методов их устранения / В. В. Кашко, С. А. Олейникова // Информационные технологии моделирования и управления. — 2023. — Т. 132, № 2. — С. 124–128.

26. Кашко, В. В. Анализ сходимости и устойчивости алгоритма глубокого обучения с подкреплением исполнитель критик A2C / В. В. Кашко, С. А. Олейникова // Вестник Воронежского государственного технического университета. — 2024. — Т. 21, № 1. — С. 12–19.

27. Кашко, В. В. Архитектура нейронной сети многофункционального агента на базе глубокого обучения с подкреплением / В. В. Кашко, С. А. Олейникова // Вестник Воронежского государственного технического университета. — 2026. — Т. 22, № 1. — С. 35–44.

28. Кашко, В. В. Влияние dropout регуляризации глубокой сети на сходимость и устойчивость алгоритма градиента стратегии Advantage Actor Critic (A2C) / В. В. Кашко, С. А. Олейникова // Вестник Пензенского государственного университета. — 2025. — № 1(49). — С. 62–65.

29. Кашко, В. В. Влияние архитектуры глубокой нейронной сети на функционирование алгоритма градиента стратегии исполнитель критик A2C / В. В. Кашко, С. А. Олейникова // Актуальные проблемы прикладной математики, информатики и механики : сборник трудов Международной научной конференции. — 2025. — С. 233–238.

30. Кашко В.В., Олейникова С.А. Математическая модель универсальной системы управления шагающим роботом на основе методов обучения с подкреплением. Моделирование, оптимизация и информационные технологии. 2024;12(1). URL: https://moitvivt.ru/ru/journal/pdf?id=1520 DOI: 10.26102/2310-6018/2024.44.1.025.

31. Кашко, В. В. Обобщённый алгоритм решения задачи управления шагающим роботом на базе интеллектуального агента с использованием методов глубокого обучения с подкреплением / В. В. Кашко, С. А. Олейникова // Научная опора Воронежской области : сборник трудов победителей конкурса научно исследовательских работ студентов и аспирантов ВГТУ по приоритетным направлениям развития науки и технологий. — Воронеж, 2025. — С. 155–158.

32. Кашко, В. В. Общий алгоритм ликвидации критических состояний для решения задачи управления реальным шагающим роботом на основе методов глубокого обучения с подкреплением / В. В. Кашко, С. А. Олейникова // Программные системы и вычислительные методы. — Москва, 2025. — № 3. — С. 103–114.

33. Кашко, В. В. Построение многофункционального агента на базе глубокого обучения с подкреплением / В. В. Кашко, С. А. Олейникова // Актуальные проблемы прикладной математики, информатики и механики : сборник трудов Международной научной конференции. — 2025. — С. 285–291.

34. Кашко, В. В. Применение методов обучения с подкреплением для реализации движения шагающих роботов / В. В. Кашко, С. А. Олейникова // Современные информационные технологии. Теория и практика. — 2024. — С. 256–262.

35. Кашко, В. В. Разработка подхода для сопряжения нейронных групп внутри полносвязной сети кластера с использованием нейронов повторителей / В. В. Кашко, С. А. Олейникова // ИННОТЕХ. — 2026.

36. Кашко, В. В. Сравнительный анализ алгоритмов A2C и PPO с различными архитектурами нейронных сетей на примере среды CartPole / В. В. Кашко, С. А. Олейникова // Вестник Воронежского государственного университета. — 2026. — Т. 20, № 3. — С. 37–43.

37. Кашко, В. В. Формализация задачи управления шагающим роботом на основе алгоритмов обучения с подкреплением / В. В. Кашко, С. А. Олейникова // Интеллектуальные информационные системы : труды Международной научно практической конференции. — Воронеж, 2025. — С. 243–247.

38. Кашко, В. В. Экспериментальное обоснование эффективности разработанной системы управления шагающим роботом / В. В. Кашко, С. А. Олейникова // Конференция по искусственному интеллекту. — Воронеж, 2026.

39. Кошманова, Н. П. Управление манипулятором с помощью обучения с подкреплением / Н. П. Кошманова, Д. С. Трифонов, В. Е. Павловский // Russian Journal of Nonlinear Dynamics. — 2012. — Т. 8, № 4. — С. 689–704.

40. Лапина, Н. А. Анализ методов управления кинематикой и динамикой робототехнических систем / Н. А. Лапина, М. Е. Королев // Проблемы науки. — 2017. — № 1 (83). — URL: https://cyberleninka.ru/article/n/analiz-metodov-upravleniya-kinematikoy-i-dinamikoy-robototehnicheskih-sistem (дата обращения: 23.07.2025).

41. Лонца, А. Алгоритмы обучения с подкреплением на Python / А. Лонца ; пер. с англ. А. А. Слинкина. — Москва : ДМК Пресс, 2020. — 286 с. : ил. — ISBN 978-5-97060-855-5.

42. Львович, И. Я. Оптимизация принятия структурно компонентных проектных решений в САПР киберфизических систем / И. Я. Львович, А. П. Преображенский, Я. Е. Львович // Моделирование, оптимизация и информационные технологии. — 2024. — Т. 12, № 2 (45).

43. Львович, Я. Е. Моделирование и оптимизация процессов управления информационно телекоммуникационными системами / Я. Е. Львович, А. П. Преображенский, Ю. П. Преображенский, Т. В. Аветисян // Электромагнитные волны и электронные системы. — 2024. — Т. 29, № 3. — С. 41–48.

44. Лю, Ю. (Х.) Обучение с подкреплением на PyTorch: сборник рецептов / Ю. (Х.) Лю ; пер. с англ. А. А. Слинкина. — Москва : ДМК Пресс, 2020. — 282 с. : ил. — ISBN 978-5-97060-853-1.

45. Мещеряков, Р. В. Интеллектуальные робототехнические системы // Датчики и системы. — 2024. — № 2 (274). — С. 12–17.

46. Мещеряков, Р. В. Исследование индикаторов компрометации для средств защиты информационных и киберфизических систем / Р. В. Мещеряков, С. Ю. Исхаков // Вопросы кибербезопасности. — 2022. — № 5 (51). — С. 82–99.

47. Мещеряков, Р. В. Исследование методов формирования индикаторов компрометации от внутренних источников информационных и киберфизических систем / Р. В. Мещеряков, С. Ю. Исхаков // Вопросы кибербезопасности. — 2023. — № 6 (58). — С. 35–49.

48. Модернизация трансмиссии гусеничного скиддера / Р. Ю. Добрецов, А. П. Чайкин, С. А. Войнаш [и др.] // Аграрный научный журнал. — 2024. — № 9. — С. 114–120. — DOI: 10.28983/asj.y2024i9pp114-120.

49. Моралес, М. Грокаем глубокое обучение с подкреплением : учебное пособие / М. Моралес. — Санкт Петербург : Питер, 2023. — 464 с. : ил. — (Серия «Библиотека программиста»). — ISBN 978-5-4461-3944-6.

50. Мунасыпов, Р. А. Иерархический регулятор на основе алгоритма обучения с подкреплением для реконфигурируемого многомодульного шагающего мобильного робота / Р. А. Мунасыпов, Т. Р. Шахмаметьев, С. С. Москвичев, И. Х. Хамадеев // Вестник Уфимского государственного авиационного технического университета. — 2013. — Т. 17, № 5 (58). — С. 31–37.

51. Носкова, О. Е. Теория механизмов и машин [Электронный ресурс] : учебное пособие / О. Е. Носкова ; Красноярский государственный аграрный университет. — Красноярск, 2021. — 200 с.

52. Павловский, В. Е. О разработках шагающих машин / В. Е. Павловский // Препринты ИПМ им. М. В. Келдыша. — 2013. — № 101. — 32 с. — URL: http://library.keldysh.ru/preprint.asp?id=2013-101 .

53. Преображенский, А. П. Моделирование процессов в киберфизической системе с использованием имитационного подхода / А. П. Преображенский, Т. В. Аветисян, Ю. П. Преображенский // International Journal of Advanced Studies. — 2024. — Т. 14, № 4. — С. 46–64.

54. Самойлова, А. С. Система управления шагающим мобильным роботом с использованием генетического алгоритма / А. С. Самойлова, П. А. Груничев, С. А. Воротников // Известия Тульского государственного университета. Технические науки. — 2020. — № 10. — С. 153–162.

55. Самойлова, А. С. Система управления шагающим роботом, адаптивным к изменению кинематической схемы / А. С. Самойлова, С. А. Воротников // Мехатроника, автоматизация, управление. — Москва : Новые технологии, 2021. — Т. 22 : Роботы, мехатроника и робототехнические системы. — № 11. — С. 601–609.

56. Саттон, Р. С. Обучение с подкреплением : введение / Р. С. Саттон, Э. Барто ; пер. с англ. — 2 е изд. — Москва : ДМК Пресс, 2020. — 552 с. : ил. — ISBN 978-5-97060-097-9.

57. Сергеев, А. П. Введение в нейросетевое моделирование : учебное пособие / А. П. Сергеев, Д. А. Тарасов ; под общ. ред. А. П. Сергеева. — Екатеринбург : Изд-во Урал. ун-та, 2017. — 128 с. — ISBN 978-5-7996-2124-7.

58. Сиволобов, С. В. Математическое моделирование походки человека на основе пятизвенной модели антропоморфного механизма с использованием методов оптимизации / С. В. Сиволобов // Математическая физика и компьютерное моделирование. — 2024. — Т. 27, № 1. — С. 62–85. — DOI: 10.15688/mpcm.jvolsu.2024.1.5.

59. Сидоренко, А. В. Обучение с подкреплением при навигации мобильных роботов / А. В. Сидоренко // Вестник Полоцкого государственного университета. Серия С. Фундаментальные науки. — 2021. — № 12. — С. 21–24.

60. Стерехова, В. С. Кинематический анализ антропоморфного робота DARwin OP / В. С. Стерехова, А. А. Протопопов ; науч. рук. Е. Е. Шеломенцев // Молодёжь и современные информационные технологии : сб. тр. XIV Междунар. науч.-практ. конф. студентов, аспирантов и молодых учёных (г. Томск, 7–11 ноября 2016 г.) : в 2 т. — Томск : Изд во ТПУ, 2016. — Т. 1. — С. 341–342.

61. Стивенс, Э. PyTorch. Освещая глубокое обучение : учебное пособие / Э. Стивенс, Л. Антига, Т. Виман. — Санкт Петербург : Питер, 2022. — 576 с. : ил. — (Библиотека программиста). — ISBN 978-5-4461-1945-5.

62. Страшнов, Е. В. Моделирование полуавтоматического режима управления движением двуногих шагающих роботов в системах виртуального окружения / Е. В. Страшнов, Л. А. Финагин // Автоматизация и моделирование в проектировании и управлении. — 2022. — № 3 (17). — С. 57–67.

63. Траск, Э. Грокаем глубокое обучение : учебное пособие / Э. Траск. — Санкт Петербург : Питер, 2021. — 352 с. : ил. — (Библиотека программиста). — ISBN 978-5-4461-1334-7.

64. Уиндер, Ф. Обучение с подкреплением для реальных задач / Ф. Уиндер ; пер. с англ. — Санкт Петербург : БХВ Петербург, 2023. — 400 с. : ил. — ISBN 978-5-9775-6885-2.

65. Фокин, В. Г. Структура системы управления шестиногого шагающего робота Гексабот / В. Г. Фокин, С. В. Шаныгин // Вестник евразийской науки. — 2016. — Т. 8, № 5 (36). — С. 103.

66. Хайкин, С. Нейронные сети: полный курс. 2 е изд. / С. Хайкин ; пер. с англ. — Москва : Вильямс, 2006. — 1104 с. : ил. — ISBN 5-8459-0890-6.

67. Харбанс, Р. Грокаем алгоритмы искусственного интеллекта / Р. Харбанс. — Санкт Петербург : Питер, 2023. — 368 с. : ил. — (Серия «Библиотека программиста»). — ISBN 978-5-4461-2924-9.

68. Хусаинов, Р. Р. Оптимизация параметров движения двуногого шагающего робота / Р. Р. Хусаинов, А. С. Климчик, Е. А. Магид // Известия Волгоградского государственного технического университета. — 2018. — № 13 (223). — С. 119–125.

69. Шарибаев, А. Н. Проблемы в области глубокого обучения с подкреплением / А. Н. Шарибаев, Р. Н. Шарибаев, Б. Т. Абдулазизов, М. Р. Тохиржонова // Форум молодых учёных. — 2023. — № 6 (82). — URL: https://cyberleninka.ru/article/n/problemy-v-oblasti-glubokogo-obucheniya-s-podkrepleniem (дата обращения: 05.04.2024).

70. Юревич, Е. И. Основы робототехники : учебное пособие / Е. И. Юревич. — 4 е изд., перераб. и доп. — Санкт Петербург : БХВ Петербург, 2017. — 304 с. : ил. — (Учебная литература для вузов). — ISBN 978-5-9775-3851-0.

71. Achiam, J. Constrained policy optimization / J. Achiam, D. Held, A. Tamar, P. Abbeel // Proceedings of the 34th International Conference on Machine Learning. — 2017. — Vol. 70. — P. 22–31.

72. Andrew A.M. Another efficient algorithm for convex hulls in two dimensions // Information Processing Letters. – 1979. – Vol. 9. – Pp. 216–219.

73. Arbulu M., Kaynov D., Balaguer C. The Rh 1 Full Size Humanoid Robot: Control System Design and Walking Pattern Generation // Book chapter / IntechOpen. — 2010. — DOI: 10.5772/10422.

74. Azayev, T. Blind hexapod locomotion in complex terrain with gait adaptation using deep reinforcement learning and classification / T. Azayev, K. Zimmerman // Journal of Intelligent & Robotic Systems. — 2020. — Vol. 99. — P. 659–671. — DOI: 10.1007/s10846-020-01162-8.

75. Bal, C. Neural coupled central pattern generator based smooth gait transition of a biomimetic hexapod robot / C. Bal // Neurocomputing. — 2021. — Vol. 420. — P. 210–226. — DOI: 10.1016/j.neucom.2020.07.114.

76. Barfoot, T. D. Experiments in learning distributed control for a hexapod robot // Robot. Auton. Syst. — 2006. — Vol. 54. — P. 864–872. — DOI: 10.1016/j.robot.2006.04.009.

77. Blind bipedal stair traversal via sim-to-real reinforcement learning / J. Siekmann, K. Green, J. Warila [et al.] : arXiv preprint arXiv:2105.08328. — 2021.

78. Capi, G. Application of Genetic Algorithms for biped robot gait synthesis optimization during walking and going up stairs / G. Capi, Y. Nasu, L. Barolli, K. Mitobe, K. Takeda // Advanced Robotics. — 2001. — Vol. 15. — P. 675–694. — DOI: 10.1163/156855301317035197.

79. Chalodhorn R., Grimes D., Grochow K., Rao R. Learning to Walk through Imitation // Conference proceedings. — 2007. — P. 2084–2090.

80. Chignoli, M. The MIT humanoid robot: design, motion planning, and control for acrobatic behaviors / M. Chignoli, D. Kim, E. Stanger Jones, S. Kim //

2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids). — IEEE, 2021. — P. 1–8.

81. Chung, H.-Y. Hexapod moving in complex terrains via a new adaptive CPG gait design / H.-Y. Chung, C. C. Hou, S. Y. Hsu // Industrial Robot. — 2015. — Vol. 42. — P. 129–141. — DOI: 10.1108/IR-10-2014-0403.

82. Cobbe, K. Quantifying generalization in reinforcement learning / K. Cobbe, O. Klimov, C. Hesse, T. Kim, J. Schulman. — 2018. — arXiv:1812.02341.

83. Dallali, H. On Global Optimization of Walking Gaits for the Compliant Humanoid Robot COMAN Using Reinforcement Learning / H. Dallali, P. Kormushev, Z. Li [et al.] // International Journal of Cybernetics and Information Technologies. — 2012. — Vol. 12. — DOI: 10.2478/cait-2012-0020.

84. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills / X. B. Peng, P. Abbeel, S. Levine, M. Van de Panne // ACM Transactions On Graphics (TOG). — 2018. — Vol. 37, № 4. — P. 1–14.

85. Distributed learning of decentralized control policies for articulated mobile robots / G. Sartoretti, W. Paivine, Y. Shi [et al.] // IEEE Trans. Robot. — 2019. — Vol. 35. — P. 1109–1122. — DOI: 10.1109/TRO.2019.2922493.

86. Dubenko, Y. Multi Agent Reinforcement Learning for Robot Collaboration / Y. Dubenko, E. Dyshkant, D. Gura // Smart Innovation, Systems and Technologies. — 2022. — Vol. 247. — P. 607–623.

87. Falasconi, A. Dynamic basal ganglia output signals license and suppress forelimb movements / A. Falasconi, H. Kanodia, S. Arber // Nature. — 2025. — T. 644. — C. 749–758. — DOI: 10.1038/s41586-025-09066-z.

88. Finn, C. Guided cost learning: Deep inverse optimal control via policy optimization / C. Finn, S. Levine, P. Abbeel // Proceedings of the 33rd International Conference on Machine Learning (ICML). — 2016. — P. 49–58.

89. Fokin, V. G. The structure of the control system of the Hexabot six-legged walking robot / V. G. Fokin, S. V. Shanygin // Bulletin of Eurasian Science. — 2016. — Vol. 8, № 5 (36). — P. 103.

90. Fu, H. Deep Reinforcement Learning for Multi contact Motion Planning of Hexapod Robots / H. Fu, T. Kaiqiang, P. Li [et al.] // Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). — 2021. — P. 2381–2388. — DOI: 10.24963/ijcai.2021/328.

91. Fujimoto, S. Addressing Function Approximation Error in Actor Critic Methods / S. Fujimoto, H. Hoof, D. Meger. — 2018. — DOI: 10.48550/arXiv.1802.09477.

92. Gangapurwala, S. Guided constrained policy optimization for dynamic quadrupedal robot locomotion / S. Gangapurwala, A. Mitchell, I. Hacoutis // IEEE Robotics and Automation Letters. — 2020. — Vol. 5. — P. 3642–3649. — DOI: 10.1109/LRA.2020.2979656.

93. Geng, T. Fast Biped Walking with a Sensor driven Neuronal Controller and Real time Online Learning / T. Geng // The International Journal of Robotics Research. — 2006. — Vol. 25. — P. 243–259. — DOI: 10.1177/0278364906063822.

94. Gym Documentation. [Электронный ресурс] URL: https://www.gymlibrary.dev/ (дата обращения: 25.01.2025).

95. Haarnoja, T. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor / T. Haarnoja, A. Zhou, P. Abbeel, S. Levine. — 2018. — arXiv:1801.01290.

96. Hinton, G. Distilling the Knowledge in a Neural Network / G. Hinton, O. Vinyals, J. Dean // CoRR. — 2015. — abs/1503.02531.

97. Hrdlicka, I. Reinforcement learning in control systems for walking hexapod robots / I. Hrdlicka, P. Kutilek // Cybernetic Letters. — 2005. — Vol. 3. — P. 1–13.

98. Huan, T. Optimal stable gait for nonlinear uncertain humanoid robot using central force optimization algorithm / T. Huan, H. Pham Huy Anh // Engineering Computations. — 2019. — Vol. 36. — DOI: 10.1108/EC-03-2018-0154.

99. Hwangbo, J. Learning Agile and Dynamic Motor Skills for Legged Robots / J. Hwangbo, J. Lee, A. Dosovitskiy [et al.] // Science Robotics. — 2019. — Vol. 4. — Art. eaau5872. — DOI: 10.1126/scirobotics.aau5872.

100. Ijspeert, A. J. Central pattern generators for locomotion control in animals and robots: a review // Neural Networks. — 2008. — Vol. 21, № 4.

101. Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors / S. Bohez, S. Tunyasuvunakool, P. Brakel [et al.] : arXiv preprint arXiv:2203.17138. — 2022.

102. Ishihara, K. Full-body optimal control toward versatile and agile behaviors in a humanoid robot / K. Ishihara, T. D. Itoh, J. Morimoto // IEEE Robotics and Automation Letters. — 2019. — Vol. 5, № 1. — P. 119–126.

103. Kajita, S. Dynamic Walk Control of a Biped Robot with Potential Energy Conserving Orbit / S. Kajita, T. Yamaura, A. Kobayashi // IEEE Transactions on Robotics and Automation. — 1992. — Vol. 23. — P. 431–438. — DOI: 10.1109/70.149940.

104. Kakade, S. M. A natural policy gradient / S. M. Kakade // Advances in Neural Information Processing Systems / ed. by T. Dietterich, S. Becker, Z. Ghahramani. — Cambridge : MIT Press, 2001. — Vol. 14.

105. Kamioka, T. Dynamic gait transition between walking, running and hopping for push recovery / T. Kamioka, H. Kaneko, M. Kuroda [et al.] // 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids). — IEEE, 2017. — P. 1–8.

106. Kang D., Vincenti F., Adami N., Coros S. Animal Motions on Legged Robots Using Nonlinear Model Predictive Control // Conference proceedings. — 2022. — P. 11955–11962. — DOI: 10.1109/IROS47612.2022.9981945.

107. Kang D., Zimmermann S., Coros S. Animal Gaits on Quadrupedal Robots Using Motion Matching and Model Based Control // Conference proceedings. — 2021. — P. 8500–8507. — DOI: 10.1109/IROS51168.2021.9635838.

108. Kashko, V. V. Formalization of the task of controlling the movement of a walking robot / V. V. Kashko, S. A. Oleinikova // Anthropocentric sciences in education: challenges, transformations, resources. — 2024. — P. 342–345.

109. Kober, J. Reinforcement Learning in Robotics: A Survey / J. Kober, J. Bagnell, J. Peters // The International Journal of Robotics Research. — 2013. — Vol. 32. — P. 1238–1274. — DOI: 10.1177/0278364913495721.

110. Lecture 5: Совместное развитие сенсорики и робототехники [Электронный ресурс] : издание официальное / Интернет Университет Информационных Технологий (ИНТУИТ). — Москва, 2024. — URL: https://intuit.ru/en/studies/courses/22789/1324/lecture/33070?page=5 (дата публикации: 07.10.2024).

111. Lele, A. Learning to Walk: Spike Based Reinforcement Learning for Hexapod Robot Central Pattern Generation / A. Lele, Y. Fang, J. Ting [et al.] // IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS). — 2020. — P. [без указания страниц]. — DOI: 10.1109/AICAS48895.2020.9073987.

112. Leskov, A. Kinematic Description of Humanoid Robots Using Method of Block Matrices / A. Leskov, K. V. Bazhinova, E. Yu. Seliverstova // Herald of the Bauman Moscow State Technical University. Series Instrument Engineering. — 2018. — DOI: 10.18698/0236-3933-2018-6-102-111.

113. Levine, S. Guided policy search / S. Levine, V. Koltun // Proceedings of the 30th International Conference on Machine Learning (ICML). — 2013. — P. 1–9.

114. Lifelike agility and play on quadrupedal robots using reinforcement learning and generative pretrained models / L. Han, Q. Zhu, J. Sheng [et al.] : arXiv preprint arXiv:2308.15143. — 2023.

115. Lillicrap, T. Continuous control with deep reinforcement learning / T. Lillicrap, J. Hunt, A. Pritzel [et al.] // CoRR. — 2015.

116. Linar, Z. Numerical Solution Approach for the ROBOTIS OP2 Humanoid Hand Inverse Kinematics / Z. Linar, T. Tsoy, E. Magid [et al.] // Proceedings of the International Conference on Artificial Life and Robotics (ICAROB). — 27th ed. — 2022. — P. 682–685.

117. Luk, B. Using genetic algorithms to establish efficient walking gaits for an eight legged robot / B. Luk, S. Galt // International Journal of Systems Science. — 2001. — Vol. 32. — DOI: 10.1080/00207720117230.

118. Lynch, K. Modern Robotics: Mechanics, Planning, and Control / K. Lynch, F. Park. — 2024. — DOI: 10.1017/9781316661239.

119. Manglik, A. Adaptive gait generation for hexapod robot using Genetic Algorithm / A. Manglik, K. Gupta, S. Bhanot. — 2016. — P. 1–6. — DOI: 10.1109/ICPEICES.2016.7853681.

120. Maroger I., Stasse O., Watier B. Walking Human Trajectory Models and Their Application to Humanoid Robot Locomotion // Conference proceedings. — 2020. — P. 3465–3472. — DOI: 10.1109/IROS45743.2020.9341118.

121. Miura, K. Human-like walking with toe supporting for humanoids / K. Miura, M. Morisawa, F. Kanehiro [et al.] // 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE, 2011. — P. 4428–4435.

122. Mnih, V. Asynchronous Methods for Deep Reinforcement Learning / V. Mnih, A. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu. — 2016. — arXiv:1602.01783.

123. Mnih, V. Human-level control through deep reinforcement learning / V. Mnih, K. Kavukcuoglu, D. Silver [et al.] // Nature. — 2015. — Vol. 518. — P. 529–533. — DOI: 10.1038/nature14236.

124. Moro F., Badri Spröwitz A., Tuleu A., Vespignani M., Tsagarakis N., Ijspeert A. J., Caldwell D. Horse like walking, trotting, and galloping derived from kinematic Motion Primitives (kMPs) and their application to walk/trot transitions

in a compliant quadruped robot // Biological Cybernetics. — 2013. — Vol. 107. — DOI: 10.1007/s00422 013 0551 9.

125. Nemoto T., Mohan R. E., Iwase M. Realization of rolling locomotion by a wheel spider inspired hexapod robot // Robotics and Biomimetics. — 2015. — Vol. 2. — DOI: 10.1186/s40638 015 0026 7.

126. Ng, A. Y. Algorithms for inverse reinforcement learning / A. Y. Ng, S. J. Russell [et al.] // Proceedings of the 17th International Conference on Machine Learning (ICML). — 2000. — P. 663–670.

127. Optimizing bipedal locomotion for the 100 m dash with comparison to human running / D. Crowley, J. Dao, H. Duan [et al.] // 2023 IEEE International Conference on Robotics and Automation (ICRA). — IEEE, 2023. — P. 12 205–12 211.

128. Ouyang, W. Adaptive Locomotion Control of a Hexapod Robot via Bio Inspired Learning / W. Ouyang, H. Chi, J. Pang [et al.] // Frontiers in Neurorobotics. — 2021. — Vol. 15. — Art. 627157. — DOI: 10.3389/fnbot.2021.627157.

129. Paulo, J. Human gait pattern changes detection system: A multimodal vision-based and novelty detection learning approach / J. Paulo [et al.] // Biocybernetics and Biomedical Engineering. — 2017. — Vol. 37, № 4. — P. 701–717.

130. Peng, X. B. Sim-to-real transfer of robotic control with dynamics randomization / X. B. Peng, M. Andrychowicz, W. Zaremba, P. Abbeel // 2018 IEEE International Conference on Robotics and Automation (ICRA). — IEEE, 2018. — P. 3803–3810.

131. Peng, X. Learning Agile Robotic Locomotion Skills by Imitating Animals / X. Peng, E. Coumans, T. Zhang [et al.]. — 2020. — DOI: 10.48550/arXiv.2004.00784.

132. Precise switch: how the brain controls movements [Электронный ресурс] // Scientific Russia : сайт. — 2025. — URL:

https://scientificrussia.ru/articles/tocnyj-pereklucatel-kak-mozg-upravlaet-dvizeniami (дата публикации: 18.09.2025).

133. Ramdya P., Thandiackal R., Cherney R., Asselborn T., Benton R., Auke J., Ijspeert A. J., Floreano D. Climbing favours the tripod gait over alternative faster insect gaits // Nature Communications. — 2017. — Vol. 8. — DOI: 10.1038/ncomms14494.

134. Robust and versatile bipedal jumping control through multi-task reinforcement learning / Z. Li, X. B. Peng, P. Abbeel [et al.] : arXiv preprint arXiv:2302.09450. — 2023.

135. Run like a dog: Learning based whole-body control framework for quadruped gait style transfer / F. Yin, A. Tang, L. Xu [et al.] // 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE, 2021. — P. 8508–8514.

136. Rusu, A. Policy Distillation / A. Rusu, S. Colmenarejo, Ç. Gülçehre [et al.] // CoRR. — 2016. — abs/1511.06295.

137. Rusu, A. Progressive Neural Networks / A. Rusu, N. Rabinowitz, G. Desjardins [et al.]. — 2016. — DOI: 10.48550/arXiv.1606.04671.

138. Ryadchikov, I. Генетический алгоритм поиска параметров ПИД регуляторов системы стабилизации шагающего робота / I. Ryadchikov, A. Gusev, S. Sechenev, E. Nikulchev // Труды НГТУ им. Р. Е. Алексеева. — 2019. — С. 57–65. — DOI: 10.46960/1816-210X_2019_1_52.

139. Schilling, M. Decentralized Deep Reinforcement Learning for a Distributed and Adaptive Locomotion Controller of a Hexapod Robot / M. Schilling, K. Konen, F. Ohl [et al.]. — 2020. — DOI: 10.48550/arXiv.2005.11164.

140. Schulman, J. High Dimensional Continuous Control Using Generalized Advantage Estimation / J. Schulman, P. Moritz, S. Levine [et al.]. — 2015. — DOI: 10.48550/arXiv.1506.02438.

141. Schulman, J. High-Dimensional Continuous Control Using Generalized Advantage Estimation / J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel. — 2015. — arXiv:1506.02438.

142. Schulman, J. Proximal Policy Optimization Algorithms / J. Schulman, F. Wolski, P. Dhariwal [et al.]. — 2017. — DOI: 10.48550/arXiv.1707.06347.

143. Schulman, J. Trust Region Policy Optimization / J. Schulman, S. Levine, P. Moritz [et al.]. — 2015.

144. Sewak, M. Deep Q Network (DQN), Double DQN, and Dueling DQN: A Step Towards General Artificial Intelligence / M. Sewak // — 2019. — DOI: 10.1007/978-981-13-8285-7_8.

145. Shao, Y. Learning free gait transition for quadruped robots via phase-guided controller / Y. Shao, Y. Jin, X. Liu, W. He, H. Wang, W. Yang // IEEE Robotics and Automation Letters. — 2021. — Vol. 7, № 2. — P. 1230–1237.

146. Shimmyo S., Sato T., Ohnishi K. Biped Walking Pattern Generation by Using Preview Control Based on Three Mass Model // IEEE Transactions on Industrial Electronics. — 2013. — Vol. 60. — P. 5137–5147. — DOI: 10.1109/TIE.2012.2221111.

147. Siekmann, J. Sim-to-real learning of all common bipedal gaits via periodic reward composition / J. Siekmann, Y. Godse, A. Fern, J. Hurst // 2021 IEEE International Conference on Robotics and Automation (ICRA). — IEEE, 2021. — P. 7309–7315.

148. Singh, R. P. Learning bipedal walking on planned footsteps for humanoid robots / R. P. Singh, M. Benallegue, M. Morisawa [et al.] // 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids). — IEEE, 2022. — P. 686–693.

149. Smith, L. Learning and Adapting Agile Locomotion Skills by Transferring Experience / L. Smith, J. Kew, T. Li [et al.]. — 2023. — DOI: 10.48550/arXiv.2304.09834.

150. Solovyeva, E. Controlling System Based on Neural Networks with Reinforcement Learning for Robotic Manipulator = Система управления на основе нейронных сетей при обучении с подкреплением для робота манипулятора / E. Solovyeva, A. Abbas // Information and Control Systems. — 2020. — № 5. — P. 24–32. — DOI: 10.31799/1684-8853-2020-5-24-32.

151. Song, W. GeRM: A Generalist Robotic Model with Mixture-of-Experts for Quadruped Robot / W. Song, H. Zhao, P. Ding [et al.] // IEEE International Conference on Intelligent Robots and Systems (IROS). — 2024. — P. 11879–11886. — DOI: 10.1109/IROS58592.2024.10801816.

152. Song, Z. An optimal motion planning framework for quadruped jumping / Z. Song, L. Yue, G. Sun, Y. Ling, H. Wei, L. Gui, Y.-H. Liu // 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE, 2022. — P. 11366–11373.

153. Sugihara, T. 3D biped locomotion control including seamless transition between walking and running via 3D ZMP manipulation / T. Sugihara, K. Imanishi, T. Yamamoto, S. Caron // 2021 IEEE International Conference on Robotics and Automation (ICRA). — IEEE, 2021. — P. 6258–6263.

154. Tang A., Hiraoka T., Hiraoka N., Shi F., Kawaharazuka K., Kojima K., Okada K., Inaba M. HumanMimic: Learning Natural Locomotion and Transitions for Humanoid Robot via Wasserstein Adversarial Imitation // Conference proceedings. — 2024. — P. 13107–13114. — DOI: 10.1109/ICRA57147.2024.10610449.

155. Tobin, J. Domain randomization for transferring deep neural networks from simulation to the real world / J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel // 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE, 2017. — P. 23–30.

156. Torabi, F. Behavioral cloning from observation / F. Torabi, G. Warnell, P. Stone. — 2018. — arXiv:1805.01954.

157. Tsounis, V. DeepGait: planning and control of quadrupedal gaits using deep reinforcement learning / V. Tsounis, M. Alge, J. Lee // IEEE Robotics and Automation Letters. — 2020. — Vol. 5. — P. 3699–3706. — DOI: 10.1109/LRA.2020.2979660.

158. Vadakkepat, P. Genetic algorithm based optimal bipedal walking gait synthesis considering tradeoff between stability margin and speed / P. Vadakkepat, P. Kien // Robotica. — 2009. — Vol. 27. — P. 355–365. — DOI: 10.1017/S026357470800475X.

159. Van Hasselt, H. Deep Reinforcement Learning with Double Q Learning / H. Van Hasselt, A. Guez, D. Silver // Proceedings of the AAAI Conference on Artificial Intelligence. — 2015. — Vol. 30. — DOI: 10.1609/aaai.v30i1.10295.

160. Wang, Z. Dueling Network Architectures for Deep Reinforcement Learning / Z. Wang, N. Freitas, M. Lanctot. — 2015.

161. Wensing, P. M. High-speed humanoid running through control with a 3D-SLIP model / P. M. Wensing, D. E. Orin // 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). — IEEE, 2013. — P. 5134–5140.

162. Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning / R. J. Williams // Machine Learning. — 1992. — Vol. 8. — P. 229–256. — DOI: 10.1007/BF00992696.

163. Wulfmeier, M. Large-scale cost function learning for path planning using deep inverse reinforcement learning / M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, I. Posner // The International Journal of Robotics Research. — 2017. — Vol. 36, № 10. — P. 1073–1087.

164. Yilmazlar, E. Walking Pattern Generation and Control for a Bipedal Robot / E. Yilmazlar, H. Kuscu // Machines. Technologies. Materials. — 2021. — Vol. 15, Issue 3. — P. 99–102.

165. Yu, H. Enhancing adaptability with local reactive behaviors for hexapod walking robot via sensory feedback integrated central pattern generator / H. Yu, H. Gao, Z. Deng // Robotics and Autonomous Systems. — 2020. — Vol. 124. — Art. 103401. — DOI: 10.1016/j.robot.2019.103401.

166. Yu, T. Reinforcement learning and convolutional neural network system for firefighting rescue robot / T. Yu, Y. Chieh, H. Samani // MATEC Web of Conferences. — 2018. — Vol. 161. — Art. 03028. — DOI: 10.1051/matecconf/201816103028.

167. Zhang, F. Modular deep Q networks for sim-to-real transfer of visuo-motor policies / F. Zhang, J. Leitner, M. Milford, P. Corke. — 2016. — arXiv:1610.06781.

ПРИЛОЖЕНИЕ А
АКТЫ О ВНЕДРЕНИИ РЕЗУЛЬТАТОВ РАБОТЫ

УТВЕРЖДАЮ
Проректор по учебной работе
ФГБОУ ВО «ВГТУ»
_____ С.А. Яременко
«10» _____02_____ 2026 г.

А К Т
внедрения результатов кандидатской диссертации в учебный процесс ФГБОУ ВО «Воронежский государственный технический университет»

Тема диссертации: «Адаптивная система управления интеллектуальным агентом на основе глубокого обучения с подкреплением».

Автор: Кашко Василий Васильевич

Научный руководитель: Олейникова Светлана Александровна

Выполненной в ФГБОУ ВО «Воронежский государственный технический университет» на кафедре автоматизированных и вычислительных систем в рамках основного научного направления «Информатика и вычислительная техника»

В период с «1» сентября 2025 г. по н.в. внедрены в учебный процесс кафедры по группе научных специальностей 2.3 «Информационные технологии и телекоммуникации», научной специальности 2.3.1 «Системный анализ, управление и обработка информации, статистика», на основании решения кафедры АВС от «_09_» _сентября_ 2025 г., протокол № 1.

1. Вид результатов внедренных в учебный процесс: математическое и алгоритмическое обеспечение системы управления шагающим роботом, основанное на дискретных марковских процессах в контексте методов обучения с подкреплением, разработанное в ходе диссертационного исследования.

2. Область применения: лекционные и лабораторные занятия по дисциплине «Интеллектуальные системы» и «Системы искусственного интеллекта», выполнение курсовых проектов, выпускных квалификационных работ.

3. Форма внедрения: разработанные в диссертационном исследовании математическая модель системы управления шагающим роботом,

архитектура нейросетевого агента управления, основанного на обучении с подкреплением и комплекс алгоритмов его обучения были внедрены в образовательный процесс в виде моделей и алгоритмов с целью обучения студентов особенностям функционирования специализированных интеллектуальных систем управления на основе алгоритмов обучения с подкреплением.

4. Эффект от внедрения. Повышение качества образования: применение нового математического и алгоритмического обеспечения позволило улучшить результаты обучения интеллектуальных агентов по показателям стабильности, эффективности и качества функционирования.

Научный руководитель диссертанта

_____ Олейникова С.А.
(подпись, Ф.И.О.)

«_9_»__02__2026 г.

Начальник УМУ

_____ Скляров К.А.
(подпись, Ф.И.О.)

«_10_»__02__2026 г.

Диссертант

_____ Кашко В.В.
(подпись, Ф.И.О.)

«_9_»__02__2026 г.

Декан ФИТКБ

_____ Бредихин А.В.
(подпись, Ф.И.О.)

«_10_»__02__2026 г.

Заведующий кафедрой АВС

_____ Барабанов В.Ф.
(подпись, Ф.И.О.)

«_9_»__02__2026 г.

УТВЕРЖДАЮ
Генеральный директор
ООО «Девелоперс»
Черников В.Н.

«20» апреля 2026 г.

АКТ О ВНЕДРЕНИИ

Результаты научно-исследовательской работы Кашко Василия Васильевича, связанные с разработкой адаптивной системы управления интеллектуальным агентом на основе обучения с подкреплением, включая разработанные математические модели, алгоритмы и программную реализацию внедрены в деятельность компании ООО «Девелоперс» в виде алгоритмического и программного обеспечения для построения адаптивных систем управления интеллектуальными агентами, основанного на методах обучения с подкреплением.

Эффект от внедрения заключается в повышении точности принятия решений в изменяющихся условиях и сокращении времени на ручную настройку параметров алгоритмов управления за счёт автоматического обучения.

Генеральный директор Черников В.Н.

ПРИЛОЖЕНИЕ В
СВИДЕТЕЛЬСТВА О РЕГИСТРАЦИИ ПРОГРАММЫ ДЛЯ ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023665763

Интеллектуальная система поддержки принятия решений на базе аппарата искусственных нейронных сетей

Правообладатели: ***Гребенникова Наталия Ивановна (RU), Кашко Василий Васильевич (RU), Олейникова Светлана Александровна (RU)***

Авторы: ***Кашко Василий Васильевич (RU), Гребенникова Наталия Ивановна (RU), Олейникова Светлана Александровна (RU)***

Заявка № **2023664609**
Дата поступления **02 июля 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***20 июля 2023 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Ю.С. Зубов

ПРИЛОЖЕНИЕ С
ФОРМИРОВАНИЕ ВЕКТОРА СКОРОСТЕЙ СОЧЛЕНЕНИЙ

Для обеспечения демпфирования в процессе движения необходимо указать угловые скорости приводов, которые рассчитываются на основе разности углов:

$$\upsilon_{servo} = \frac{\theta(t) - \theta(t-1)}{\Delta t} \tag{C.1}$$

Основной проблемой данной формулы является наличие шумов и скачков из-за неточности сенсорных данных. Для ликвидации соответствующих эффектов производится фильтрация скоростей:

$$\upsilon_{filtered} = \alpha\upsilon_{servo}(t) + (1-\alpha)\upsilon_{servo}(t-1), \tag{C.2}$$

где α – коэффициент фильтрации, $\upsilon_{servo}(t)$ – скорость в момент времени t, $\upsilon_{servo}(t-1)$ – скорость в предыдущий момент времени.

Коэффициент фильтрации отвечает за качество перехода привода из старого положения в новое. При $\alpha = 0.1$ движение происходит плавно, но с дрожанием. Оптимальными значениями для соответствующего параметра считаются $0.2 - 0.3$. Если $\alpha = 0.5$, то переход осуществляется быстро, но с наличием шумов. К сожалению, даже после фильтрации возможны «выбросы». Для их ликвидации используется ограничение по максимальной скорости:

$$-\upsilon_{\max} \le \upsilon \le \upsilon_{\max} \tag{C.3}$$

Значение максимальной скорости $\upsilon_{\max}$ определяется практическим путём и обозначает не скорость привода «по паспорту», а «типичный» верхний предел:

$$\upsilon_{\max} = \beta(\upsilon_{typical}), \tag{C.4}$$

где $\beta \in [1.2,1.5]$, $\upsilon_{typical}$ – полученная эмпирически типичная максимальная скорость вращения.

Также, она может быть определена эвристически. Задаётся начальное значение и осуществляется анализ движения. Если привод колеблется при перемещении, то скорость $\upsilon_{\max}$ уменьшается. Если привод плохо реагирует, то значение увеличивают.

После «обрезки» значений, требуется произвести ликвидацию мелких шумов. Для этого вводится некоторое значение минимальной скорости $\upsilon_{\min}$, скорости ниже которой приравниваются к 0.

Для обеспечения демпфирования в процессе движения необходимо указать угловые скорости приводов, которые рассчитываются на основе разности углов:

$$\upsilon_{servo} = \frac{\theta(t) - \theta(t-1)}{\Delta t} \tag{C.5}$$

Основной проблемой данной формулы является наличие шумов и скачков из-за неточности сенсорных данных. Для ликвидации соответствующих эффектов производится фильтрация скоростей:

$$\upsilon_{filtered} = \alpha\upsilon_{servo}(t) + (1-\alpha)\upsilon_{servo}(t-1), \tag{C.6}$$

где α – коэффициент фильтрации, $\upsilon_{servo}(t)$ – скорость в момент времени t, $\upsilon_{servo}(t-1)$ – скорость в предыдущий момент времени.

Коэффициент фильтрации отвечает за качество перехода привода из старого положения в новое. При $\alpha = 0.1$ движение происходит плавно, но с дрожанием. Оптимальными значениями для соответствующего параметра считаются $0.2-0.3$. Если $\alpha = 0.5$, то переход осуществляется быстро, но с наличием шумов. К сожалению, даже после фильтрации возможны «выбросы». Для их ликвидации используется ограничение по максимальной скорости:

$$-\upsilon_{\max} \le \upsilon \le \upsilon_{\max} \qquad \text{(C.7)}$$

Значение максимальной скорости $\upsilon_{\max}$ определяется практическим путём и обозначает не скорость привода «по паспорту», а «типичный» верхний предел:

$$\upsilon_{\max} = \beta(\upsilon_{typical}), \qquad \text{(C.8)}$$

где $\beta \in [1.2,1.5]$, $\upsilon_{typical}$ – полученная эмпирически типичная максимальная скорость вращения.

Также, она может быть определена эвристически. Задаётся начальное значение и осуществляется анализ движения. Если привод колеблется при перемещении, то скорость $\upsilon_{\max}$ уменьшается. Если привод плохо реагирует, то значение увеличивают.

После «обрезки» значений, требуется произвести ликвидацию мелких шумов. Для этого вводится некоторое значение минимальной скорости $\upsilon_{\min}$, скорости ниже которой приравниваются к 0.

ПРИЛОЖЕНИЕ D
ПРИМЕНЕНИЕ SE(3) ГРУППЫ И ПОДХОДА «ПРОИЗВЕДЕНИЕ ЭКСПОНЕНТ» (PRODUCT OF EXPONENTIAL, POE) ДЛЯ ОПИСАНИЯ ПРЯМОЙ КИНЕМАТИКИ ШАГАЮЩЕГО РОБОТА.

Для описания робота в пространстве можно было бы использовать шесть координат, три из которых представляют собой позицию (x, y, z), а остальные – ориентацию в пространстве (например, углы Эйлера). Но такой подход неудобен для выполнения вычислений. При этом часто могут возникать сингулярности, сложность комбинирования движений и трудности обработки в матричной форме. По этой причине, в современной робототехнике используют группу преобразований. SE(3) или Special Euclidean Grope in 3D – это специальная евклидова группа в трёх измерениях, представляющая множество всех возможных жёстких преобразований в трёхмерном пространстве. Под жёсткими преобразованиями понимаются такие, которые сохраняют расстояния, углы и не деформируют объект движения. К ним относятся вращение, перенос или их комбинация. SE(3) называется группой, поскольку выполняются следующие свойства: замкнутость – композиция двух преобразований есть преобразование $T_1T_2 \in SE(3)$, наличие единичного элемента, что означает отсутствие преобразования $I = \begin{bmatrix} I_3 & 0 \\ 0 & 1 \end{bmatrix}$, наличие обратного преобразования T^{-1}, ассоциативность – $(T_1T_2)T_3 = T_1(T_2T_3)$.

SE(3) представляет собой пространство всех возможных поз робота. Любая поза включает позицию и ориентацию в пространстве. Элементом SE(3) группы является матрица, составленная из матрицы вращения R и вектора перемещения p:

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}, \tag{D.1}$$

где R – матрица вращения, p – вектор переноса.

Используя этот элемент можно описать любое положение твёрдого тела. Матрица преобразования представляет собой матрицу 4×4. Группа SE(3) обладает обратным элементом следующего вида:

$$T^{-1}=\begin{bmatrix} R^T & -R^T p \\ 0 & 1 \end{bmatrix}, \tag{D.2}$$

где T^{-1} – обратное преобразование, R^T – транспонированная матрица вращения.

Каждая конфигурация (поза) робота представляет собой элемент SE(3). Преобразование систем координат является совокупностью преобразований – цепочка произведений элементов SE(3).

Рассмотрим корпус робота. Он обладает ориентацией и положением в пространстве. Положение и ориентация образуют SE(3) элемент. Вектор переноса p имеет следующий вид:

$$p=\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}, \tag{D.3}$$

где p_x, p_y, p_z – координаты начала локальной системы координат тела в глобальной системе.

Матрица вращения R имеет следующий вид:

$$R=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{D.4}$$

Столбцы матрицы R представляют собой оси локальной системы координат, выраженные в глобальной системе.

Если задана локальная система координат робота F_B, с осями x_B, y_B и z_B, то каждый столбец представляет собой направление соответствующей оси тела в глобальной системе координат:

$$R = \begin{bmatrix} | & | & | \\ x_B & y_B & z_B \\ | & | & | \end{bmatrix} \tag{D.5}$$

Матрицы поворота на угол θ вокруг осей X, Y и Z имеют следующий вид:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{D.6}$$

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{D.7}$$

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{D.8}$$

Матрица R обладает следующими свойствами: ортогональность $R^T R = I$ и $\det(R) = 1$, и действует на точку следующим образом. Пусть p_B – точка в теле робота, относительно локальной системы координат F_B. Тогда её глобальные координаты определяются, как $p_W = Rp_B$, а полное преобразование имеет следующий вид:

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$$
$$p_W = Rp_B + p \tag{D.9}$$

В глобальной системе координат F_W, p представляет собой положение корпуса робота в пространстве, а R – наклон и поворот, что в совокупности характеризует конфигурацию (позу) механизма в конкретный момент времени.

Пространство se(3) представляет собой пространство мгновенных движений тела (twists), то есть скорости движения и является алгеброй Ли группы SE(3). Twist – это математический объект, описывающий мгновенное движение твёрдого тела в трёхмерном пространстве. Если SE(3) описывает расположение тела, то se(3) движение тела. Элемент se(3) называется twist вектором и выглядит следующим образом:

$$Twist = \begin{bmatrix} \omega \\ \upsilon \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ \upsilon_x \\ \upsilon_y \\ \upsilon_z \end{bmatrix}, \tag{D.10}$$

где $\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$ – угловая характеристика, $\upsilon = \begin{bmatrix} \upsilon_x \\ \upsilon_y \\ \upsilon_z \end{bmatrix}$ – линейная характеристика.

Элемент se(3) имеет матричную форму следующего вида:

$$[Twist] = \begin{bmatrix} [\omega] & \upsilon \\ 0 & 0 \end{bmatrix}, \tag{D.11}$$

где матрица $[\omega]$ – скошенная матрица (skew-symmetric).

Скошенная матрица $[\omega]$ кодирует операцию векторного произведения $[\omega]p = \omega \times p$, и имеет следующий вид:

$$[\omega] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{D.12}$$

В робототехнике выделяются два типа twist векторов: twist S и twist V. Они имеют одинаковую структуру, оба являются элементами twist se(3), но обладают разным физическим смыслом. Twist V – это мгновенная скорость твёрдого тела, представленная в виде вектора (D.10), у которого ω – угловая скорость, а υ – линейная скорость. Это реальный физический twist, зависящий от времени. Twist V подразделяется на body twist V_B, рассматриваемый в системе координат тела F_B и spatial velocity twist V_S – скорость в пространственной системе координат.

Twist S – это винтовая ось сустава (screw axis). Она также обладает видом (D.10), но её смысл заключается в геометрии движения сустава. Twist S не зависит от скорости и состоит из направления оси вращения ω (единичный вектор оси) и направления поступательного движения υ. Элементы twist S и twist V связаны между собой. Если сустав вращается со скоростью $\dot{\theta}$, то:

$$V = S\dot{\theta} \tag{D.13}$$

Вектор S – единичный вектор направления движения в пространстве twists, V – это реальный вектор скорости.

Переход $se(3) \to SE(3)$ осуществляется через матричную экспоненту $T = e^{[Twist]\theta}$ и обозначает положение тела в пространстве при движении тела с $Twist$, где $[V]$ – матричная форма элемента se(3), а θ – величина движения.

В общем случае, матричная форма элемента se(3) имеет следующий вид:

$$[Twist]=\begin{bmatrix} 0 & -\omega_z & \omega_y & \upsilon_x \\ \omega_z & 0 & -\omega_x & \upsilon_y \\ -\omega_y & \omega_x & 0 & \upsilon_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{D.14}$$

Переход $Twist$ из матричной формы (D.14) в векторную форму (D.10) осуществляется путём непосредственного извлечения компонент. Рассмотрим скошенную матрицу (D.12), формирующую матричную форму (D.14). Компоненты ω заполняются следующим образом:

$$\begin{aligned} \omega_x &= [\omega]_{2,1} \\ \omega_y &= [\omega]_{0,2} \\ \omega_z &= [\omega]_{1,0} \end{aligned}, \tag{D.15}$$

где $[\omega]_{i,j}$ – элемент, расположенный на пересечении i-ой строки с j-ым столбцом матрицы $[\omega]$.

Формирование скошенной матрицы (D.12) осуществляется по обратному принципу – установкой компонент ω вектора $Twist$ в соответствующие координаты матрицы, как представлено в (D.15) и добавлением отрицательных компонент симметрично положительным в положения:

$$\begin{aligned} -\omega_x &= [\omega]_{1,2} \\ -\omega_y &= [\omega]_{2,0} \\ -\omega_z &= [\omega]_{0,1} \end{aligned} \tag{D.16}$$

Переход $SE(3) \to se(3)$ осуществляется через матричный логарифм $\log(T)$ и производит извлечение мгновенного движения из положения. В итоге,

SE(3) описывает расположение тела в пространстве, а se(3) – движение тела. se(3) является касательным пространством к SE(3). *Twist* обозначает мгновенную скорость, а $e^{[V]\theta}$ – интегрирование скорости, что эквивалентно получению позы. Вектор *Twist* используется для хранения скоростей, а матричная форма предназначена для вычисления экспоненты, которая функционирует только с матрицами. Исходя из этого, возникает потребность в операции преобразования $V \to [V]$.

При решении задач кинематики, часто можно встретить использование классического подхода, с применением параметров Денавита-Хартенберга (DH). Его основными недостатками являются привязанность к последовательной структуре, неудобство в применении для манипуляторов со сложной геометрией и плохая масштабируемость. В контексте данной работы, главным критерием построения системы управления является универсальность, которую сложно обеспечить вышеупомянутым методом.

В настоящее время существует современный способ описания кинематики робототехнических систем, который не зависит от количества суставов и конфигурации робота, идеально подходит для n-ногих систем и легко расширяется на динамику. Это подход «Произведение экспонент» (Product of Exponentials, PoE). Он основывается на теореме Часла, которая утверждает, что любое движение твёрдого тела в пространстве можно представить как совокупность вращения вокруг некоторой оси и поступательного движения вдоль этой оси, что является винтовым движением (screw motion). Основной идеей PoE является представление любой последовательности суставов в виде последовательности винтовых движений. Движение твёрдого тела описывается дифференциальным уравнением $\dot{T} = [S]T$, решением которого является $T = e^{[S]\theta}$. Экспонента представляет собой интеграл мгновенного движения – позу робота. Каждый сустав создаёт элемент SE(3) $e^{[S_i]\theta_i}$, а вся кинематика конечности представляется в виде произведения соответствующих экспонент:

$$T(q) = e^{[S_1]\theta_1} e^{[S_2]\theta_2} \ldots e^{[S_n]\theta_n} M\,, \tag{D.17}$$

где S_i – винтовая ось i-ого сустава (Twist S), θ_i – угол i-ого сустава, M – поза конечности при нулевых углах (положение стопы когда все углы равны 0).

Винтовая ось представляет собой геометрическую линию в пространстве, вокруг которой происходит движение тела и характеризуется направлением, точкой, через которую проходит ось и шагом винта (pitch). Винтовая ось сустава, 6-и мерный вектор S имеет следующий вид:

$$S = \begin{bmatrix} \omega \\ \upsilon \end{bmatrix} = \begin{bmatrix} \omega \\ -\omega \times q \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ \upsilon_x \\ \upsilon_y \\ \upsilon_z \end{bmatrix}, \tag{D.18}$$

где ω – направление оси вращения (единичный вектор оси), $\upsilon = -\omega \times q = \begin{bmatrix} \upsilon_x \\ \upsilon_y \\ \upsilon_z \end{bmatrix}$ – направление поступательного движения (линейная часть винта), где q – любая точка на оси вращения сустава в локальной системе F_B в позе M (определяет, как вращение вызывает линейное движение).

Точка q на практике выбирается в центре сустава, там, где соединяются два звена между собой, и проходит ось вращения. Единичный вектор оси вращения ω – это вектор, показывающий направление оси, вокруг которой вращается сустав:

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{D.20}$$

Данный вектор удовлетворяет условию $\|\omega\| = 1$, то есть $\sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2 = 1}$. На практике, при вращении вокруг осей X, Y и Z, вектор ω имеет следующий вид:

$$\omega_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \omega_y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \omega_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{D.21}$$

В случае если ось наклонена и направлена вдоль некоторого вектора d, то $\omega = \frac{d}{\|d\|}$. Когда тело вращается вокруг оси, то точка на оси не движется, а точки вокруг оси имеют скорость, которую и характеризует υ.

Матричная экспонента $e^{[S]\theta}$ винтовой оси превращает винтовое вращение в конечное перемещение. Задача ставится следующим образом. Если задана винтовая ось сустава S и перемещение сустава θ, то требуется найти матрицу позы $T(\theta)$, которая имеет вид экспоненты $T(\theta) = e^{[S]\theta}$, где $[S]$ – матричная форма вектора винтовой оси S. Для взятия экспоненты требуется представить $S \in R^6$ в виде матрицы $[S]$ 4×4. Она имеет вид, соответствующий общей форме *Twist* (D.14):

$$[S] = \begin{bmatrix} [\omega] & \upsilon \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\omega_z & \omega_y & \upsilon_x \\ \omega_z & 0 & -\omega_x & \upsilon_y \\ -\omega_y & \omega_x & 0 & \upsilon_z \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{D.22}$$

где $[\omega]$ – скошенная матрица, $\upsilon = -\omega \times q$ – направление поступательного движения (линейная часть винта). Элемент $[\omega]$ представляет собой

скошенную матрицу (skew-symmetric), аналогичную (D.12), которая кодирует операцию векторного произведения $[\omega]p = \omega \times p$. Преобразование вектора оси S в матричную форму необходимо, поскольку экспонента матрицы определяется только для квадратных матриц в виде следующего ряда:

$$e^{A} = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \ldots \tag{D.23}$$

Экспоненту $e^{[S]\theta}$ можно представить в виде ряда:

$$e^{[S]\theta} = I + [S]\theta + \frac{([S]\theta)^2}{2!} + \frac{([S]\theta)^3}{3!} + \ldots \tag{D.24}$$

При подсчёте степеней, получается, что:

$$e^{[S]\theta} = \begin{bmatrix} e^{[\omega]\theta} & p \\ 0 & 1 \end{bmatrix}, \tag{D.25}$$

где $R = e^{[\omega]\theta}$ – матрица вращения, p – вектор переноса.

Элемент $e^{[S]\theta}$ для вращательного сустава имеет следующий вид:

$$e^{[S]\theta} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} e^{[\omega]\theta} & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} e^{[\omega]\theta} & G(\theta)\upsilon \\ 0 & 1 \end{bmatrix}, \tag{D.26}$$

где $e^{[\omega]\theta} = I + \sin\theta[\omega] + (1 - \cos\theta)[\omega]^2$ – формула Родрига, формирующая матрицу вращения, $p = G(\theta)\upsilon = \int_0^{\theta} e^{[\omega]s}\upsilon ds$ – (интеграл вращения), который превращает линейную часть винта в реальное смещение (вектор переноса), s – параметр

интегрирования, обозначающий значение текущего угла поворота в процессе вращения от 0 до θ.

Матрица $G(\theta)$ обладает размерностью 3×3, зависит от оси вращения и угла поворота и рассчитывается по следующей формуле:

$$G(\theta) = I\theta + (1-\cos\theta)[\omega] + (\theta-\sin\theta)[\omega]^2, \tag{D.27}$$

С геометрической точки зрения $G(\theta)$ является итоговым смещением точки при винтовом движении. Если вращение осуществляется вокруг оси и ось не проходит через начало координат, то тело описывает дугу, которую вычисляет $G(\theta)$.

Матричный логарифм имеет вид:

$$\log(T) = \begin{bmatrix} [\omega]\theta & \upsilon\theta \\ 0 & 0 \end{bmatrix} \tag{D.28}$$

Для вычисления логарифма необходимо определить угол вращения θ, ось вращения $[\omega]$и линейную часть υ через матрицу вращения Rи вектор переноса p элемента T:

$$\theta = \cos^{-1}\left(\frac{trace(R)-1}{2}\right) \tag{D.29}$$

$$[\omega] = \frac{1}{2\sin\theta}\left(R - R^T\right) \tag{D.30}$$

$$\upsilon = G^{-1}(\theta)p \tag{D.31}$$

$$G^{-1} = \frac{1}{\theta}I - \frac{1}{2}[\omega] + \left(\frac{1}{\theta} - \frac{1}{2}ctg\frac{\theta}{2}\right)[\omega]^2 \tag{D.32}$$

Операция *trace* представляет собой сумму элементов, расположенных на главной диагонали матрицы:

$$trace(A) = \sum_{i=1}^{n} A_{ii} \tag{D.33}$$

В формуле (D.29) $trace(R) = R_{00} + R_{11} + R_{22}$. Twist ошибки формируется из матричной формы согласно правилу (D.15).

ПРИЛОЖЕНИЕ Е
ДЕТАЛИ ОПРЕДЕЛЕНИЯ КООРДИНАТ И СКОРОСТИ ЦЕНТРА, КООРДИНАТ ТОЧКИ НУЛЕВОГО МОМЕНТА И ТОЧКИ ЗАХВАТА ПРЕДСТАВЛЕНЫ

Центр масс (Center of Mass, CoM) – это точка, в которой сосредоточена вся масса тела при описании его поступательного движения. CoM определяется по следующей формуле:

$$r_{CoM} = \frac{1}{M}\sum_{i=1}^{n} m_i r_i \,, \tag{E.1}$$

где m_i – масса i-ой части, r_i – координаты i-ой части, $M = \sum m_i$ – полная масса системы.

Робот разбивается на твёрдые звенья: корпус, бедро, голень, стопа. Обычно, координаты центра масс каждого звена робота, совместно с массой m_i, задаются в локальной системе координат звена:

$$p_{CoM_i}^{local} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{E.2}$$

У каждого звена вводится собственная система координат – локальная система координат звена, которая жёстко закреплена на звене, движется вместе со звеном. Обычно она располагается либо в суставе, либо в геометрическом центре звена.

Для определения координат центра масс звена в мировой системе F_W, необходимо выполнить преобразование координат через матрицу позы звена $T_i = \begin{bmatrix} R_i & p_i \\ 0 & 1 \end{bmatrix}$ следующим образом:

$$p_{CoM_i}^{world} = R_i p_{CoM_i}^{local} + p_i \tag{E.3}$$

Или в однородных координатах:

$$p_{CoM_i}^{world} = T_i \begin{bmatrix} p_{CoM_i}^{local} \\ 1 \end{bmatrix} \tag{E.4}$$

После этого становится возможным определение координат точки CoM всего робота:

$$p_{CoM}^{world} = \frac{\sum_{i=1}^{L} m_i p_{CoM_i}^{world}}{\sum_{i=1}^{L} m_i}, \tag{E.5}$$

где L – число звеньев робота.

Определение ускорения центра масс осуществляется на основе датчика гироскопа-акселерометра (IMU), где вводится упрощение без учёта гравитации, следующего вида: $\ddot{p}_{CoM} \approx \ddot{p}_{IMU}$. Оно уместно, если гироскоп-акселерометр расположен близко к фактическому центру масс (расстояние меньше 10-20 сантиметров). При построении простого баланс контроллера достаточно использовать горизонтальные ускорения.

Точка нулевого момента (ZMP) – это точка, принадлежащая опорной плоскости, в которой отсутствуют компоненты суммарного момента всех действующих на систему сил в плоскости поверхности. При этом делается предположение, что трение препятствует скольжению и поверхность контакта – плоская поверхность. Пусть задана плоскость Π – опорная поверхность и проведена нормаль n. Тогда точка нулевого момента $r_{ZMP} \in \Pi$ определяется следующим образом:

$$n \cdot (\sum_i (r_i - r_{ZMP}) \times F_i) = 0 \tag{E.6}$$

где, F_i – сила i, действующая на робота, r_i – точки приложения силы i.

В процессе выбора системы координат, опорная плоскость принимается, как $z = 0$, а нормаль направлена вверх, так, что точка нулевого момента имеет вид: $r_{ZMP} = (x_{ZMP}, y_{ZMP}, 0)$. Отсюда получается, что $M_{ZMP}^{xy} = 0$ – значения моментов сил, вокруг осей X и Y равны нулю. Для шагающих роботов часто на практике используют модель линейного инвертированного маятника (Linear Inverted Pendulum Model, LIPM). Согласно данной модели предполагается, что центр масс расположен на постоянной высоте и угловой момент вокруг него примерно равен нулю. В результате координаты точки ZMP определяются согласно следующей формуле:

$$\begin{aligned} x_{ZMP} &= x_{CoM} - \frac{z_{CoM}}{g} \ddot{x}_{CoM} \\ y_{ZMP} &= y_{CoM} - \frac{z_{CoM}}{g} \ddot{y}_{CoM} \end{aligned}, \tag{E.7}$$

где x_{CoM}, y_{CoM} – координаты центра масс, $\ddot{x}_{CoM}, \ddot{y}_{CoM}$ – ускорения центра масс, z_{CoM} – высота центра масс, g – гравитация.

Точка ZMP существует до тех пор, пока условие $\sum_k F_{z,k} \le 0$ является не выполнимым. В противном случае, робот находится в состоянии потери контакта с опорой (полёт или опора «вывернута»). В таком случае, для определения баланса следует использовать точку захвата (Capture Point).

Тока захвата (Capture Point, CP) – это точка, расположенная на опоре, в которую необходимо установить конечность робота после некоторого возмущения, чтобы выполнить остановку его движения или осуществить переход в устойчивую ходьбу. Отсутствие такой точки

гарантирует падение робота на следующем шаге. Теория CP точки базируется на модели LIPM со следующими предположениями:

1. Центр тяжести перемещается в горизонтальной плоскости;
2. Центр тяжести имеет постоянную высоту;
3. Вращательная динамика корпуса компенсирована или полностью отсутствует.

Уравнение движения имеет следующий вид:

$$\ddot{x} = \omega^2 (x - p)$$
$$\omega = \sqrt{\frac{g}{h}} \tag{E.8}$$

где, $\ddot{x}$ – ускорение центра тяжести (горизонтальное), x – положение центра масс (горизонтальное), p – точка опоры (конечность / ZMP), $x - p$ – плечо силы тяжести, ω^2 – «жёсткость» гравитационного маятника, h – высота CoM, g – ускорение свободного падения.

Исходя из модели, точка захвата определяется следующим образом:

$$x_{CP} = x + \frac{\dot{x}}{\omega}, \tag{E.9}$$

где x – положение центра масс (горизонтальное), $\dot{x}$ – скорость центра масс (горизонтальная).

В контексте определения CP точки вводится состояние системы $s = (x, \dot{x})$, которое предварительно определяется на основе кинематики и датчика IMU. Далее производится расчёт согласно формуле (B.9). Точка CP рассматривается в рамках области достижимых шагов – множество возможных позиций робота на следующем шаге, обеспечивающих реализацию шага за доступный интервал времени, ограниченное длиной и

шириной шага и отклонениями углов суставов. Определяется по следующему алгоритму:

1. Определяется максимальная зона досягаемости конечности (простая фигура, например овал) на базе кинематики;
2. Учитывая время реакции, производится уменьшение области;
3. Производится исключение точек требующих резких движений, усилий, нарушающих баланс;
4. Производится исключение априорно непригодных для выполнения шага областей (препятствия, ямы).

ПРИЛОЖЕНИЕ F
ОПРЕДЕЛЕНИЕ КООРДИНАТНЫХ СИСТЕМ SE(3) ЭЛЕМЕНТОВ И ПРЯМОЙ КИНЕМАТИКИ ШАГАЮЩЕГО РОБОТА С ДВУМЯ КОНЕЧНОСТЯМИ

На рисунке F.1 изображены требуемые системы координат, на примере двуногого шагающего робота, используемого для проведения экспериментов.

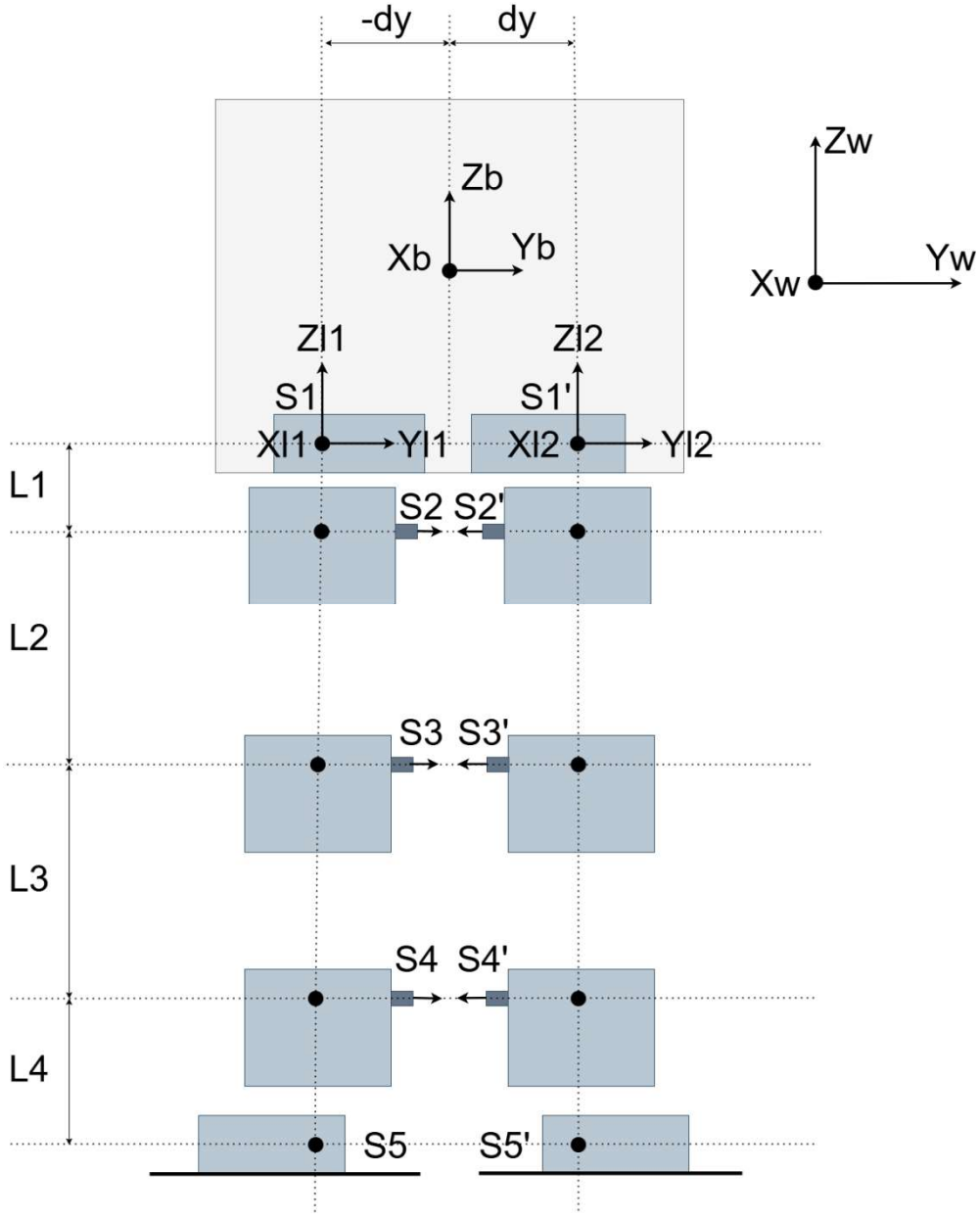


Рисунок F.1 – Схема робота

Выбор двуногого робота обусловлен его нестабильностью, по сравнению с другими конфигурациями, обладающими большим числом конечностей, обеспечивающих наибольшую устойчивость конструкции. Робот представлен во фронтальной плоскости. Мировая система координат F_W, система координат корпуса F_B и системы координат конечностей F_{L_i} имеют одинаковую ориентацию в пространстве, где ось X – движение вперёд (большой палец), ось Z – направлена вверх (средний палец), а ось Y выбирается таким образом, чтобы системы были правыми, то есть, направлена вправо (указательный палец). На рисунке F.1 изображены оси винтового вращения S_i – для правой конечности, $S_i^{'}$ – для левой конечности. Выбор одинаковых по направлениям систем координат конечностей обусловлен требованием модульности, которое позволяет произвести описание кинематики для одной конечности и использовать её для других, с учётом их расположения. Произведём описание кинематики педипуляторов на основе элементов SE(3).

Винтовые оси S_1 и $S_1^{'}$ осуществляют вращение вдоль оси X, локальных систем координат конечности. Следовательно, $\omega_{S_1} = \omega_{S_1^{'}} = [1,0,0]$. Возьмём начала координат, в качестве точек на соответствующих осях: $p_{S_1} = p_{S_1^{'}} = [0,0,0]$. Определим направления поступательного движения υ_{S_1} и $\upsilon_{S_1^{'}}$, исходя из того, что $\upsilon = -\omega \times q$. В результате, $\upsilon_{S_1} = \upsilon_{S_1^{'}} = [0,0,0]$. Следовательно, оси S_1 и $S_1^{'}$ имеют следующий вид: $S_1 = S_1^{'} = [1,0,0,0,0,0]$.

Определим S_2 и $S_2^{'}$. Вращение осуществляется вдоль оси Y. Поскольку S_2 имеет тоже направление, что и Y, то $\omega_{S_2} = [0,1,0]$. Ось вращения $S_2^{'}$ направлена в противоположную сторону, в результате чего $\omega_{S_2^{'}} = [0,-1,0]$. И S_2 и $S_2^{'}$ являются смещёнными по оси Z на величину $-L_1$. Следовательно, точки на соответствующих осях будут равны $p_{S_1} = p_{S_1^{'}} = [0,0,-L_1]$. В результате направления поступательного движения υ_{S2} и $\upsilon_{S_2^{'}}$ будут равны $\upsilon_{S2} = [-L_1,0,0]$ и

$\upsilon_{S_2'} = [L_1,0,0]$ соответственно. Получим, что $S_2 = [0,1,0,-L_1,0,0]$ и $S_2' = [0,-1,0,L_1,0,0]$. Оси S_3, S_3', S_4 и S_4' осуществляют вращение по оси Y и рассчитываются по аналогии с S_2 и S_2', учитывая, что S_3 и S_3' смещены относительно центра системы координат педипулятора на расстояние вдоль оси Z на величину $-(L_1+L_2)$, S_4 и S_4' на $-(L_1+L_2+L_3)$. В результате, $S_3 = [0,1,0,-(L_1+L_2),0,0]$, $S_3' = [0,-1,0,(L_1+L_2),0,0]$, $S_4 = [0,1,0,-(L_1+L_2+L_3),0,0]$ и $S_4' = [0,1,0,(L_1+L_2+L_3),0,0]$. Определение S_5 и S_5' эквивалентно S_1 и S_1'. С учётом смещения на величину $-(L_1+L_2+L_3+L_4)$ по оси Z, $S_5 = [0,1,0,-(L_1+L_2+L_3+L_4),0,0]$ и $S_5' = [0,1,0,(L_1+L_2+L_3+L_4),0,0]$.

Определим матрицу M нулевой позы конечности. Для этого необходимо рассчитать положение p каждой стопы при нулевых значениях углов сочленений. В результате получим, что $p_{Leg_1} = p_{Leg_2} = \begin{bmatrix} 0 \\ 0 \\ -(L_1+L_2+L_3+L_4) \end{bmatrix}$.

Исходя из того, что $M = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$, а стопы ориентированы также как и соответствующие им системы координат, то есть $R = I$, то получим:

$$M_{Leg_1} = M_{Leg_2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -(L_1+L_2+L_3+L_4) \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Определив все винтовые оси и стартовую матрицу для каждой конечности, становится возможным определение прямой кинематики относительно системы координат педипулятора. Для каждого из педипуляторов она имеет следующий вид, согласно подходу PoE: $T_{L_1}(\theta) = e^{[S_1]\theta_1} e^{[S_2]\theta_2} e^{[S_3]\theta_3} e^{[S_4]\theta_4} e^{[S_5]\theta_5} M_{L_1}$ и $T_{L_2}(\theta) = e^{[S_1']\theta_6} e^{[S_2']\theta_7} e^{[S_3']\theta_8} e^{[S_4']\theta_9} e^{[S_5']\theta_{10}} M_{L_2}$ соответственно.

Далее требуется составить преобразование T_{BL_i}, определяющее координаты конечности относительно системы координат робота F_B.

Поскольку система координат каждого педипулятора имеет ту же ориентацию, что и F_B, то $R = I$. Системы координат конечностей смещены относительно центра системы F_B на величину dx по оси X, на величину $-dz$ по оси Z. По оси Y правая конечность смещена на величину $-dy$, а правая на dy. В результате получим: $T_{BL_1} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & -dy \\ 0 & 0 & 1 & -dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$ и $T_{BL_2} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & -dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

Определение $T_{WB} = \begin{bmatrix} R_{WB} & p_{WB} \\ 0 & 1 \end{bmatrix}$ рассматривается относительно положения робота в «мире» и не зависит от его геометрии. Для получения матрицы вращения R_{WB} используются показания датчика гироскопа-акселерометра. Гироскоп формирует вектор угловых скоростей (рад/с) $g = (g_x, g_y, g_z)$, а акселерометр вектор ускорений $a = (a_x, a_y, a_z)$, причём в покое, вектор ускорений примерно соответствует вектору гравитации. Для представления ориентации используется кватернион $q = (\omega, x, y, z)$ или $q = \omega + xi + yj + zk$, где ω – угол вращения, (x, y, z) – ось вращения. Он формируется следующим образом. Предварительно производится нормализация вектора ускорений акселерометра $a = \frac{a}{\|a\|}$ и далее вычисляются углы отклонения. Угол крена (наклон вбок) $roll$ и угол тангажа (наклон вперёд) $pitch$, определяются следующим образом:

$$roll = \varphi = \arctan 2(a_y, a_z) \tag{F.1}$$

$$pitch = \theta = \arctan 2\left(-a_x, \sqrt{a_y^2 + a_z^2}\right), \tag{F.2}$$

где

$$\arctan 2 = \begin{cases} \arctan\left(\frac{y}{x}\right), & x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi, & x < 0, y \geq 0 \\ \arctan\left(\frac{y}{x}\right) - \pi, & x < 0, y < 0 \\ +\frac{\pi}{2}, & x = 0, y > 0 \\ -\frac{\pi}{2}, & x = 0, y < 0 \\ \textit{неопределено} & x = 0, y = 0 \end{cases} \quad \text{(F.3)}$$

Угол рыскания *yaw* имеет сильный «дрейф». Поэтому для точного определения используется датчик магнитометр, который в рамках исследования не предусмотрен в конструкции робота. В результате $yaw = 0$. Исходя из полученных величин углов, формируется кватернион положения:

$$\begin{aligned} \omega &= \cos\left(\frac{\varphi}{2}\right)\cos\left(\frac{\theta}{2}\right) \\ x &= \sin\left(\frac{\varphi}{2}\right)\cos\left(\frac{\theta}{2}\right) \\ y &= \cos\left(\frac{\varphi}{2}\right)\sin\left(\frac{\theta}{2}\right) \\ z &= -\sin\left(\frac{\varphi}{2}\right)\sin\left(\frac{\theta}{2}\right) \end{aligned} \quad \text{(F.4)}$$

Из полученного кватерниона формируется матрица вращения R следующим образом:

$$R = \begin{bmatrix} 1-2\left(y^2+z^2\right) & 2\left(xy-\omega z\right) & 2\left(xz+\omega y\right) \\ 2\left(xy+\omega z\right) & 1-2\left(x^2+z^2\right) & 2\left(yz-\omega x\right) \\ 2\left(xz-\omega y\right) & 2\left(yz-\omega x\right) & 1-2\left(x^2+y^2\right) \end{bmatrix} \quad \text{(F.5)}$$

Кватернион изменяется со временем. Величина изменения определяется также на основе показаний гироскопа. Для этого угловая

скорость представляется в виде кватерниона вида $\Omega = (0, g_x, g_y, g_z)$, а приращение кватерниона определяется следующей формулой:

$$\dot{q} = \frac{1}{2} q \otimes \Omega = \frac{1}{2} \begin{bmatrix} -\omega & -x & -y & -z \\ x & \omega & -z & y \\ y & z & \omega & -x \\ z & -y & x & \omega \end{bmatrix} \cdot \begin{bmatrix} 0 \\ g_x \\ g_y \\ g_z \end{bmatrix} \quad \text{(F.6)}$$

В результате новый кватернион равен:

$$q = q + \dot{q} dt \quad \text{(F.7)}$$

Выполнив обязательную нормализацию, получим результирующий кватернион:

$$q = \frac{q}{\|q\|} \quad \text{(F.8)}$$

К сожалению, гироскоп обладает «дрейфом» по причине внешних воздействий, что пагубно влияет на качество показаний. Это критично для расчёта приращения и коррекции кватерниона положения. Для его корректировки используются показания акселерометра. Предварительно требуется стартовая калибровка датчика, в результате которой получается медленно изменяющаяся ошибка (смещение) гироскопа $bias\_gyro$, представляющая собой отклонение показаний сенсора при нахождении устройства в состоянии статического покоя. Измеренные показания имеют вид:

$$g_{mean} = g_{true} + bias\_gyro + noise, \quad \text{(F.9)}$$

где g_{mean} – измеренные показания гироскопа, g_{true} – истинные показания гироскопа, $bias\_gyro$ – ошибка гироскопа, $noise$ – шум.

В результате, текущие показания гироскопа имеют вид:

$$g = g_{mean} - bias\_gyro \tag{F.10}$$

Смещение показаний происходит, например, по причине температурного дрейфа, напряжения питания и медленно изменяется. Для получения смещения, на практике, в начале (инициализация), применяется усреднение показаний по N считываниям:

$$bias\_gyro = \frac{1}{N}\sum_{i=1}^{N} g_i \tag{F.11}$$

Обычно N соответствует 500 – 2000 измерений (1-2 секунды). Данная операция производит начальное значение смещения. Далее, в реальном времени, показания корректируются на основании показаний акселерометра при помощи фильтра Махони (Mahony). Идея подхода заключается в том, что если показания акселерометра демонстрируют отсутствие вращения, а гироскопа наоборот, свидетельствуют о вращении, то это означает наличие смещения. В результате, ошибка ориентации имеет следующий вид:

$$e = a \times g(q), \tag{F.12}$$

где e – ошибка ориентации, демонстрирующая направление поворота, a – вектор ускорений акселерометра (реальный «низ»), $g(q)$ – получаемый из кватерниона положения q вектор гравитации (текущая оценка «низа»). Вектор гравитации $g(q)$ определяется следующим образом:

$$\begin{aligned} g_x &= 2(xz - \omega y) \\ g_y &= 2(yz + \omega x) \\ g_z &= 1 - 2(x^2 + y^2) \end{aligned} \tag{F.13}$$

Показания акселерометра показывают реальное направление гравитации. Вектор $g(q)$ – предполагаемое направление. Если появляется ошибка, следовательно, ориентация «уплыла» и требуется выполнить корректировку текущих показаний гироскопа при помощи ПИ-регулятора:

$$g = g + K_p e + K_i \int e dt, \tag{F.14}$$

откуда:

$$bias\_gyro = bias\_gyro + K_i \cdot e \cdot dt, \tag{F.15}$$

$$g = g - bias\_gyro + K_p \cdot e, \tag{F.16}$$

где K_p – пропорциональный коэффициент в диапазоне от 1 до 5 (быстро исправляет наклон), K_i – коэффициент интегрирования в диапазоне от 0,01 до 0,1 (убирает ошибку гироскопа).

В результате общий алгоритм получения кватерниона положения, необходимого для формирования матрицы вращения R имеет следующий вид:

Шаг 0: Определить стартовый $bias\_gyro$*, как усреднённое значение по* N *запускам и стартовый кватернион* $q = (1,0,0,0)$*;*

Начало цикла:

Шаг 1: Считывание показаний гироскопа и акселерометра;

Шаг 2: Нормализация вектора ускорений акселерометра;

Шаг 3: Устранение искажения показаний гироскопа $bias\_gyro$*;*

Шаг 4: Предсказание вектора гравитации $g(q)$*;*

Шаг 5: Вычисление ошибки гироскопа;

Шаг 6: Вычисление коррекции кватерниона положения и $bias\_gyro$*;*

Шаг 7: Коррекция кватерниона положения;

Шаг 8: Нормализация кватерниона положения;

Шаг 9: Определение матрицы R*;*

Конец цикла

Текущее положение робота определяется на основе координат точки центра тяжести. На этапе инициализации, вектор $p=[0,0,h]$, где h – высота робота. Благодаря тому, что измерения производятся за дискретный промежуток времени dt и используется датчик гироскопа-акселерометра, на основе которого возможно определение скорости точки центра масс, то вектор p определяется следующим образом:

$$p(t) = p(t_0) + \int_{t_0}^{t} \upsilon(\tau) d\tau , \qquad \text{(F.17)}$$

где $\upsilon(\tau)$ – вектор линейной скорости в момент τ, $p(t_0)$ – начальное положение $[0,0,h]$.

На практике, изменение координат вектора p за дискретное время Δt производится в следующем виде:

$$p_{k+1} = p_k + \upsilon_k \cdot \Delta t , \qquad \text{(F.18)}$$

где Δt – шаг дискретизации.

В результате кинематика T_{WE}^{i} для педипуляторов имеет следующий вид:

$$T_{WE}^{1} = T_{WB} \cdot T_{BL_1} \cdot T_{L_1}$$
$$T_{WE}^{2} = T_{WB} \cdot T_{BL_2} \cdot T_{L_2} \quad \text{(F.19)}$$

ПРИЛОЖЕНИЕ G
ФРАГМЕНТ ПРОГРАММНОЙ РЕАЛИЗАЦИИ АДАПТИВНОЙ СИСТЕМЫ УПРАВЛЕНИЯ ШАГАЮЩИМ РОБОТОМ

Файл SwitchNeurons.py

```
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np

class CustomDropoutMLP(nn.Module):
    def __init__(self, input_size, hidden_sizes, output_size, dropout_prob=0.5):
        super(CustomDropoutMLP, self).__init__()
        self.input_size = input_size
        self.hidden_sizes = hidden_sizes
        self.output_size = output_size
        self.dropout_prob = dropout_prob

        # Создаём слои MLP
        layers = []
        prev_size = input_size

        for hidden_size in hidden_sizes:
            layers.append(nn.Linear(prev_size, hidden_size))
            layers.append(nn.ReLU())
            prev_size = hidden_size

        layers.append(nn.Linear(prev_size, output_size))
        self.network = nn.Sequential(*layers)

    def forward(self, x, custom_mask=None):
        # Применяем стандартную dropout маску, если пользовательская не
задана
        if custom_mask is None:
            return self.network(x)

        # Сохраняем исходное состояние dropout слоёв
        original_dropout_states = []
        for layer in self.network:
```

```
            if isinstance(layer, nn.Dropout):
                original_dropout_states.append(layer.training)
                layer.training = False  # Отключаем стандартный dropout

        output = x
        layer_idx = 0

        for i, layer in enumerate(self.network):
            output = layer(output)

            # Применяем пользовательскую маску после ReLU (перед следующим
линейным слоем)
            if isinstance(layer, nn.ReLU) and layer_idx < len(custom_mask):
                if layer_idx < len(custom_mask) and custom_mask[layer_idx] is not
None:
                    mask = custom_mask[layer_idx].to(output.device)
                    output = output * mask
                layer_idx += 1

        # Восстанавливаем исходное состояние dropout слоёв
        dropout_idx = 0
        for layer in self.network:
            if isinstance(layer, nn.Dropout):
                layer.training = original_dropout_states[dropout_idx]
                dropout_idx += 1

        return output

    def generate_custom_mask(self, batch_size):
        """Генерирует пользовательскую маску dropout для всех скрытых
слоёв"""
        masks = []
        for hidden_size in self.hidden_sizes:
            # Берём Bernoulli распределение с вероятностью 1 - dropout_prob (1 =
оставить, 0 = выключить)
            mask = torch.bernoulli(torch.ones(batch_size, hidden_size) * (1 -
self.dropout_prob))
            masks.append(mask)
        return masks

# Пример использования
if __name__ == "__main__":
    # Параметры сети
    input_size = 784   # Например, для MNIST
```

```
    hidden_sizes = [128, 64]  # Два скрытых слоя
    output_size = 10     # 10 классов
    dropout_prob = 0.3
    batch_size = 32

    # Создаём модель
    model = CustomDropoutMLP(input_size, hidden_sizes, output_size,
dropout_prob)


    # Генерируем случайный вход
    x = torch.randn(batch_size, input_size)

    # Вариант 1: стандартный forward pass (без пользовательской маски)
    print("Стандартный forward pass:")
    output1 = model(x)
    print(f"Выход: {output1.shape}")

    # Вариант 2: с пользовательской маской
    print("\nForward pass с пользовательской маской:")
    custom_mask = model.generate_custom_mask(batch_size)
    output2 = model(x, custom_mask)
    print(f"Выход: {output2.shape}")
    print(f"Количество активных нейронов в первом скрытом слое:
{custom_mask[0].sum(dim=1).mean():.2f}")

    # Вариант 3: с полностью выключенными нейронами в первом слое
    print("\nForward pass с полностью выключенным первым скрытым слоем:")
    mask_all_zero = [torch.zeros(batch_size, hidden_sizes[0]), custom_mask[1]]
    output3 = model(x, mask_all_zero)
    print(f"Выход: {output3.shape}")

    #Как задавать пользовательские маски
    #Пример 1: Фиксированная маска для всех примеров в батче
    fixed_mask = [
    torch.tensor([[1., 0., 1., 0.]] * batch_size).reshape(batch_size, -1),  # Первый
слой
    torch.tensor([[1., 1., 0.]] * batch_size).reshape(batch_size, -1)   # Второй слой
    ]
    output = model(x, fixed_mask)
    print(output)
```

Файл ppo_critical.py

```
# ppo_critical_cartpole.py
# PPO + Heuristic Risk + Learned Risk C_phi + Critical States + Metrics
# Compatible with old Gym API (CartPole-v0)

import gym
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import random
import matplotlib.pyplot as plt

env = gym.make("CartPole-v0")

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

gamma = 0.99
lr = 3e-4
clip_eps = 0.2
epochs = 5

lambda_c = 0.1
C_hard = 0.8
k_back = 10
p_reset = 0.4

alpha_start = 1.0
alpha_end = 0.0
transition_steps = 300

reward_history = []
length_history = []
mean_C_history = []
max_C_history = []
critical_history = []
fall_history = []
completion_history = []
alpha_history = []
c_loss_history = []
ppo_loss_history = []
```

```
last_traj_cheur = []
last_traj_cphi = []
last_traj_ctotal = []

#Эвристическое представление функции риска
def compute_C_heur(state):
    x, x_dot, theta, theta_dot = state

    theta_max = 0.209
    theta_dot_max = 2.0

    C_tilt = abs(theta) / theta_max
    C_angvel = abs(theta_dot) / theta_dot_max

    return min(1.0, 0.7 * C_tilt + 0.3 * C_angvel)


class ActorCritic(nn.Module):
    def __init__(self):
        super().__init__()

        self.shared = nn.Sequential(
            nn.Linear(4, 64),
            nn.Tanh(),
            nn.Linear(64, 64),
            nn.Tanh()
        )

        self.policy = nn.Linear(64, 2)
        self.value = nn.Linear(64, 1)

    def forward(self, x):
        x = self.shared(x)
        return self.policy(x), self.value(x)


class CriticalNet(nn.Module):
    def __init__(self):
        super().__init__()

        self.net = nn.Sequential(
            nn.Linear(4, 64),
            nn.ReLU(),
            nn.Linear(64, 64),
```

```
            nn.ReLU(),
            nn.Linear(64, 1),
            nn.Sigmoid()
        )

    def forward(self, x):
        return self.net(x)


model = ActorCritic().to(device)
C_model = CriticalNet().to(device)

optimizer = optim.Adam(model.parameters(), lr=lr)
C_optimizer = optim.Adam(C_model.parameters(), lr=1e-3)


class Buffer:
    def __init__(self):
        self.states = []
        self.actions = []
        self.log_probs = []
        self.rewards = []
        self.values = []
        self.dones = []
        self.Cs = []

    def clear(self):
        self.__init__()


buffer = Buffer()
C_dataset = []


def get_alpha(step):
    return max(alpha_end, alpha_start - step / transition_steps)


def compute_C_total(state, step):

    alpha = get_alpha(step)

    C_h = compute_C_heur(state)
```

```
    with torch.no_grad():
        s = torch.FloatTensor(state).to(device)
        C_l = C_model(s).item()

    C_total = alpha * C_h + (1 - alpha) * C_l

    return C_total, C_h, C_l


def select_action(state):

    s = torch.FloatTensor(state).to(device)

    logits, value = model(s)

    dist = torch.distributions.Categorical(logits=logits)

    action = dist.sample()

    return action.item(), dist.log_prob(action), value


def compute_returns(rewards, dones):

    returns = []
    G = 0

    for r, d in zip(reversed(rewards), reversed(dones)):
        if d:
            G = 0

        G = r + gamma * G
        returns.insert(0, G)

    return returns


def ppo_update():

    if len(buffer.states) == 0:
        return None

    states = torch.FloatTensor(buffer.states).to(device)
    actions = torch.LongTensor(buffer.actions).to(device)
```

```
old_log_probs = torch.stack(buffer.log_probs).detach()

returns = torch.FloatTensor(
    compute_returns(buffer.rewards, buffer.dones)
).to(device)

last_loss = None

for _ in range(epochs):

    logits, values = model(states)

    dist = torch.distributions.Categorical(logits=logits)

    new_log_probs = dist.log_prob(actions)

    ratio = (new_log_probs - old_log_probs).exp()

    advantages = returns - values.squeeze().detach()

    surr1 = ratio * advantages
    surr2 = torch.clamp(
        ratio,
        1 - clip_eps,
        1 + clip_eps
    ) * advantages

    policy_loss = -torch.min(surr1, surr2).mean()

    value_loss = 0.5 * (
        returns - values.squeeze()
    ).pow(2).mean()

    loss = policy_loss + value_loss

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    last_loss = loss.item()

return last_loss
```

```
def train_C_model():

    if len(C_dataset) < 100:
        return None

    batch = random.sample(
        C_dataset,
        min(64, len(C_dataset))
    )

    states, labels = zip(*batch)

    states = torch.FloatTensor(states).to(device)
    labels = torch.FloatTensor(labels).unsqueeze(1).to(device)

    preds = C_model(states)

    positives = labels.sum().item()
    negatives = len(labels) - positives

    if positives > 0:
        w1 = negatives / max(positives, 1.0)
    else:
        w1 = 1.0

    weights = torch.where(
        labels > 0.5,
        torch.tensor(w1, device=device),
        torch.tensor(1.0, device=device)
    )

    loss = nn.BCELoss(weight=weights)(preds, labels)

    C_optimizer.zero_grad()
    loss.backward()
    C_optimizer.step()

    return loss.item()


critical_buffer = []

global_step = 0
num_episodes = 1000
```

```
for episode in range(num_episodes):

    if len(critical_buffer) > 0 and random.random() < p_reset:
        state = np.array(random.choice(critical_buffer))
        env.state = state.copy()
    else:
        state = env.reset()

    trajectory = []

    episode_reward = 0
    episode_C = []

    critical_termination = False
    real_fall = False

    traj_cheur = []
    traj_cphi = []
    traj_ctotal = []

    for t in range(200):

        action, log_prob, value = select_action(state)

        next_state, reward, done, _ = env.step(action)

        C, C_h, C_l = compute_C_total(
            state,
            global_step
        )

        traj_cheur.append(C_h)
        traj_cphi.append(C_l)
        traj_ctotal.append(C)

        reward_mod = reward - lambda_c * C

        buffer.states.append(state)
        buffer.actions.append(action)
        buffer.log_probs.append(log_prob)
        buffer.rewards.append(reward_mod)
        buffer.values.append(value)
        buffer.dones.append(done)
```

```
    buffer.Cs.append(C)

    trajectory.append(state)

    episode_reward += reward_mod
    episode_C.append(C)

    if C >= C_hard:

        critical_termination = True
        done = True

        for i in range(len(trajectory)):
            label = 1 if i >= max(
                0,
                len(trajectory) - k_back
            ) else 0

            C_dataset.append(
                (trajectory[i], label)
            )

        if len(trajectory) > k_back:
            critical_buffer.append(
                trajectory[-k_back]
            )

    if done:
        break

    state = next_state
    global_step += 1

if done and not critical_termination and t < 199:
    real_fall = True

ppo_loss = ppo_update()
c_loss = train_C_model()

reward_history.append(episode_reward)
length_history.append(t + 1)

mean_C_history.append(
    np.mean(episode_C)
```

```
            if len(episode_C) > 0 else 0
        )

        max_C_history.append(
            np.max(episode_C)
            if len(episode_C) > 0 else 0
        )

        critical_history.append(
            int(critical_termination)
        )

        fall_history.append(
            int(real_fall)
        )

        completion_history.append(
            int(t >= 199)
        )

        alpha_history.append(
            get_alpha(global_step)
        )

        if c_loss is not None:
            c_loss_history.append(c_loss)

        if ppo_loss is not None:
            ppo_loss_history.append(ppo_loss)

        last_traj_cheur = traj_cheur
        last_traj_cphi = traj_cphi
        last_traj_ctotal = traj_ctotal

        buffer.clear()

        print(
            f"Episode={episode} "
            f"Len={t+1} "
            f"Reward={episode_reward:.1f} "
            f"Alpha={get_alpha(global_step):.3f}"
        )

plt.figure(figsize=(16, 12))
```

```
plt.subplot(3, 3, 1)
plt.plot(reward_history)
plt.title("Episode Reward")

plt.subplot(3, 3, 2)
plt.plot(length_history)
plt.title("Episode Length")

plt.subplot(3, 3, 3)
plt.plot(mean_C_history)
plt.title("Mean Risk")

plt.subplot(3, 3, 4)
plt.plot(max_C_history)
plt.title("Max Risk")

plt.subplot(3, 3, 5)
plt.plot(np.cumsum(critical_history))
plt.title("Critical Stops")

plt.subplot(3, 3, 6)
plt.plot(np.cumsum(fall_history))
plt.title("Falls")

window = 20
completion_rate = []

for i in range(len(completion_history)):
    start = max(0, i - window)
    completion_rate.append(
        np.mean(completion_history[start:i+1])
    )

plt.subplot(3, 3, 7)
plt.plot(completion_rate)
plt.title("Completion Rate")

plt.subplot(3, 3, 8)
plt.plot(alpha_history)
plt.title("Alpha Transition")

plt.subplot(3, 3, 9)
plt.plot(c_loss_history)
```

```
plt.title("C_phi Loss")

plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 5))

plt.plot(last_traj_cheur, label="C_heur")
plt.plot(last_traj_cphi, label="C_phi")
plt.plot(last_traj_ctotal, label="C_total")

plt.legend()
plt.title("Risk Evolution on Last Trajectory")
plt.xlabel("Step")
plt.ylabel("Risk")

plt.show()
```

Файл SE3.py

```
"""
Модуль кинематики шагающего робота
--Производится расчёт скорости сочленений и её фильтрация
--Производится получение корректных данных гироскопа-акселерометра
--Кодируется применение SE(3) группы и подхода PoE
"""
import numpy as np

#
==========================================================
==========
# ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ ДЛЯ ПОЛУЧЕНИЯ ДАННЫХ
СКОРОСТИ СОЧЛЕНЕНИЙ
#
==========================================================
==========
def make_filtered_joints_velocities(current_joints_angles, previous_joints_angles,
previous_velocity, alpha, negative_speed_limit, positive_speed_limit):
    """
        Метод, выполняющий фильтрацию скорости угловых сочленений
        конечностей робота
    """
    # считаем сырую скорость
```

```
    raw_velocity = (current_joints_angles - previous_joints_angles) / dt
    # фильтрация скорости
    filtered_velocity = alpha * raw_velocity + (1 - alpha) * previous_velocity
    # ограничение скорости
    filtered_velocity = np.clip(filtered_velocity, -negative_speed_limit,
positive_speed_limit)

    return filtered_velocity

#
==========================================================================
==========
# ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ ДЛЯ ПОЛУЧЕНИЯ ДАННЫХ
ГИРОСКОПА-АКСЕЛЕРОМЕТРА
#
==========================================================================
==========

def vector_normalization(vector):
    """
        Производит нормализацию вектора значений
    """
    normolized_vector = np.linalg.norm(vector)
    if normolized_vector == 0:
        return vector
    return vector / normolized_vector


def quaternions_multiplication(q, r):
    """
        Выполняет операцию q ⊗ r
    """
    w0, x0, y0, z0 = q
    w1, x1, y1, z1 = r

    return np.array([
        w0*w1 - x0*x1 - y0*y1 - z0*z1,
        w0*x1 + x0*w1 + y0*z1 - z0*y1,
        w0*y1 + y0*w1 + z0*x1 - x0*z1,
        w0*z1 + z0*w1 + x0*y1 - y0*x1
    ])


def quaternion_normalization(q):
```

```
    """
        Производит нормализацию кватерниона
    """
    return q / np.linalg.norm(q)


def quaternion_to_rotation_matrix(q):
    """
        Преобразует кватернион в матрицу вращения
    """
    x, y, z, w = q

    return np.array([
        [1 - 2*(y*y + z*z), 2*(x*y - z*w),     2*(x*z + y*w)],
        [2*(x*y + z*w),     1 - 2*(x*x + z*z), 2*(y*z - x*w)],
        [2*(x*z - y*w),     2*(y*z + x*w),     1 - 2*(x*x + y*y)]
    ])

def compute_gravity_data_from_quaternion(q):
    """
        Получение вектора гравитации g(q)
        из кватерниона
    """
    w, x, y, z = q

    return np.array([
        2*(x*z - w*y),
        2*(y*z + w*x),
        1 - 2*(x*x + y*y)
    ])


# ============================
# ФИЛЬТР МАХОНИ(MAHONY)
# ============================

class MahonyFilter:
    def __init__(self, Kp=2.0, Ki=0.05, sample_frequancy=200.0):
        self.Kp = Kp
        self.Ki = Ki
        self.dt = 1.0 / sample_frequancy

        self.q = np.array([1.0, 0.0, 0.0, 0.0])  # начальная ориентация
        self.bias = np.array([0.0, 0.0, 0.0])    # bias гироскопа
```

```
    def update(self, gyro_data, accelerations):
        """
        gyro_data: [gx, gy, gz] (rad/s)
        accelerations:  [ax, ay, az]
        """

        # 1. нормализация акселерометра
        accelerations = vector_normalization(accelerations)

        # 2. оценка гравитации из q
        gravity_estimation = compute_gravity_data_from_quaternion(self.q)

        # 3. ошибка
        error = np.cross(accelerations, gravity_estimation)

        # 4. обновление bias
        self.bias += self.Ki * error * self.dt

        # 5. коррекция гироскопа
        gyro_data_corrected = gyro_data - self.bias + self.Kp * error

        # 6. считаем q_dot
        q = self.q
        gx, gy, gz = gyro_data_corrected

        q_dot = 0.5 * np.array([
            -q[1]*gx - q[2]*gy - q[3]*gz,
             q[0]*gx + q[2]*gz - q[3]*gy,
             q[0]*gy + q[3]*gx - q[1]*gz,
             q[0]*gz + q[1]*gy - q[2]*gx
        ])

        # 7. интеграция
        self.q += q_dot * self.dt

        # 8. нормализация
        self.q = quaternion_normalization(self.q)

        return self.q


# ========================
# КАЛИБРОВКА BIAS
```

```
# ==========================

def calibrate_gyro_bias(gyro_data):
    """
    gyro_data: список измерений [(gx, gy, gz), ...]
    устройство должно быть неподвижно
    """
    return np.mean(gyro_data, axis=0)

"""
# ==========================
# ПРИМЕР ИСПОЛЬЗОВАНИЯ
# ==========================

# Создаем фильтр
filter_mahony = MahonyFilter(Kp=2.0, Ki=0.05, sample_frequancy=200)

# --- КАЛИБРОВКА ---
# (пример: 100 измерений стоящего устройства)
gyro_samples = [np.array([0.01, -0.02, 0.005]) for _ in range(100)]
filter_mahony.bias = calibrate_gyro_bias(gyro_samples)

print("Initial bias:", filter_mahony.bias)

# --- ОСНОВНОЙ ЦИКЛ ---
for i in range(1000):

    # Пример данных (замени на реальные IMU!)
    gyro = np.array([0.01, 0.02, 0.015])  # rad/s
    acc  = np.array([0.0, 0.0, 1.0])      # гравитация

    # обновление
    quaternion = filter_mahony.update(gyro, acc)

    # матрица вращения
    rotation_matrix = quaternion_to_rotation_matrix(quaternion)

    if i % 100 == 0:
        print("quaternion:", quaternion)
        print("rotation_matrix:\n", rotation_matrix)
        print()
"""

# ============================================================
```

```
# SE(3) МАТЕМАТИКА И ПОДХОД PRODUCTS OF EXPONENTIALS
# ==================================================

def make_screw_matrix(w):
    return np.array([
        [0, -w[2], w[1]],
        [w[2], 0, -w[0]],
        [-w[1], w[0], 0]
    ])


def make_transform_matrix(rotation_matrix, position_vector):
    t_matrix = np.eye(4)
    t_matrix[:3, :3] = rotation_matrix
    t_matrix[:3, 3] = position_vector
    return t_matrix


# ==========================
# POE КИНЕМАТИКА НОГИ
# ==========================

def exponential_twist(screw, theta):
    w = screw[:3]
    v = screw[3:]

    w_hat = make_screw_matrix(w)

    rotation_matrix = (
        np.eye(3)
        + np.sin(theta) * w_hat
        + (1 - np.cos(theta)) * (w_hat @ w_hat)
    )

    G = (
        np.eye(3) * theta
        + (1 - np.cos(theta)) * w_hat
        + (theta - np.sin(theta)) * (w_hat @ w_hat)
    )

    position_vector = G @ v

    transformation = np.eye(4)
    transformation[:3, :3] = rotation_matrix
```

```
    transformation[:3, 3] = position_vector

    return transformation


def leg_forward_kinematics(screws_list, start_robot_configuration, thetas):
    transformation = np.eye(4)
    for screws_list, theta in zip(S_list, thetas):
        transformation = transformation @ exponential_twist(screws_list, theta)
    return transformation @ start_robot_configuration


# ============================
# ОПИСАНИЕ НОГИ
# ============================

def get_robot_joints_screws(L1, L2, L3, L4):
    S1 = np.array([0, 0, 1, 0, 0, 0])
    S2 = np.array([1, 0, 0, 0, -L1, 0])
    S3 = np.array([1, 0, 0, 0, -(L1 + L2), 0])
    S4 = np.array([1, 0, 0, 0, -(L1 + L2 + L3), 0])
    S5 = np.array([1, 0, 0, 0, -(L1 + L2 + L3 + L4), 0])
    return [S1, S2, S3, S4, S5]


def get_M(L1, L2, L3, L4):
    M = np.eye(4)
    M[2, 3] = -(L1 + L2 + L3 + L4)
    return M


# ============================
# ОЦЕНКА КОРПУСА
# ============================

def foot_position_in_body(T_BL, S_list, M, thetas):
    T_LE = leg_forward_kinematics(S_list, M, thetas)
    T_BE = T_BL @ T_LE
    return T_BE[:3, 3]


def estimate_pwb(p_foot_world, R_WB, p_BE):
    return p_foot_world - R_WB @ p_BE
```

```
"""
# ============================
# ПРИМЕР ИСПОЛЬЗОВАНИЯ
# ============================

# ---- Геометрия ноги ----
L1, L2, L3, L4 = 0.05, 0.1, 0.1, 0.05

S_list = get_robot_joints_screws(L1, L2, L3, L4)
M = get_M(L1, L2, L3, L4)

# ---- Положение ноги на корпусе ----
# левая нога (например справа по Y)
T_BL = make_transform_matrix(np.eye(3), np.array([0.0, 0.1, 0.0]))

# ---- Углы суставов ----
thetas = [0.0, 0.3, -0.5, 0.2, 0.1]

# ---- IMU (кватернион) ----
q_imu = [0, 0, 0, 1]  # без вращения

R_WB = quaternion_to_rotation_matrix(q_imu)

# ---- "Якорь" — стопа на земле ----
p_foot_world = np.array([0.2, 0.1, 0.0])

# ---- FK ноги ----
p_BE = foot_position_in_body(T_BL, S_list, M, thetas)

# ---- Оценка позиции корпуса ----
p_WB = estimate_pwb(p_foot_world, R_WB, p_BE)

# ---- Итоговая матрица ----
T_WB = make_transform_matrix(R_WB, p_WB)

# ---- Вывод ----
print("Foot in body frame:", p_BE)
print("Body position p_WB:", p_WB)
print("T_WB:")
print(T_WB)
"""
```