Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»
Институт радиоэлектроники и информационных технологий - РТФ
Кафедра информационных технологий и систем управления

ДОПУСТИТЬ К ЗАЩИТЕ ПЕРЕД ГЭК

Зав. кафедрой ИТиСУ
_____ Е.В. Кислицын
(подпись) (Ф.И.О.)
«03» 06 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА СИСТЕМЫ МАШИННОГО ОБУЧЕНИЯ НА БАЗЕ UNITY ML-AGENT ДЛЯ СИМУЛЯТОРА РОБОТА

Научный руководитель: Денисов Дмитрий Вадимович
к.т.н., доцент _____
подпись

Нормоконтролер: Бредихина Наталья Сергеевна _____
подпись

Студент группы: РИМ-220907 Осенчугов Никита Андреевич _____
подпись

Екатеринбург
2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
**«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»**

Институт радиоэлектроники и информационных технологий - РТФ
Кафедра информационных технологий и систем управления
Направление подготовки 09.04.01 Информатика и вычислительная техника
Образовательная программа Инженерия искусственного интеллекта

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студента Осенчугова Никиты Андреевича группы РИМ-220907
(фамилия, имя, отчество)

1. Тема выпускной квалификационной работы Разработка системы машинного обучения на базе Unity ML-Agent для симулятора робота

Утверждена распоряжением по институту от «4» декабря 2023 г. № 33.02-05/298

2. Научный руководитель Денисов Дмитрий Вадимович, доцент, кандидат технических наук
(Ф.И.О., должность, ученая степень, ученое звание)

3. Исходные данные к работе 3D-модель робота-манипулятора

4. Перечень демонстрационных материалов *презентация в MS PowerPoint*

5. Календарный план

№ п/п	Наименование этапов выполнения работы	Срок выполнения этапов работы	Отметка о выполнении
1.	Глава 1. Анализ существующих симуляторов ML-Agent и методов их разработки	до 23.03.2024 г.	
2.	Глава 2. Создание виртуальной среды	до 29.04.2024 г.	
3.	Глава 3. Оценка результатов обучения	до 19.05.2024 г.	
4.	ВКР в целом	до 24.05.2024 г.	

Научный руководитель Денисов Дмитрий Вадимович — С. Денисов
Ф.И.О. (подпись)

Студент задание принял к исполнению 25.02.2024 — (подпись)
дата

6. Допустить Осенчугова Никиту Андреевича к защите выпускной квалификационной работы в экзаменационной комиссии

Зав. кафедрой ИТиСУ (подпись) Е.В. Кислицын
Ф.И.О.

РЕФЕРАТ

Магистерская диссертация 60 с., 20 рис., 1 табл., 51 источн., 0 прил.

ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ, UNITY ML-AGENT, АГЕНТ МАШИННОГО ОБУЧЕНИЯ, РОБОТОТЕХНИКА, УПРАВЛЕНИЕ РОБОТОМ-МАНИПУЛЯТОРОМ, ВИРТУАЛЬНАЯ СРЕДА UNITY

Объект исследования – разработка среды для обучения агента, управляющего действиями модели робота-манипулятора.

Предмет исследования – применение технологии Unity ML-Agent для обучения агента, контролирующего действия модели робота-манипулятора.

Цель работы – разработка системы для обучения агента Unity ML-Agents, управляющего моделью робота-манипулятора, достижению целевого объекта.

Методы исследования: математическое моделирование, анализ данных, экспериментальный метод.

Результатом магистерской работы является успешное создание системы машинного обучения для управления роботом-манипулятором в виртуальной среде Unity.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 6

1 Анализ существующих симуляторов ML-Agents и методов их разработки .. 9

1.1 Анализ существующих симуляторов ML-Agents 9

1.2 Анализ существующих методов машинного обучения симуляторов ML-Agents 12

1.3 Параметры машинного обучения ML-Agents 14

1.4 Преимущества реализации цифрового двойника 17

1.5 Вывод 18

2 Создание виртуальной среды 19

2.1 Используемые библиотеки и модули 19

2.1.1 Модуль ML-Agents для Unity 19

2.1.2 mlagents-learn 20

2.1.3 TensorFlow 20

2.1.4 Tensorboard 21

2.1.5 Pytorch 22

2.2 Модель робота-манипулятора 23

2.3 Создание среды для обучения агента 24

2.3.1 Сцена для обучения агента 26

2.3.2 Сцена с обученным агентом 27

2.4 Наблюдения и действия агента 28

2.4.1 Наблюдения 28

2.4.2 Действия 28

2.5 Награда за обучение 29

2.6 Тип обучения 30
2.7 Конфигурация гиперпараметров обучения 33
2.8 Запуск обучения 37
2.9 Вывод 38
3 Оценка результатов обучения 40
3.1 Критерии оценки 40
3.2 Визуализация в TensorBoard 42
3.3 Стратегии обучения и возникшие проблемы 42
3.4 Оценка эпизодов обучения 49
3.5 Вывод 50
ЗАКЛЮЧЕНИЕ 52
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 55

ВВЕДЕНИЕ

Первоначальный успех алгоритма обучения deep Q-learning на Atari продемонстрировал полезность симуляторов в обучении с подкреплением и исследованиях интеллекта на основе агентов машинного обучения в целом [1–2].

С тех пор было создано множество игровых сред и платформ, включая OpenAI Gym Retro, Universe, ProcGen, Hide&Seek, мультиагентную среду частиц, OpenSpiel от DeepMind, Lab и Control Suite, обучающая среда NetHack от FAIR, Unity ML-Agents и другие, включая VizDoom, PommerMan, Griddly и MineRL [3–15]. Такой широкий спектр задач создал пространство для разработки стабильных методов обучения на основе агентов машинного обучения. Крупномасштабные проекты обучения с подкреплением победили даже профессионалов в Go, DoTA 2 и StarCraft 2 [16–18].

В настоящее время такие компании, как автопроизводители, устанавливают и программируют роботов для выполнения задач на производственных площадях [19]. Это дорого, и не всегда есть уверенность в том, что будет принятый алгоритм будет оптимальным [20]. Более того, если на производственной линии происходят какие-либо изменения, величина этого влияния на эффективность производства часто не учитывается при разработке новых алгоритмов. Предполагается, что внедрение агента ML-Agents позволит расширить возможности робота-манипулятора.

Роботизированные манипуляторы в контексте производства должны не только находить наиболее эффективное решение поставленной задачи, но и быть способны адаптироваться в режиме реального времени к меняющимся ситуациям. Манипуляторы-роботы, как правило, запрограммированы на детерминированное выполнение заданной задачи. Для полностью автономного завода такие устройства, как роботы-манипуляторы, автоматизированные управляемые транспортные средства или даже системы заказа доставки, имеют преимущества, поскольку они интегрированы в одну

взаимосвязанную систему и управляются ИИ. Эти преимущества варьируются от повышения производительности и выпуска продукции до повышения окупаемости инвестиций в аппаратное обеспечение. Такие компании, как Microsoft, Amazon и Google, предоставляют инженерам платформы, фреймворки и инструменты для обучения ИИ. Игровые движки, такие как Unity и Unreal Engine 4, не только помогают инженерам лучше понимать и оценивать обученные данные, но и могут моделировать процесс на этапе обучения, обеспечивая большую устойчивость к изменениям в производственных процессах. Это позволяет обучать несколько ИИ одновременно и, таким образом, увеличивает стабильность тренировок, а также сокращает их длительность.

Цель данной работы – разработать систему для обучения агента Unity ML-Agents, управляющего моделью робота-манипулятора, достижению целевого объекта.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1) Проанализировать существующие симуляторы ML-Agents,
2) Разработать обучающую среду в Unity,
3) Обучить агента в обучающей среде,
4) Оценить качество обучения агента.

Объектом исследования является разработка среды для обучения агента, управляющего действиями модели робота-манипулятора.

Предметом исследования является применение технологии Unity ML-Agent для обучения агента, контролирующего действия модели робота-манипулятора.

При написании данной работы использовались следующие методы: математическое моделирование, анализ данных, экспериментальный метод.

Научная новизна работы заключается в следующем:

Разработана среда машинного обучения агента, управляющего действиями робота-манипулятора в среде Unity.

Приведена классификация методов и алгоритмов обучения агентов в симуляционных средах.

Разработан и протестирован алгоритм обучения агента с использованием метода Proximal Policy Optimization для управления робот-манипулятором.

Практической ценностью работы является то, что разработанная система управления роботом-манипулятором с использованием алгоритма PPO может быть применена в реальных промышленных и образовательных проектах.

Результатом магистерской работы является успешное создание системы машинного обучения для управления роботом-манипулятором в виртуальной среде Unity, что позволяет данному подходу конкурировать с существующими методами в задачах робототехники.

Данная работа состоит из введения, трех разделов, заключения и списка использованных источников, включающего 51 наименований. Работа изложена на 60 страницах машинного текста, иллюстрирована 20 рисунками и сопровождается 1 таблицей.

1 Анализ существующих симуляторов ML-Agents и методов их разработки

1.1 Анализ существующих симуляторов ML-Agents

Набор для разработки программного обеспечения Unity Machine Learning Agents (ML-Agents SDK) – это плагин с открытым исходным кодом для Unity, который позволяет создавать среды моделирования с помощью редактора Unity. Исследователи и разработчики могут легко взаимодействовать с ML-Agents с помощью простого в использовании Python API. Эти агенты могут быть обучены с использованием обучения с подкреплением, имитационного обучения или других методов машинного обучения. Инструментарий Unity ML-Agents предлагает набор основных функциональных возможностей, который позволяет разработчикам создавать среды машинного обучения с помощью редактора Unity и связанных с ним скриптов C#, которые затем могут предоставлять среды для взаимодействия с использованием API Python. Unity ML-Agents основана на TensorFlow, который является платформой машинного обучения, созданной Google. TensorFlow позволяет разработчикам сосредоточиться на общей логике своих приложений и не беспокоиться о реализации сложных алгоритмов. На Рисунке 1 представлены простые примеры сред с использованием Unity ML-Agents.

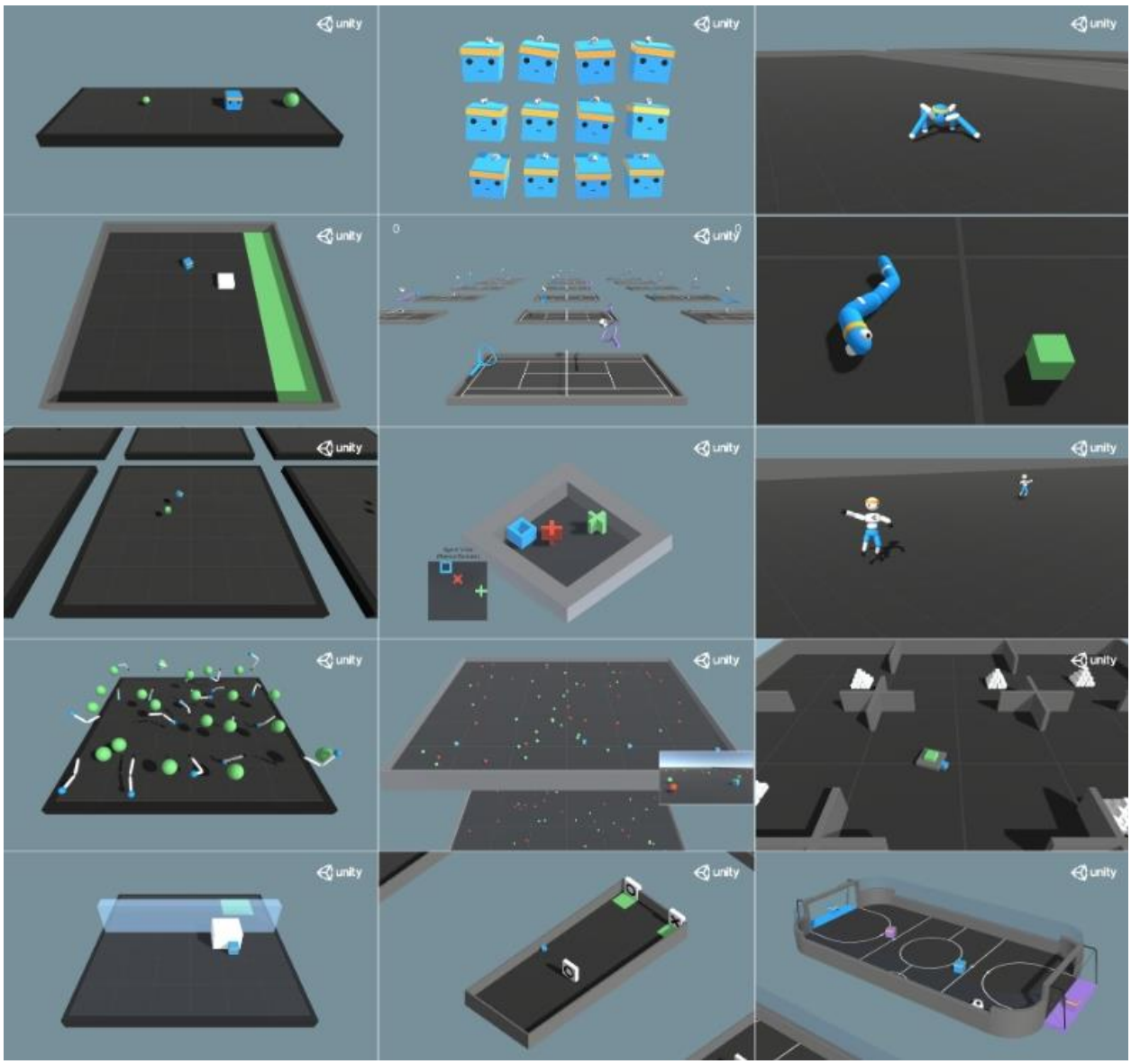

Рисунок 1. Изображения примеров сред Unity ML-Agents Toolkit. Слева направо, сверху вниз: (1) Базовая среда, (2) 3D Шар, (3) Ползун, (4) Движение блока, (5) Теннис, (6) Червь, (7) Вышибала, (8) Сеточный мир, (9) Ходунки, (10) Ричер, (11) Сборщик еды, (12) Пирамиды, (13) Прыжок через стену, (14) Коридор, (15) Футбол.

Ниже приведено краткое описание цели агента в каждой из сред, представленных на Рисунке 1:

1) Базовая среда – задача линейного перемещения, в которой агент (синий кубик) должен перемещаться влево или вправо, чтобы получать

вознаграждение. Цель состоит в том, чтобы перейти в наиболее выгодное состояние;

2) 3D-Шар – Задача с балансирующим мячом, в которой агент управляет вращением платформы. Цель состоит в том, чтобы сбалансировать платформу, чтобы удерживать мяч на ней как можно дольше;

3) Ползун - основанные на физике существа с 4 руками и 4 предплечьями. Цель состоит в том, чтобы двигаться в направлении цели как можно быстрее, не падая;

4) Движение блока – среда платформера, в которой агент может перемещать блок. Цель состоит в том, чтобы переместить блок в целевую область (черно-белая сетка);

5) Теннис – игра для двух игроков, в которой агенты управляют ракетками, чтобы мяч отскакивал от сетки. Цель состоит в том, чтобы отскочить мячом в другую сторону, а не уронить мяч или отправить его за пределы поля;

6) Червь – основанный на физике агент передвижения с тремя суставами, который должен двигаться к цели как можно быстрее;

7) Вышибала – прыгающее задание, в котором агент (синий кубик) может прыгать с определенной скоростью и угол, когда он коснется земли. Цель состоит в том, чтобы поймать плавающий пищевой объект за как можно меньшее количество прыжков;

8) Сеточный мир – версия классической задачи grid-world. Сцена содержит агента (синий квадрат), цель и препятствия. Цель состоит в том, чтобы двигаться к цели, избегая препятствий;

9) Ходунки – гуманоиды, основанные на физике, с 26 степенями свободы частей тела. Цель состоит в том, чтобы двигаться в направлении цели как можно быстрее, не падая;

10) Ричер – рука с двумя суставами, которая может перемещаться в заданные места. Цель состоит в том, чтобы переместить ее руку к целевому местоположению (зеленая сфера) и удерживать ее там;

11) Сборщик еды – Мультиагентная среда, в которой агенты (синий куб) соревнуются за сбор бананов. Цель состоит в том, чтобы собрать как можно больше желтых бананов, избегая синих бананов;

12) Пирамиды – среда, в которой агенту (синий куб) нужно нажать кнопку, чтобы создать пирамиду, затем перейти к пирамиде, опрокинуть ее и перейти к золотому кирпичу наверху. Цель состоит в том, чтобы переместиться к золотому кирпичу на вершине созданной пирамиды;

13) Прыжок через стену – среда-платформер со стеной и желтым блоком, который может быть толкаемый, и агент (синий куб), который может двигаться, вращаться и прыгать. Цель состоит в том, чтобы достичь цели (черно-белая сетка) по другую сторону стены. Если стена слишком высока, агенту иногда нужно подтолкнуть белый блок к стене, запрыгнуть на него, чтобы достичь своей цели. Агент разрабатывает две стратегии — одну для больших стен (требуется маленький блок) и одну для маленьких стен;

14) Коридор – среда, в которой агенту (синий куб) необходимо найти информацию в комнате, запомнить ее и использовать для перемещения к нужной цели. Цель состоит в том, чтобы переместиться к цели (черно-белая сетка), которая соответствует цвету блока в комнате;

15) Футбол – среда, в которой четыре агента соревнуются в игрушечном футболе 2 на 2. Все агенты равны, и перед ними стоит задача не допустить попадания мяча в свои ворота и забить гол в ворота соперника.

1.2 Анализ существующих методов машинного обучения симуляторов ML-Agents

В этом разделе рассмотрены основные методы машинного обучения ML-Agents: обучение с учителем, обучение без учителя и обучение с подкреплением.

а) Обучение с учителем в машинном обучении основывается на разметке собранных данных, чтобы точно указать машине, что искать. При

обучении алгоритму требуется полный набор помеченных данных. Алгоритм наблюдает и идентифицирует конкретные закономерности в данных, позволяющие компьютеру извлекать уроки из наблюдений. Благодаря этому компьютер повышает производительность прогнозирования по мере того, как он просматривает больше наблюдений;

б) При обучении с учителем должен быть предоставлен полный помеченный набор данных для изучения и идентификации одних и тех же закономерностей в данных, в то время как обучение без учителя происходит наоборот. При обучении без учителя передается немаркированный набор данных без каких-либо инструкций о том, что машина должна с ним делать. Обучающий набор данных не содержит какого-либо конкретного результата или правильного ответа, в отличие от контролируемого обучения, и нейронная сеть автоматически попытается найти структуру в данных. Обучение без учителя автоматически найдет и извлечет конкретные признаки и паттерны в данных. Кластеризация - это способ для модели обучения без учителя организовать данные, опираясь на конкретные признаки, которые затем группируются вместе;

в) Обучение с подкреплением (RL) - это когда агент принимает последовательность решений для достижения заданной цели в окружающей среде. Агент искусственного интеллекта будет методом проб и ошибок находить наилучшее возможное решение проблемы, и при этом награждается за правильные действия, которые он выполняет и получает штрафы за негативные действия, которые он не должен делать. Таким образом, агент искусственного интеллекта будет делать то, что хочет программист. Обучение с подкреплением состоит из трех основных компонентов. Агент, который является лицом, принимающим решения в модели обучения, среда, с которой агент будет взаимодействовать, и действия, которые представляют собой список решений, которые может принять агент для взаимодействия со средой.

По умолчанию в среде Unity ML-Agents используется обучение с подкреплением, но в дополнении к этому есть возможность записывать

демонстрационные фрагменты, чтобы дать агенту более наглядное представление желаемого поведения.

1.3 Параметры машинного обучения ML-Agents

В этом разделе описаны ключевые параметры, лежащие в основе технологии ML-Agents. От их грамотной настройки зависит производительность и результаты обучения агента. Оптимальные параметры во многом зависят от конкретной задачи, поэтому подбираются путем экспериментов в ряде последовательных итераций. Ниже представлены основополагающие параметры машинного обучения, необходимые для построения обучающей среды:

а) Архитектура нейронной сети: В ML-Agents обычно используются различные типы нейронных сетей, такие как полносвязные (fully connected), сверточные (convolutional), рекуррентные (recurrent) и их комбинации. Выбор определенного типа зависит от типа входных данных и природы задачи.

Помимо этого, важным аспектом архитектуры является выбор количества слоев и нейронов в каждом слое. Глубокие сети с большим количеством слоев могут выучивать более сложные зависимости в данных, но при этом требуют больше вычислительных ресурсов и времени для обучения. Количество нейронов в каждом слое также может варьироваться в зависимости от задачи.

В дополнении к этому, необходимо подобрать функцию активации, которая определяет нелинейные свойства нейронной сети. Различные функции активации могут быть более или менее подходящими для конкретной задачи. Например, ReLU (Rectified Linear Unit) часто используется из-за своей простоты и эффективности, но для некоторых задач могут быть предпочтительны другие функции, такие как tanh или sigmoid.

Для предотвращения переобучения нейронной сети могут применяться различные методы регуляризации, такие как Dropout, L2-регуляризация и другие.

Наконец, структура входных данных, которые предоставляются нейронной сети, также влияет на ее архитектуру. Например, если входные данные представляют собой изображения, то для них часто применяются сверточные слои. Если данные имеют последовательную структуру (например, временные ряды), то могут использоваться рекуррентные слои. В случае ML-Agents, наблюдениями являются те или иные переменные, которые содержат компоненты, закрепленные в свою очередь за объектами на сцене;

б) Гиперпараметры алгоритма обучения: Это параметры, которые необходимо настроить для оптимального обучения агентов. Они включают в себя скорость обучения, коэффициент снижения, параметры оптимизатора, размер пакета и другие. Конфигурация гиперпараметров для обучения робота-манипулятора приведена в Главе 2.6;

в) Структура окружающей среды: Изменение параметров среды, таких как размеры, скорости объектов, наличие препятствий и других факторов, может повлиять на сложность задачи и, следовательно, на процесс обучения. Размер среды может напрямую влиять на производительность алгоритма обучения.

Каждый объект в среде может иметь различные свойства, такие как форма, размер, цвет, текстура, масса и т.д. Эти свойства могут определять, как агенты взаимодействуют с объектами и какие решения они принимают.

В некоторых средах объекты могут быть подвержены динамическим изменениям, таким как движение, изменение формы, появление и исчезновение. Это могут быть физические объекты, такие как мячи, или агенты в среде, которые также обучаются.

Наконец, обратная связь, которую получают агенты от среды, и данные, которые агенты видят или получают из среды, также являются важной частью структуры окружающей среды. Эти данные определяют, как агенты

воспринимают окружающую среду и какие решения они принимают на основе этой информации;

г) Методы наград и функции штрафов: Методы наград и функции штрафов определяют, какие действия агентов в окружающей среде считаются желательными и нежелательными, соответственно. Они играют ключевую роль в обучении агентов методами обучения с подкреплением.

Функция вознаграждения – это функция, которая оценивает действия агента и назначает им численную награду в зависимости от того, насколько они приближают агента к достижению цели. Функция вознаграждения должна быть разработана в соответствии с конкретной задачей и целями обучения. Она может быть как простой, так и сложной, и может включать в себя различные компоненты и условия. Например, в задаче управления роботом-манипулятором функция вознаграждения может награждать агента за достижение некого объекта.

Штрафная функция – это функция, которая оценивает действия агента и назначает им отрицательную награду (или штраф) в случае выполнения нежелательных действий. Функция штрафов помогает агенту избегать нежелательных поведенческих стратегий и сосредоточиться на достижении целей. Она может быть использована в сочетании с функцией вознаграждения для более точного управления поведением агента. Например, в задаче управления роботом-манипулятором функция штрафов может штрафовать за столкновения с препятствиями.

Подробнее достоинства различных вариантов функций вознаграждения и штрафных функций в контексте построения обучающей среды для робота-манипулятора описаны в Главе 2.5;

д) Алгоритмы и методы обучения: ML-Agents поддерживает различные алгоритмы обучения, такие как PPO (Proximal Policy Optimization), SAC (Soft Actor-Critic) и др. Выбор подходящего алгоритма может существенно повлиять на скорость обучения и качество получаемых

результатов. Подробнее преимущества того или иного алгоритма в контексте построения среды для обучения робота-манипулятора описаны в Главе 2.4.

1.4 Преимущества реализации цифрового двойника

Цифровые двойники становятся неотъемлемой частью развития многих компаний как в России, так и за рубежом, применяемые в различных отраслях. Они значительно облегчают поддержку системы, экономят ресурсы предприятия и сокращают вероятность ошибок и сбоев, что продлевает срок службы оборудования. Все эти преимущества способствуют увеличению прибыли от инвестиций, повышению конкурентоспособности и укреплению лояльности клиентов.

Исследование рынка, проведенное компанией MarketsandMarkets, специализирующейся на анализе бизнес-рынков, показывает внушительный рост глобального рынка цифровых двойников. С 3,1 миллиарда долларов США в 2020 году объем рынка увеличился до 48,2 миллиарда долларов США. Ожидается, что рынок будет расти со среднегодовым темпом 58% [21]. Аналитики Market Research Future прогнозируют, что к концу 2025 года объем рынка достигнет 35,462 миллиарда долларов США, средний годовой темп роста составит 42,54% [22]. Еще в 2021 году Gartner прогнозировали, что в течение года более половины промышленных компаний будут использовать цифровых двойников, увеличивая эффективность деятельности на 10% [23].

Использование цифровых двойников значительно упрощает доступ к данным для клиентов и снижает затраты на обслуживание в долгосрочной перспективе. Это позволяет им принимать более обоснованные решения относительно изменений в рабочих процессах и существенно экономить ресурсы, оптимизируя обслуживание и повышая операционную эффективность. Успешное внедрение цифрового двойника начального проекта сказывается на последующих этапах работы, поскольку большая часть расходов связана с проектированием и строительством.

Использование цифровых двойников при обслуживании и эксплуатации помогает оптимизировать операционную деятельность, сокращает время простоя и расходы на обслуживание и персонал. Возможность работы с данными в реальном времени изменяет подход к принятию решений, делая его более обоснованным и эффективным. Визуализация и симуляция сложных операций в 3D-среде не только оптимизируют работу с ресурсами, но и открывают новые возможности для создания и использования физических объектов и пространств [24].

1.5 Вывод

В данной главе был проведен анализ существующих симуляторов ML-Agents и методов их разработки, а также рассмотрены преимущества внедрения цифровых двойников в производственные процессы.

Применение разнообразных сред, разработанных на платформе Unity ML-Agents, подчеркивает значимость использования симуляторов для обучения с подкреплением и исследований в области машинного обучения. Такие инструменты позволяют создавать среды моделирования, в которых агенты машинного обучения могут обучаться и совершенствоваться.

Внедрение цифровых двойников в производственные процессы также является ключевым фактором для улучшения эффективности и оптимизации операций предприятий. Они не только облегчают доступ к данным и сокращают затраты на обслуживание, но и позволяют предприятиям принимать обоснованные решения и управлять изменениями в рабочих процессах в режиме реального времени.

Таким образом, внедрение симуляторов ML-Agents и цифровых двойников представляет собой важный шаг в развитии современных производственных процессов и исследований в области машинного обучения, открывая новые возможности для оптимизации и инноваций в различных отраслях.

2 Создание виртуальной среды

2.1 Используемые библиотеки и модули

2.1.1 Модуль ML-Agents для Unity

ML-Agents - это пакет, разработанный Unity Technologies, который позволяет создавать и обучать агентов искусственного интеллекта в среде Unity. Этот пакет позволяет разработчикам использовать машинное обучение для создания умных агентов, которые могут взаимодействовать с игровым миром и другими объектами в Unity-сцене.

Функции:

– Интеграция среды Unity с библиотеками машинного обучения, такими как TensorFlow, для обучения агентов в среде реального времени,

– Создание и обучение агентов для выполнения различных задач в игровой среде, таких как навигация, управление объектами, соревнования и т.д,

– Использование различных алгоритмов машинного обучения, включая обучение с подкреплением и глубокое обучение, для обучения агентов,

– Встроенные инструменты для создания и настройки среды, агентов и обучения,

– Поддержка обучения на CPU и GPU,

– Возможность масштабирования обучения для многопользовательских сценариев.

ML-Agents интегрирован с Unity и предоставляет средства для создания и обучения агентов непосредственно в редакторе Unity. Разработчики могут определить агентов, среду и алгоритмы обучения прямо в Unity Editor, а затем

запустить обучение, чтобы агенты могли начать учиться взаимодействовать со средой и другими объектами в сцене Unity.

2.1.2 mlagents-learn

mlagents-learn – это библиотека машинного обучения для python, разработанная Unity Technologies совместно с пакетом ML-Agents для Unity, которая позволяет разработчикам создавать и обучать агентов искусственного интеллекта в среде Unity.

Функции:

– Обучение агентов в Unity-среде с использованием алгоритмов обучения с подкреплением, включая глубокое обучение,

– Интеграция с средой Unity для создания сред обучения и управления взаимодействием агентов с ними,

– Возможность обучения агентов для выполнения различных задач, таких как управление объектами в виртуальном мире, прохождение уровней, соревнования и т.д,

– Поддержка обучения на CPU и GPU,

– Масштабируемость для обучения агентов в многопользовательской среде.

mlagents-learn интегрирован с Unity и предоставляет средства для создания и обучения агентов прямо в редакторе Unity. Он предоставляет набор инструментов и API для определения среды, агентов и обучения, а также возможности для настройки и запуска обучения внутри Unity Editor.

2.1.3 TensorFlow

TensorFlow – это открытая библиотека машинного обучения, разработанная Google, которая предоставляет инструменты для создания и обучения различных моделей машинного обучения, включая нейронные сети.

Функции:

– Создание и обучение различных типов моделей машинного обучения, включая нейронные сети, рекуррентные сети, сверточные нейронные сети и т.д,

– Мощные инструменты для предобработки и анализа данных.

– Использование графовых вычислений для оптимизации производительности и эффективного использования ресурсов,

– Поддержка обучения на CPU и GPU,

– Широкий выбор предварительно обученных моделей и алгоритмов, доступных в TensorFlow Hub,

– Интеграция с другими библиотеками и инструментами машинного обучения,

– Возможность развертывания обученных моделей с помощью TensorFlow Serving.

TensorFlow предоставляет удобный интерфейс для создания и обучения моделей машинного обучения с использованием высокоуровневых API, таких как Keras, а также низкоуровневых API для более гибкого управления моделями и вычислениями. Он поддерживает создание и обучение моделей как на локальной машине, так и в облачных средах, а также предоставляет мощные инструменты для масштабирования обучения на крупных наборах данных. TensorFlow используется во многих областях, включая компьютерное зрение, обработку естественного языка, обучение с подкреплением, анализ данных и т.д.

2.1.4 Tensorboard

TensorBoard - это инструмент визуализации и мониторинга, разработанный Google для анализа и отладки моделей машинного обучения, созданных с использованием TensorFlow.

Функции:

– Визуализация графа вычислений модели,

– Мониторинг метрик обучения и проверки (например, потери, точность),

– Анализ распределений весов модели и гистограмм активаций,

– Отслеживание изменений во времени, таких как обучение и оценка,

– Визуализация изображений, векторов, текста и других данных.

TensorBoard интегрирован с TensorFlow и запускается из командной строки. Он сохраняет данные мониторинга во время обучения в специальных журнальных файлах, которые потом можно анализировать и визуализировать с помощью TensorBoard.

2.1.5 Pytorch

PyTorch - это фреймворк машинного обучения с открытым исходным кодом, который предоставляет гибкие инструменты для создания и обучения глубоких нейронных сетей. Вот несколько ключевых функций и возможностей PyTorch:

а) Динамический граф вычислений. PyTorch использует динамический граф вычислений, что означает, что граф формируется на лету при выполнении кода. Это облегчает отладку и экспериментирование с моделями, а также позволяет использовать стандартные языковые конструкции, такие как циклы и условные операторы;

б) Нейронные сети и оптимизаторы. PyTorch предоставляет широкий спектр предопределенных моделей глубокого обучения, таких как сверточные нейронные сети (CNN), рекуррентные нейронные сети (RNN) и трансформеры (Transformers), а также различные оптимизаторы для обучения моделей, такие как SGD, Adam, RMSprop и другие;

в) Автоматическое дифференцирование. PyTorch автоматически вычисляет градиенты для параметров модели с помощью алгоритма обратного распространения ошибки (backpropagation). Это позволяет удобно оптимизировать параметры модели в процессе обучения;

г) GPU ускорение. PyTorch позволяет выполнять вычисления на графических процессорах (GPU), что значительно ускоряет обучение глубоких нейронных сетей. Он обеспечивает простой и удобный интерфейс для работы с GPU и автоматическое перемещение данных между CPU и GPU;

д) Модули для обработки данных. PyTorch предоставляет набор модулей и утилит для работы с данными, таких как DataLoader для загрузки данных в модель в пакетах, transforms для предварительной обработки данных, и различные наборы данных и загрузчики для работы с различными типами данных;

е) Расширяемость и гибкость. PyTorch предоставляет гибкие инструменты для создания и расширения пользовательских моделей и функций. Он активно использует концепцию динамического графа и поддерживает различные абстракции для работы с данными и моделями;

ж) Интеграция с другими библиотеками. PyTorch хорошо интегрируется с другими библиотеками и фреймворками машинного обучения, такими как NumPy, SciPy и scikit-learn, что позволяет использовать их функции и методы вместе с PyTorch.

PyTorch является мощным и гибким инструментом для создания и обучения глубоких нейронных сетей и широко используется в академических и промышленных исследованиях, а также в разработке приложений и продуктов машинного обучения.

2.2 Модель робота-манипулятора

В качестве объекта, действия которого контролирует агент ML-Agent, была выбрана модель робота-манипулятора из открытых источников (Рисунок 2).

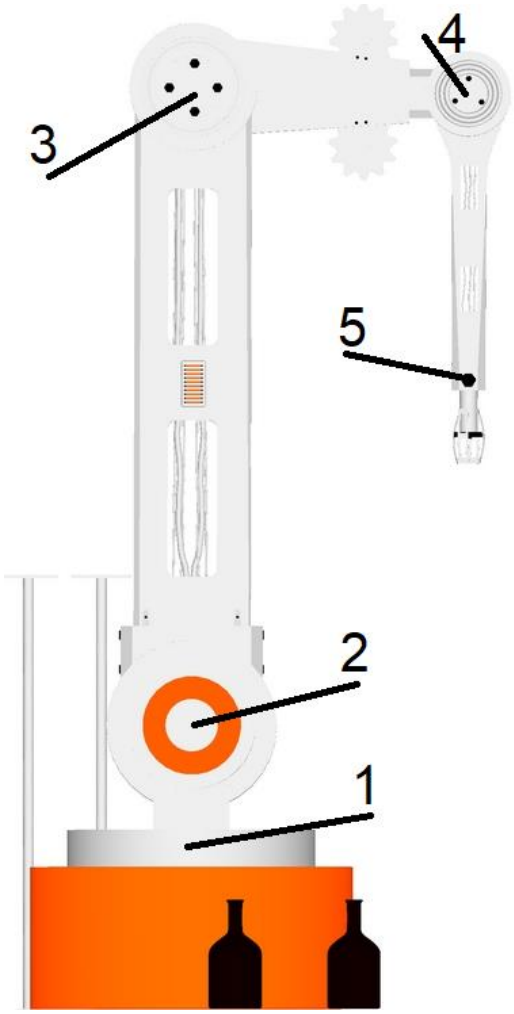

Рисунок 2. Изображение модели робота-манипулятора

Модель манипулятора обладает пятью осями вращения. Приводы 1 и 3 обеспечивают полный оборот на 360°, в то время как приводы 2, 4 и 5 ограничены вращением на 180°. Такая конфигурация позволяет манипулятору оперировать объектами в трехмерном пространстве в радиусе 360° по трем осям, обеспечивая широкий спектр возможных действий.

2.3 Создание среды для обучения агента

Обучающая среда для агента ML-Agent была создана в редакторе Unity версии 2023.1.20f. Цель обучения состоит в том, чтобы агент, управляющий действиями робота-манипулятора, научился достигать обозначенного объекта, основываясь лишь на информации о расстоянии до него и об углах поворота собственных подвижных частей. В обучающей среде агент начинает взаимодействовать с окружением, получая наблюдения и выбирая действия. Обучение происходит через итерации, где агент пытается максимизировать суммарную награду.

Модель робота-манипулятора была импортирована в Unity и была осуществлена возможность изменять поворот каждого из пяти приводов в ручном режиме. Было принято решение отдельно создать сцену для обучения агентов и отдельно для использования обученного агента.

На Рисунке 3 представлено схематичное изображение обучающей среды. Обучающая среда состоит из двух сцен Unity, конфигурационного файла, содержащего параметры обучения, интерфейса Python, из которого запускаются методы библиотек машинного обучения, и инструмента визуализации TensorBoard. Обе сцены содержат объекты Robot Arm – сам робот манипулятор, и Target Object – объект, достижение которого является задачей агента. Компонент Transform отвечает за координаты объекта на сцене, Mesh Collider – за обработку столкновений с другими объектами, Axis 1..5 обозначает оси вращения подвижных частей робота-манипулятора. Компоненты BehaviourParameters.cs, DecisionRequester.cs и RobotArmAgent.cs являются скриптами на языке C#, от которых зависит процесс обучения. RobotArmAgent.cs описывает поведение агента, собираемые им наблюдения, результат совершенных им действий, его вознаграждения и штрафы. DecisionRequester.cs -–необходимый компонент Ml-Agents, требующий от агента совершать действия. BehaviourParameters.cs описывает величину векторов наблюдений и действий, а также позволяет более тонко настраивать параметры обучения, например – выбор для обучения CPU или GPU. Отличие сцен заключается в том, что на обучающей сцене размещено 12 копий

объектов и на ней не используется обученная модель, вместо этого к ней подключается интерфейс Python, позволяющий проводить обучения агента [25-41].

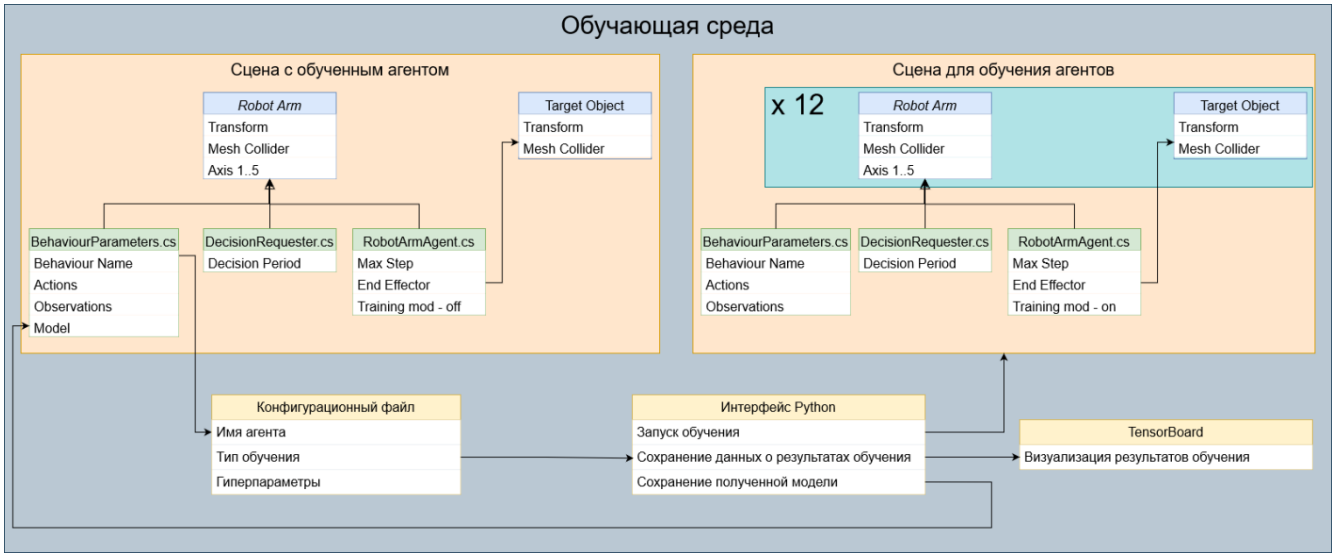

Рисунок 3. Схема обучающей среды

2.3.1 Сцена для обучения агента

На обучающей сцене было размещено 12 копий робота-манипулятора вместе с объектами, которые они должны достичь. Каждый объект, представляющий робота-манипулятора, обладает компонентом RobotArmAgent, который отвечает за взаимодействие с агентом ML-Agent. Все копии имеют одинаковую конфигурацию параметров поведения (Рисунок 4).

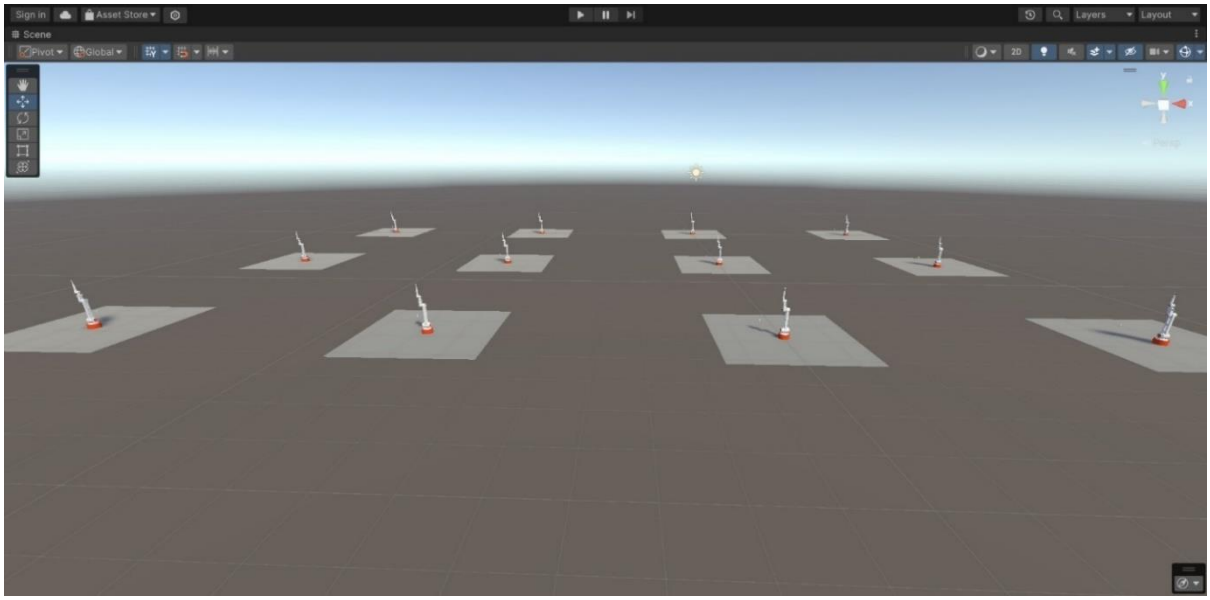

Рисунок 4. Сцена, используемая при обучении

2.3.2 Сцена с обученным агентом

Помимо сцены, на которой агент обучается, также была создана сцена для проверки результатов обучения. На ней расположена лишь одна копия робота-манипулятора, содержащая компонент, отвечающий за принятие действий, который был получен в результате обучения (Рисунок 5).

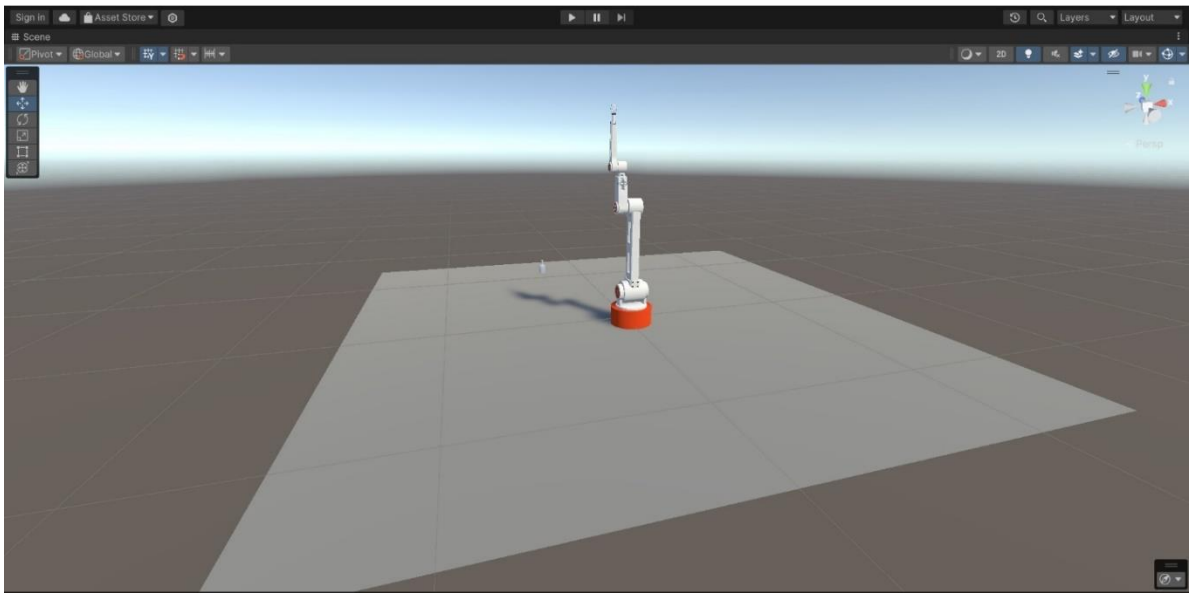

Рисунок 5. Сцена, на которой размещен обученный агент

2.4 Наблюдения и действия агента

В основе метода ML-Agent лежат наблюдения (Observation) и действия (Action) агента. Под наблюдением понимается информация о текущем состоянии того или иного компонента какого-либо объекта на сцене в произвольный момент времени. Действием же является либо дискретное число, принимающее значения в заданном диапазоне, либо число с плавающей точкой, принимающее значение от 0 до 1, которое агент может изменять по своему усмотрению.

2.4.1 Наблюдения

При разработке агента особое значение имеет определение количества наблюдений, на основе которых агент может принимать решения и совершать действия. Наблюдения используются для формирования входных данных для нейронной сети агента. В созданном агенте установлено 9 наблюдений: это три координаты манипулятора, пять углов поворота его осей и расстояние до целевого объекта.

2.4.2 Действия

Агент обучается выбирать действия на основе его текущего состояния и целей. Для того, чтобы агент мог эффективно управлять всеми подвижными частями робота-манипулятора, ему потребуется возможность совершать 5 действий, каждое из которых представлено в виде числа с плавающей точкой, принимающего значения от 0 до 1. В дальнейшем при помощи математических преобразований эти числа будут напрямую связаны с углами поворота робота-манипулятора.

2.5 Награда за обучение

Награда - это численная оценка, которую агент получает от среды за выполнение определенных действий. При этом в процессе обучения агент может как получать награду, так и терять ее. Цель агента состоит в максимизации суммарной награды за определенный период времени или количество действий.

В обучение с подкреплением существуют разные механизмы награды. Наиболее распространенные из них приведены ниже:

а) Intrinsic Reward (Внутренняя награда). Внутренняя награда представляет собой механизм награждения агента за выполнение определенных внутренних целей или достижение промежуточных состояний. Например, в игре агент может быть награжден за исследование окружающей среды или достижение новых уровней;

б) Auxiliary Reward (Дополнительная награда). Дополнительная награда представляет собой дополнительный механизм награждения, который используется вместе с внешней или внутренней наградой для улучшения обучения агента. Например, она может быть использована для награждения агента за выполнение определенных подзадач, которые помогают достичь основной цели;

в) Curiosity-driven Reward (Награда, основанная на любопытстве). Этот метод награждения стимулирует агента к исследованию и обучению на основе своих собственных прогнозов или оценок о том, как изменения в среде влияют на его состояние и окружение. Например, агент может получать награду за достижение новых состояний или предсказание последствий своих действий;

г) Sparse Reward (Разреженная награда). Разреженная награда представляет собой механизм награждения, который присваивается агенту только за выполнение определенных ключевых действий или достижение

целей. Например, агент может получать награду только за прохождение уровня в игре или выполнение определенной задачи;

д) Shaped Reward (Формированная награда). Формированная награда представляет собой награду, которая формируется на основе знания о задаче или среде. Например, она может быть разработана для награждения агента за приближение к желаемому поведению или за успешное выполнение сложных задач;

е) Extrinsic Reward (Внешняя награда). Внешняя награда - это основной механизм обратной связи, который агент получает от среды в процессе обучения с подкреплением. Эта награда непосредственно указывает на выполнение или не выполнение агентом поставленной задачи и обычно предоставляется разработчиком среды.

В данной работе было принято решение назначить положительную награду за достижение цели и отрицательную за столкновение с препятствиями. Также назначен небольшой штраф за каждый шаг симуляции, что создает для агента стимул достичь цели в максимально короткий срок, а также назначается небольшая награда, если агент в текущий момент времени находится ближе к своей цели, чем в предыдущий. Таким образом, основной механизм награды – это внешняя награда, а также применима отрицательная дополнительная награда в случае столкновения с препятствиями [42-51].

2.6 Тип обучения

При обучении агента настраивается ряд гиперпараметров, влияющих на процесс обучения. Перед определением гиперпараметров необходимо выбрать тип обучения. Краткое описание распространенных типов обучения приведено ниже:

а) SAC (Soft Actor-Critic). SAC является алгоритмом обучения с подкреплением, который работает с непрерывными пространствами действий и обычно хорошо подходит для задач с непрерывным управлением, такими как

управление роботами или управление агентами в симуляциях физических сред;

б) DDPG (Deep Deterministic Policy Gradient). DDPG также работает с непрерывными пространствами действий и является более простым в реализации, чем SAC. Он часто используется в задачах с непрерывным управлением;

в) TD3 (Twin Delayed DDPG). TD3 является улучшением DDPG и старается решить некоторые из его недостатков, такие как смещение оценки и шум в обновлении критика;

г) DDQN (Double Deep Q-Network). DDQN является вариантом алгоритма Q-learning, который использует две независимые нейронные сети для оценки ценности действий. Он обычно используется для задач с дискретными пространствами действий, такими как управление в компьютерных играх;

д) DQN (Deep Q-Network). DQN является одним из первых алгоритмов, успешно примененных для обучения с подкреплением в компьютерных играх. Он основан на методе Q-learning и использует нейронную сеть для оценки ценности действий в каждом состоянии;

е) PPO (Proximal Policy Optimization). PPO является алгоритмом обучения с подкреплением, разработанным для эффективного обучения стратегий управления агентами в задачах с дискретными или непрерывными пространствами действий. Он является итеративным алгоритмом, который обновляет параметры стратегии (политики) агента таким образом, чтобы изменения были незначительными, чтобы избежать резких изменений, которые могут привести к нестабильности в процессе обучения.

Как можно видеть, теоретически для задачи обучения робота-манипулятора подойдут варианты 1, 2, 3, и 6, так как все его действия совершаются в непрерывном пространстве. Было принято решение реализовать алгоритм PPO, так как он обладает следующими преимуществами относительно других типов обучения:

а) Стабильность обучения. PPO разработан с учетом стабильности обучения. Ограничение расстояния между старой и новой стратегиями помогает избежать резких изменений и нестабильности в процессе обучения. Это особенно важно в задачах управления роботом, где даже небольшие изменения стратегии могут привести к серьезным последствиям;

б) Сходимость и эффективность. PPO обеспечивает хорошую сходимость и эффективность обучения. Он может использовать данные собранные от нескольких агентов параллельно для обновления параметров стратегии, что ускоряет процесс обучения и повышает его эффективность;

в) Работа с непрерывными пространствами действий. В задачах управления роботом, где пространство действий может быть непрерывным и многомерным, PPO является более предпочтительным выбором, так как он хорошо работает с непрерывными пространствами действий;

г) Масштабируемость. PPO может быть легко масштабирован для работы с различными типами роботов и задачами управления. Он может быть адаптирован для различных размеров нейронных сетей и типов наблюдений, что делает его гибким в применении к различным сценариям;

д) Регуляризация и контроль изменений. PPO включает в себя механизмы регуляризации, такие как ограничение расстояния между старой и новой стратегиями, что помогает контролировать изменения стратегии и избегать переобучения.

Помимо этого, метод PPO обладает следующими ключевыми особенностями:

а) Проксимальность (Proximal). Одна из основных концепций PPO - это обеспечение проксимальности изменений стратегии. Это достигается путем ограничения расстояния (например, KL-дивергенции) между старой и новой стратегиями, чтобы изменения оставались небольшими. Это позволяет избежать нестабильности и ускоряет сходимость алгоритма;

б) Пакетное обучение (Batch Training). PPO использует пакетное обучение, где данные собираются в течение нескольких шагов среды, и потом

используются для нескольких обновлений параметров стратегии. Это помогает улучшить стабильность и эффективность обучения;

в) Многопоточность (Parallelism). PPO может эффективно использовать многопоточность для ускорения процесса обучения. Это достигается путем запуска нескольких агентов параллельно, которые собирают данные из среды и обновляют параметры стратегии независимо друг от друга;

г) Функция потерь (Loss Function). Функция потерь PPO включает в себя несколько компонентов, таких как функция потерь политики (policy loss) и функция потерь оценки (value loss). Эти компоненты обеспечивают стабильное и точное обучение;

д) Гиперпараметры (Hyperparameters). PPO имеет несколько гиперпараметров, таких как размер пакета (batch size), скорость обучения (learning rate), коэффициенты регуляризации и другие.

2.7 Конфигурация гиперпараметров обучения

От выбранных гиперпараметров зависит скорость и стабильность обучения и поведения агента. Ниже приведены некоторые из ключевых гиперпараметров PPO:

а) batch_size. Размер пакета (batch size), используемый при обновлении параметров стратегии агента. Этот параметр определяет количество обучающих примеров, которые используются для каждого обновления;

б) buffer_size. Размер буфера воспроизведения (replay buffer), который хранит предыдущие состояния и действия для использования при обучении. Больший размер буфера может помочь алгоритму лучше запоминать прошлые наблюдения и сделать обучение более стабильным;

в) learning_rate_schedule. Расписание изменения скорости обучения (learning rate) в течение процесса обучения. Это может быть, например, линейное изменение или экспоненциальное затухание скорости обучения;

г) learning_rate. Начальная скорость обучения, используемая в процессе обновления параметров стратегии. Это определяет, насколько большие шаги делаются в направлении градиента;

д) hidden_units. Количество скрытых нейронов в каждом слое нейронной сети, используемой для представления стратегии агента. Это влияет на сложность модели и ее способность к обучению сложным задачам;

е) num_layers. Количество слоев в нейронной сети. Более глубокая сеть может захватывать более сложные зависимости, но также может быть более сложной в обучении и более склонной к переобучению;

ж) gamma. Дисконтный фактор, который определяет важность будущих наград по сравнению с текущими. Этот параметр влияет на то, насколько агент учитывает будущие вознаграждения при принятии решений;

з) entropy_coefficient. Коэффициент энтропии, который регулирует уровень случайности в действиях агента. Большее значение этого параметра может вызвать более хаотичное поведение агента;

и) clip_param. Параметр ограничения (clipping parameter), который определяет максимальное изменение стратегии за одно обновление. Это помогает контролировать размер обновлений и улучшает стабильность обучения.

Оптимальные значения гиперпараметров подбираются в контексте конкретной задачи и зачастую требуют экспериментов и тщательной настройки для достижения лучших результатов.

На Рисунке 6 приведены данные конфигурационного файла, содержащего гиперпараметры для обучения агента, в формате .json.

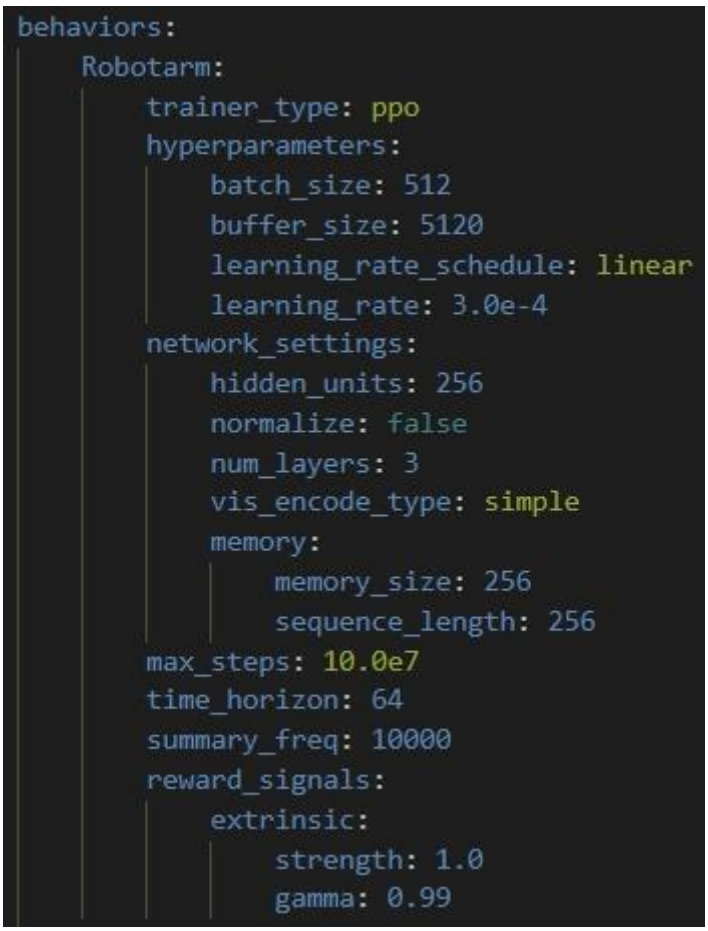

Рисунок 6. Гиперпараметры, используемые при обучении агента

Далее приведено пояснение выбора гиперпараметров:

а) Behaviors. Это ключевое слово, указывающее на начало блока определения поведения агента;

б) Robotarm. Это название поведения, в данном случае, это поведение агента робота-манипулятора. В Unity ML-Agents можно определить несколько различных поведений, каждое со своими собственными параметрами обучения;

в) Trainer_type: ppo. Это указывает на то, что для данного поведения будет использоваться алгоритм обучения PPO;

г) Hyperparameters. Этот блок содержит гиперпараметры для алгоритма PPO. Гиперпараметры — это параметры обучения, которые не обучаются моделью напрямую, но влияют на ее процесс обучения. В данном случае, устанавливаются следующие гиперпараметры:

- batch_size: 512. Размер пакета (batch), используемый при обновлении весов модели;
- buffer_size: 5120. Размер буфера воспроизведения (replay buffer), который хранит предыдущие состояния и действия для обучения;
- learning_rate_schedule: linear. Расписание изменения скорости обучения (learning rate);
- learning_rate: 3.0e-4. Начальная скорость обучения для алгоритма PPO.

д) network_settings. Этот блок содержит настройки нейронной сети, используемой для обучения агента:

- hidden_units: 256. Количество скрытых нейронов в каждом слое нейронной сети;
- normalize: false. Флаг, указывающий, следует ли нормализовать входные данные;
- num_layers: 3. Количество слоев в нейронной сети;
- vis_encode_type: simple. Тип кодирования визуальных наблюдений;
- memory. Настройки для памяти агента;
- memory_size: 256. Размер памяти;
- sequence_length: 256. Длина последовательности в памяти.

е) max_steps: 10.0e7. Максимальное количество шагов обучения, которое будет выполнено перед завершением процесса обучения;

ж) time_horizon: 64. Интервал времени, на котором агент учитывает будущие награды при принятии решений;

з) summary_freq: 10000. Частота записи отчетов о процессе обучения (например, значения функции потерь) в общий отчет;

и) reward_signals. Этот блок содержит определение вознаграждений (rewards), которые агент будет получать от среды:

– extrinsic. Внешнее вознаграждение, являющееся обратной связью, которую агент получает от среды в ответ на свои действия;
– strength: 1.0. Величина вознаграждения;
– gamma: 0.99. Фактор, который определяет важность будущих наград по сравнению с текущими.

2.8 Запуск обучения

Для инициализации процесса обучения необходимо из командной строки запустить модуль Python mlagents-learn с указанием пути к конфигурационному файлу. После инициализации он переходит в режим ожидания подключения (по умолчанию порт 5004), и в этот момент необходимо запустить обучающую сцену в редакторе Unity. При успешном соединении в командной строке начнут отображаться промежуточные результаты обучения, включающие в себя прошедшее время и собранную в текущем эпизоде награду. На Рисунке 7 представлен пример, как в командной строке выглядит успешный запуск модуля mlagents-learn.

Рисунок 7. Соединение с Unity из командной строки

2.9 Вывод

В данной главе были рассмотрены основные этапы создания обучающей среды для агента ML-Agent в редакторе Unity, а также основные принципы определения наблюдений, действий и механизмов награды для обучения агента.

Были рассмотрены такие важные шаги, как создание среды, которая максимально точно отражает реальные условия и задачи, которые агент должен будет решать. Для этого в данной работе были созданы две сцены: одна для обучения агента, в которой размещено несколько копий робота-манипулятора и объектов, а другая для проверки результатов обучения с одной копией робота-манипулятора.

Определение наблюдений и действий для агента играет ключевую роль в процессе обучения. В данной работе агент получает информацию о координатах манипулятора, углах поворота его осей и расстоянии до целевого объекта в качестве наблюдений. Действия агента представлены в виде чисел с плавающей точкой, позволяющих ему эффективно управлять роботом-манипулятором.

Механизм награды играет решающую роль в формировании стратегии обучения агента. В данной работе принято решение назначить положительную награду за достижение цели и отрицательную за столкновение с препятствиями, а также применять небольшой штраф за каждый шаг симуляции. Это создает стимул для агента достигать цели в кратчайшие сроки.

Выбор типа обучения и настройка гиперпараметров также играют важную роль в процессе обучения агента. В данной работе был выбран метод PPO (Proximal Policy Optimization) из-за его стабильности, сходимости и способности эффективно работать с непрерывными пространствами действий. Оптимальные значения гиперпараметров подбираются в ходе экспериментов и тщательной настройки для достижения лучших результатов обучения.

3 Оценка результатов обучения

Оценка результатов обучения играет ключевую роль в процессе разработки и совершенствования алгоритмов обучения с подкреплением. Важность этого этапа трудно переоценить по нескольким причинам.

Во-первых, оценка обучения позволяет нам оценить эффективность алгоритма и понять, насколько успешно он выполняет поставленные задачи. Это важно для того, чтобы определить, стоит ли продолжать обучение или нужно внести изменения в алгоритм или стратегию агента.

Во-вторых, оценка результатов обучения помогает нам понять, какие аспекты обучения требуют дополнительного внимания и улучшения. Например, анализ кривой обучения может показать, что агент сходится к оптимальному решению слишком медленно, что может указывать на необходимость изменения гиперпараметров или алгоритма обучения.

В-третьих, оценка результатов обучения помогает нам сравнивать различные подходы и методы обучения. Это позволяет нам определить, какие из них работают лучше в конкретной среде или задаче, и выбрать наиболее подходящий подход для решения конкретной задачи.

Кроме того, оценка результатов обучения помогает нам лучше понять суть задачи и требования к ней. Это может привести к новым идеям и подходам к решению проблемы, а также к созданию новых методов и алгоритмов обучения.

3.1 Критерии оценки

При оценке результатов обучения агента ML-Agent может применяться один или несколько базовых методов оценки обучения из представленных ниже.

а) Кривая обучения (Learning Curve). Кривая обучения отображает изменение награды агента или его производительности на протяжении

обучения. Это позволяет оценить, насколько быстро агент достигает определенного уровня производительности и какие тенденции наблюдаются в процессе обучения.

б) Анализ поведения (Behavioral Analysis). Анализ поведения агента в среде позволяет оценить, насколько успешно он выполняет поставленные задачи и какие стратегии и тактики он использует для этого. Это может включать визуализацию действий агента, его перемещений и взаимодействий с окружающей средой.

в) Анализ эпизодов (Episode Analysis). Анализ эпизодов агента позволяет оценить его поведение в конкретных сценариях или условиях. Например, это может включать анализ успешности агента в выполнении конкретных задач или реакции на различные типы входных данных.

г) Оценка качества обучения (Training Quality Evaluation). Оценка качества обучения позволяет оценить, насколько эффективно агент использовал свой опыт для улучшения своей стратегии и производительности. Это может включать анализ стабильности обучения, скорости сходимости и других метрик производительности.

д) Тестирование на тестовых данных (Testing on Holdout Data). После завершения обучения агента можно протестировать его на отложенных данных или в реальной среде, чтобы оценить его способность к обобщению и производительность в реальных условиях.

е) Сравнение с базовыми методами (Baseline Comparison). Сравнение результатов агента с результатами базовых методов или других агентов помогает оценить его эффективность и превосходство по сравнению с альтернативными подходами.

При выполнении текущей задачи была визуализирована кривая обучения посредством библиотеки TensorBoard, также было визуально проанализировано поведение агента – это необходимо, чтобы удостовериться, что он в действительности обучается выполнять именно заданную задачу, так как даже при высоких показателях награды за обучения может получиться так,

что агент получает награду непредусмотренным образом, или же разработчик допустил ошибку при определении награды за обучение.

3.2 Визуализация в TensorBoard

С помощью TensorBoard можно строить графики изменения метрик производительности агента во времени, таких как средняя награда на эпизод, среднее значение функции потерь или другие метрики, отражающие прогресс обучения. Это позволяет оценить, насколько успешно агент учится и как быстро он достигает желаемого уровня производительности. Также TensorBoard позволяет сравнить различные стратегии обучения, отобразив их на одном и том же графике. В следующем разделе, описывающем итерации обучения, после каждой итерации приведены графики накопленной награды за обучения и графики длительности эпизода.

3.3 Стратегии обучения и возникшие проблемы

В процессе обучения агента были проведены эксперименты с разной конфигурацией гиперпараметров, наборов наблюдений и наград. Ниже приведено описание этапов оптимизации вышеуказанных составляющих с пояснением изменений на каждом этапе, а также графики накопленного вознаграждения и средней длины эпизода, визуализированные через TensorBoard.

1) При первом запуске агент ни разу не получил положительную награду. Это связано с тем, что изначально он не имел промежуточной награды за приближение к цели, а значит мог попытаться достичь ее лишь случайно. Вдобавок к этому, изначально время обучения было сильно ограничено, так что он не успел сделать это (Рисунок 8, Рисунок 9);

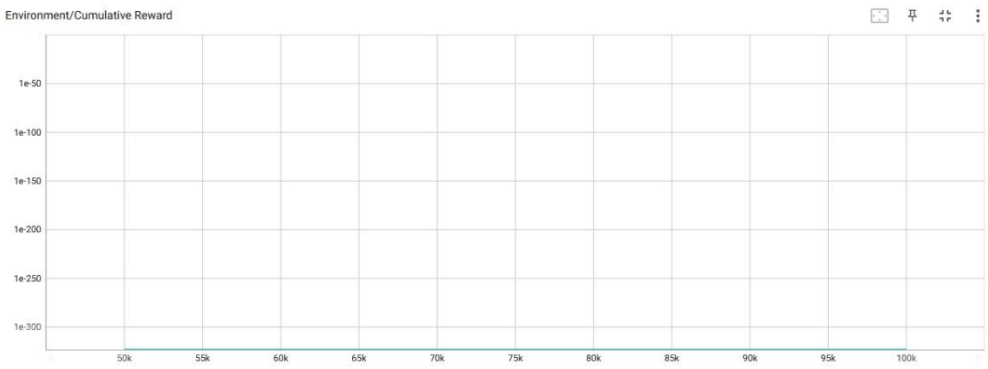

Рисунок 8. Кривая обучения итерации 1

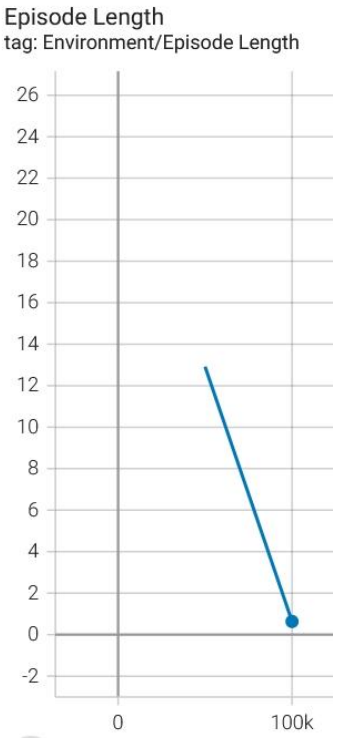

Рисунок 9. График зависимости длины эпизода от временного шага итерации 1

2) На втором этапе было увеличено допустимое время обучения и добавлено небольшое вознаграждение за сокращение расстояния до целевого объекта. Однако, как можно видеть на графике, довольно быстро наступило переобучение и агент перестал выполнять действия, направленные на получение награды. Причина была в том, что агент имел слишком большой штраф за касание препятствий, и принял решение полностью избегать задействования всех приводов, которые были бы способны заставить

манипулятор коснуться земли. На Рисунке 10 видно состояние «согнутой руки», таким образом манипулятор стремится минимизировать шанс столкновения с препятствием, но при этом он не всегда способен достигнуть целевого объекта (Рисунок 10, Рисунок 11, Рисунок 12);

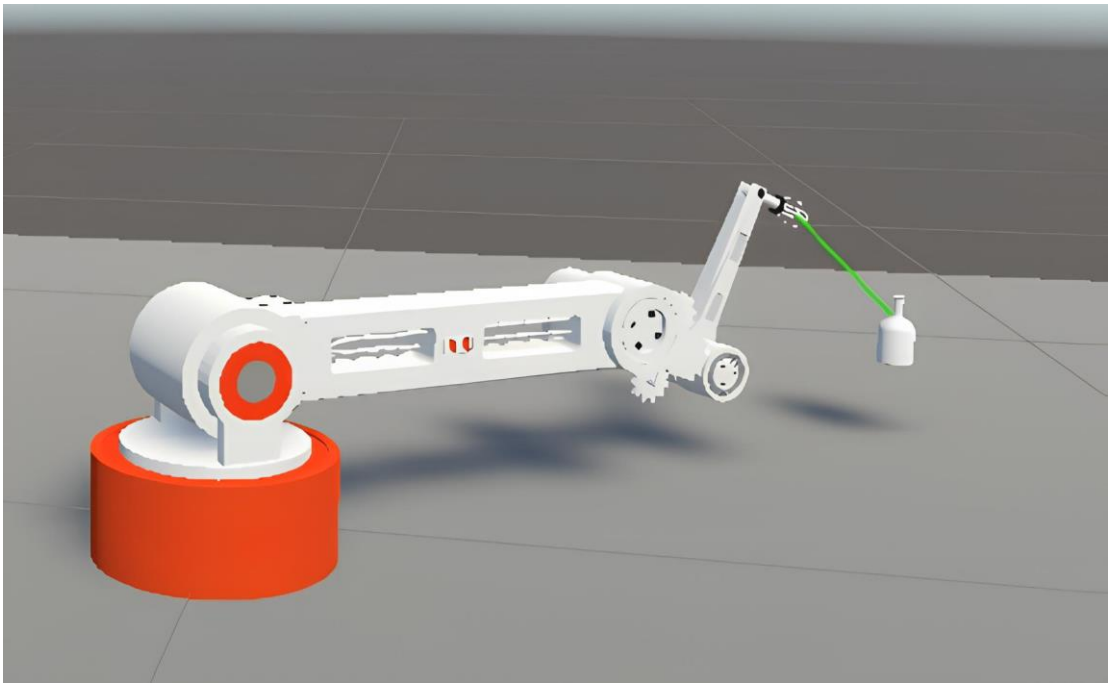

Рисунок 10. Состояние «согнутой руки».

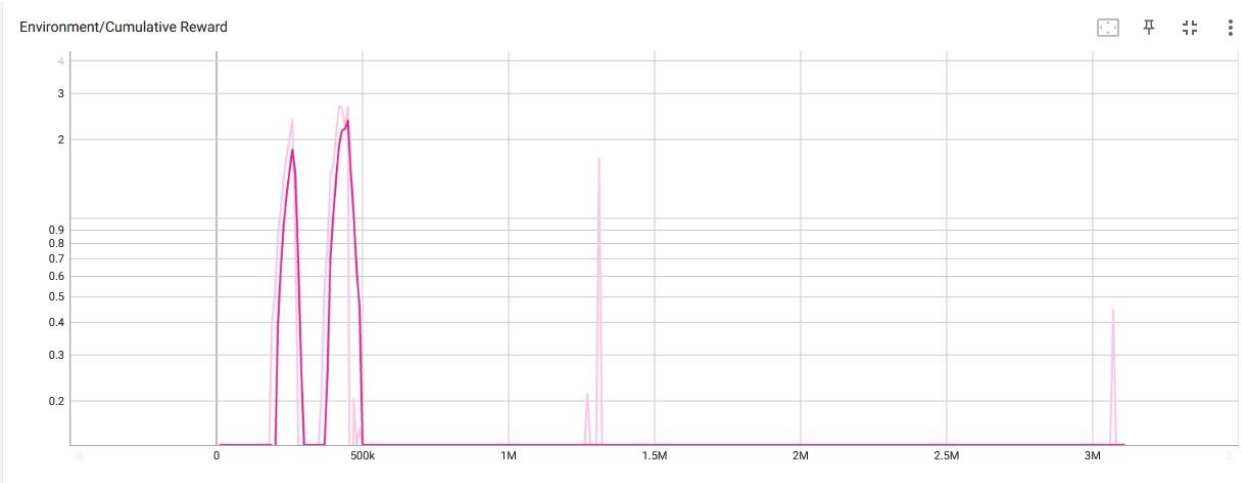

Рисунок 11. Кривая обучения итерации 2

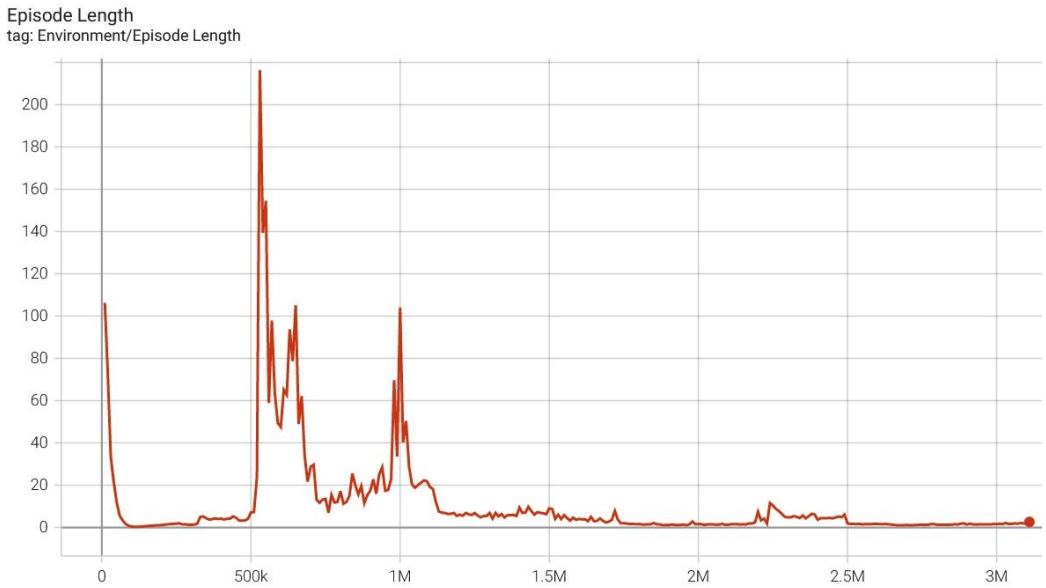

Рисунок 12. График зависимости длины эпизода от временного шага итерации 2

3) На этом этапе был уменьшен штраф за столкновение с препятствиями и впервые был достигнут стабильный рост награды за обучение, однако манипулятор все еще периодически совершал неоптимальные движения в попытке достичь целевой объект (Рисунок 13, Рисунок 14);

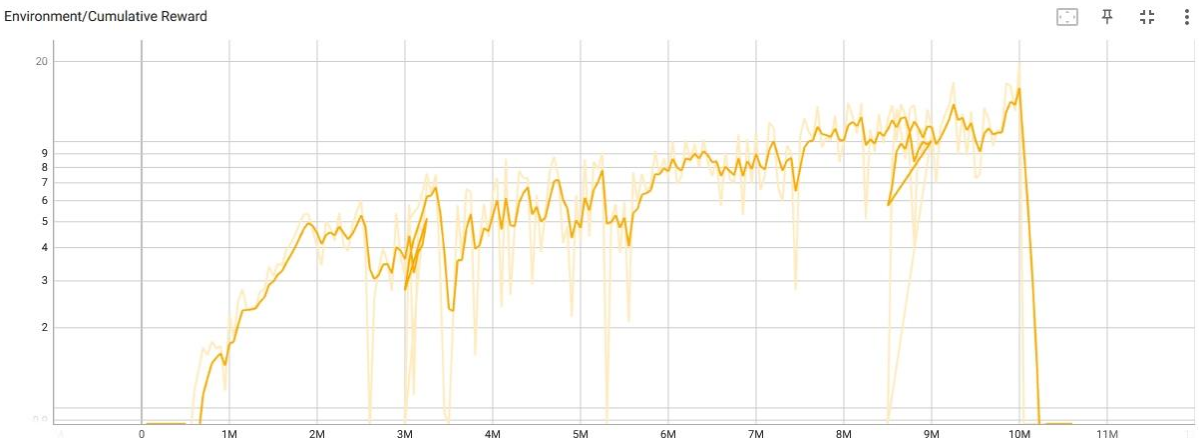

Рисунок 13. Кривая обучения итерации 3

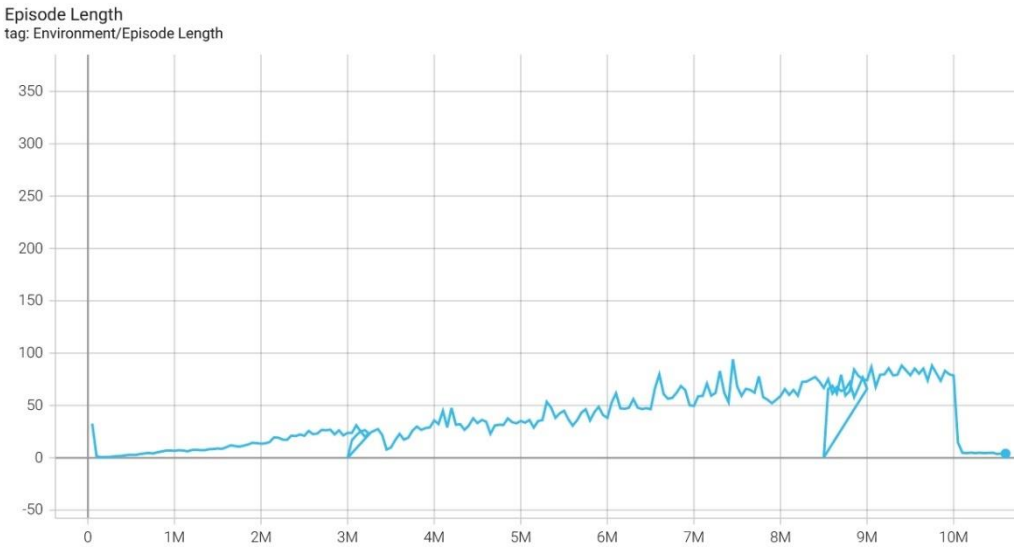

Рисунок 14. График зависимости длины эпизода от временного шага итерации 3

4) На данном этапе был проведен эксперимент по усложнению агента. Был добавлен дополнительный слой и увеличено количество скрытых нейронов в слоях нейронной сети с 256 до 512. Это привело к тому, что агент научился быстрее достигать награды в целом, но при этом процесс обучения стал менее стабильным – например, в некоторых эпизодах агент вовсе не был способен достичь награды, а в других беспрепятственно это делал (Рисунок 15, Рисунок 16);

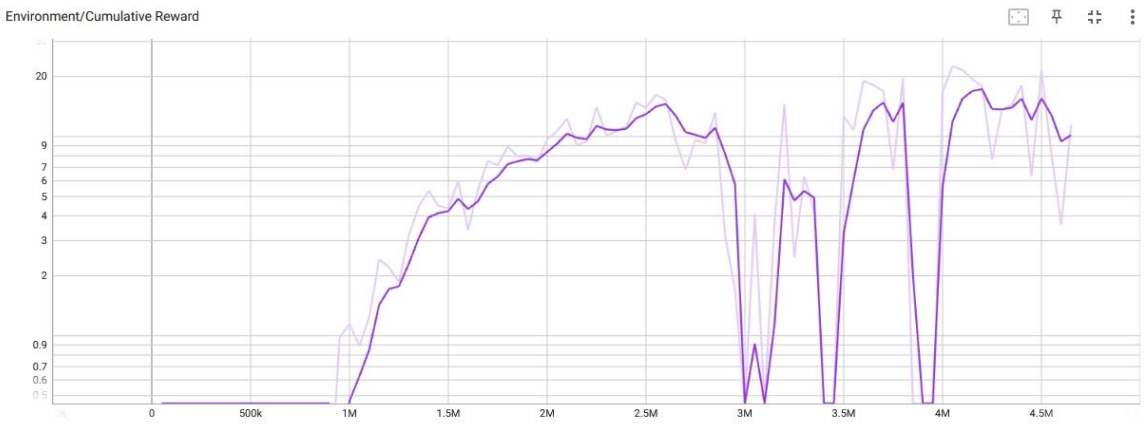

Рисунок 15. Кривая обучения итерации 4

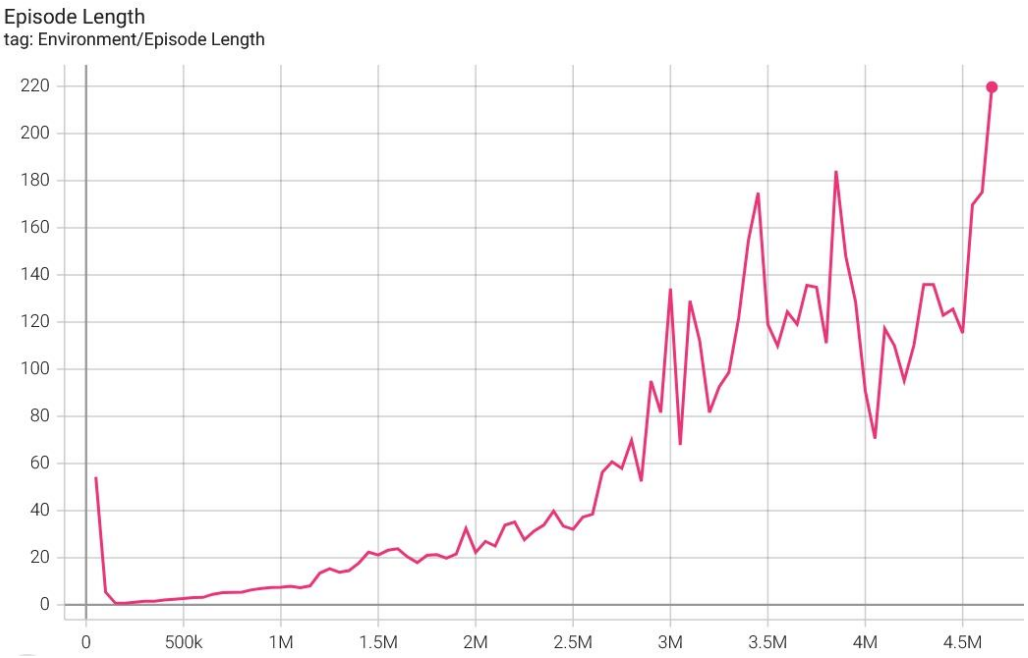

Рисунок 16. График зависимости длины эпизода от временного шага итерации 4

5) На этом этапе был проведен эксперимент с увеличением таких гиперпараметров как batch_size с 512 до 1024 и buffer_size с 5120 до 10240, а также было увеличено количество наблюдений, которые агент получает перед совершением действий. Это дало заметный прирост к накопленной в процессе обучения награде а также снизило частоту столкновения агента с препятствиями, однако в то же время агент начал тратить больше времени на принятие решений (Рисунок 17, Рисунок 18);

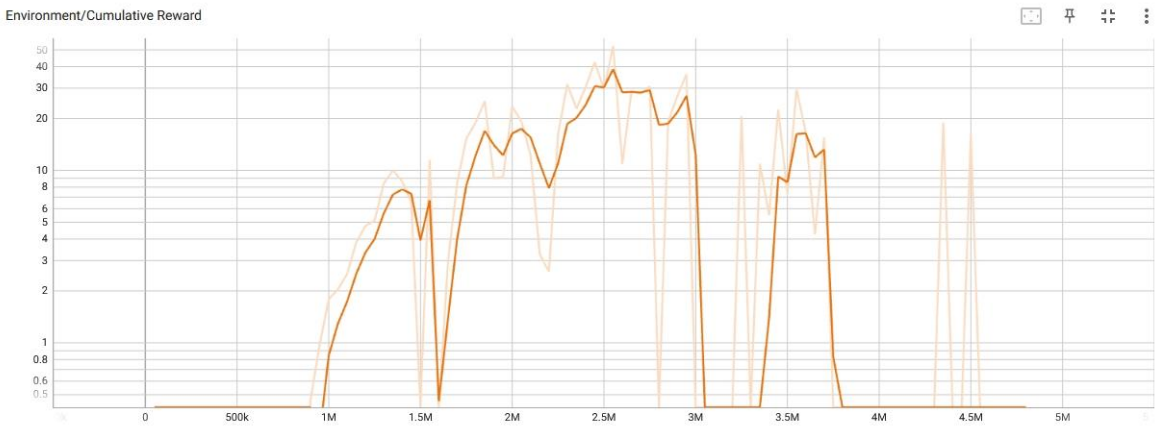

Рисунок 17. Кривая обучения итерации 5

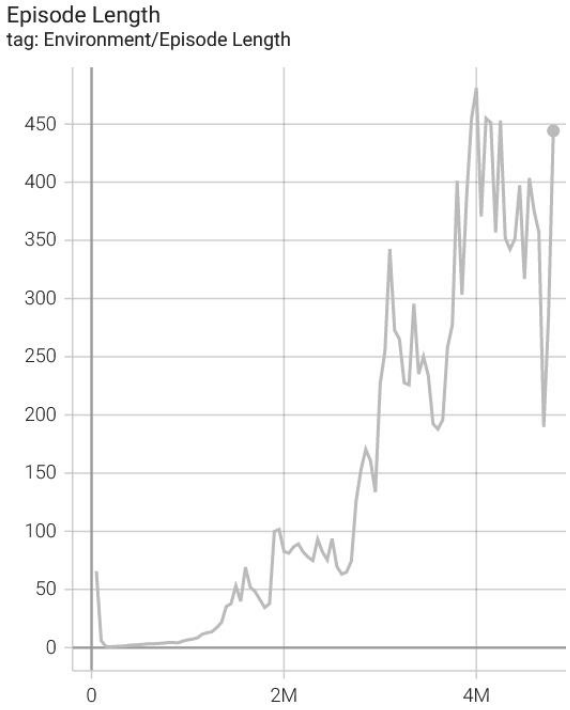

Рисунок 18. График зависимости длины эпизода от временного шага итерации 5

6) На этом этапе было принято решение продолжить эксперимент и еще больше усложнить модель, добавив еще один слой нейронной сети. Как и ранее, на некоторых эпизодах показатель награды возрос выше всех предыдущих значений. Было принято решение сохранить промежуточную версию одного из наиболее успешных эпизодов, так как она достаточно хорошо показала себя при проверке на сцене с агентом (Рисунок 19, Рисунок 20).

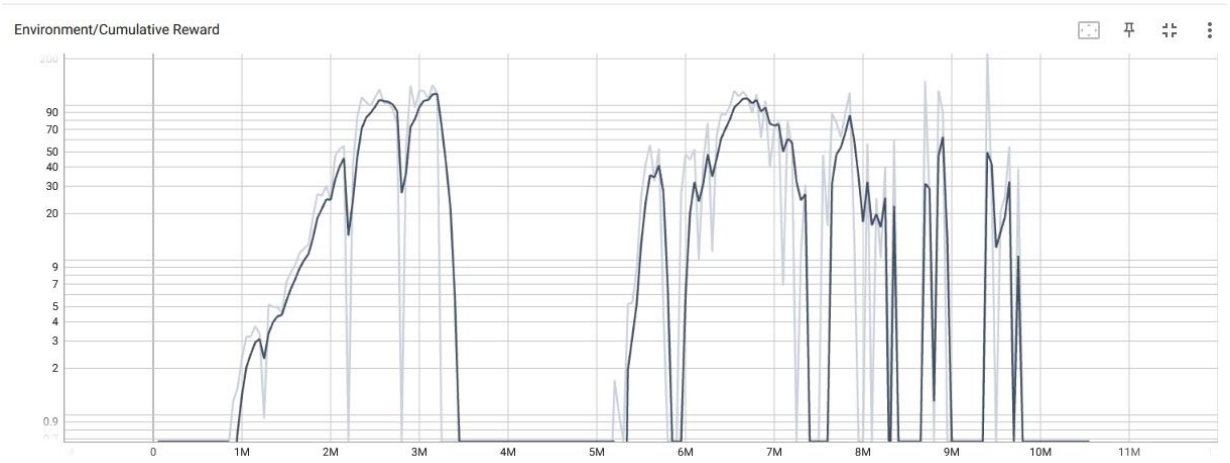

Рисунок 19. Кривая обучения итерации 6

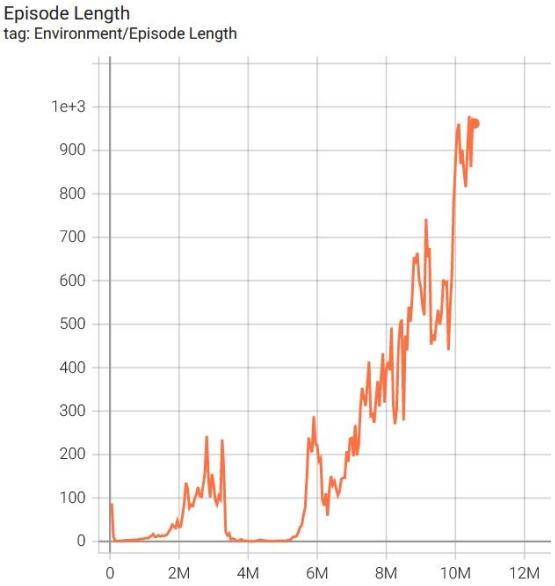

Рисунок 20. График зависимости длины эпизода от временного шага итерации 6

3.4 Оценка эпизодов обучения

В предыдущем разделе были приведены графики зависимости длины эпизода обучения от временного шага, и на всех итерациях наблюдалась тенденция к увеличению длительности эпизода. Увеличение длительности эпизода при обучении может являться как положительным, так и

отрицательным фактором, в зависимости от задач, наград, обучающей среды и прочих параметров обучения. В условиях управления роботом-манипулятором, эпизод заканчивается либо при достижении цели, либо при столкновении с препятствием. Следовательно, можно сделать вывод, что агент начинает проявлять «осторожность» в процессе обучения, то есть пытается избежать завершения эпизода со штрафом, что, соответственно приводит к увеличенной длине эпизода. Для того, чтобы убедиться, что увеличение продолжительности эпизода не является негативным фактором, было вычислено отношение успешных эпизодов к общему их количеству, состоящему из успешных и неудачных эпизодов. Успешным считается эпизод, в котором агент получил награду, а неудачным – тот, в котором он получил штраф за столкновение с препятствием. Результаты сведены в Таблицу 1.

Таблица 1. Доля успешных эпизодов

№ итерации	1	2	3	4	5	6
Доля успешных эпизодов	0	44%	21%	54%	90%	97%

3.5 Вывод

В данной главе была проведена оценка результатов обучения агента ML-Agent с использованием различных методов оценки и анализа.

Важность оценки результатов обучения заключается в том, что она позволяет оценить эффективность алгоритма и стратегии обучения, а также выявить проблемы и улучшить процесс обучения.

Для оценки результатов обучения были использованы кривая обучения, которая позволяет оценить динамику изменения награды или производительности агента в процессе обучения; анализ поведения, помогающий понять, какие стратегии и тактики использует агент для решения задачи и оценка эпизодов обучения, которая позволяет оценить длительность

эпизодов и их успешность, что важно для понимания процесса обучения и прогресса агента.

Эксперименты с различными конфигурациями гиперпараметров, наборами наблюдений и наградами позволили выявить оптимальные настройки для обучения агента. Были проведены итерации с изменением параметров, что помогло выявить проблемы и найти решения для их устранения.

В целом, оценка результатов обучения позволила получить представление о процессе обучения агента, его успешности и эффективности, а также выявить проблемы, которые требуют дополнительных исследований и улучшений.

ЗАКЛЮЧЕНИЕ

В ходе работы были рассмотрены различные аспекты обучения агента ML-Agent для управления роботом-манипулятором в среде Unity.

В первой главе были определены цели и задачи обучения, а также описана общая концепция работы агента в среде, включая наблюдения, действия и механизмы награды. Был проведен анализ симуляторов ML-Agent и методов их разработки, а также рассмотрены преимущества внедрения цифровых двойников в производственные процессы. Было подчеркнута важность использования разнообразных сред на платформе Unity ML-Agent для обучения с подкреплением и исследований в машинном обучении. Внедрение цифровых двойников способствует повышению эффективности и оптимизации операций, облегчает доступ к данным, снижает затраты на обслуживание и позволяет принимать обоснованные решения в реальном времени. Таким образом, был сделан вывод, что применение симуляторов ML-Agent может стать значительным шагом в развитии производственных процессов и машинного обучения, открывая новые возможности для оптимизации и инноваций.

Во второй главе рассмотрены основные этапы создания обучающей среды для агента ML-Agent в Unity, включая определение наблюдений, действий и механизмов награды. Созданы две сцены: одна для обучения с несколькими копиями робота-манипулятора и другая для тестирования результатов. Агент получает координаты и углы поворота манипулятора, а также расстояние до цели. Действия представлены числами с плавающей точкой для управления манипулятором. Механизм награды включает положительные награды за достижение целей, отрицательные за столкновения и штрафы за каждый шаг. Для обучения был выбран метод PPO из-за его стабильности и эффективности в непрерывных пространствах действий, а гиперпараметры тщательно настроены для достижения наилучших результатов.

В третьей главе была проведена оценка результатов обучения агента ML-Agent с использованием различных методов, таких как кривая обучения, анализ поведения и оценка эпизодов обучения. Эти методы позволили оценить эффективность алгоритма и стратегии обучения, выявить проблемы и оптимизировать процесс обучения. Эксперименты с различными конфигурациями гиперпараметров и сигналами награды помогли определить оптимальные настройки. Оценка результатов дала представление о прогрессе и эффективности обучения агента, а также позволила выявить области, требующие улучшений.

Общий вывод по всем главам заключается в следующем:

- Работа представляет собой комплексный подход к обучению агента ML-Agent для управления роботом-манипулятором;
- Создана среда для обучения агента, включая разработку сцен, определение наблюдений и действий, выбор типа обучения и настройку гиперпараметров;
- Проведена оценка результатов обучения, включая анализ кривой обучения, поведения агента, эпизодов обучения и возникших проблем;
- Полученные результаты позволяют сделать вывод о эффективности подхода и определить дальнейшие направления исследований и улучшений.

Таким образом, выполненная работа представляет собой важный шаг в исследовании обучения агентов ML-Agent в среде Unity и может быть использована в дальнейших исследованиях и практических приложениях в области управления робототехникой. Стоит отметить, что данное исследование можно развивать в разных направлениях.

Существует возможность провести исследование с использованием других моделей манипуляторов, стратегий обучения, таких как A3C (Asynchronous Advantage Actor-Critic), DDPG (Deep Deterministic Policy Gradient) или SAC (Soft Actor-Critic), а также с различными сигналами наград и штрафов.

На основе созданной в данной работе среды можно разрабатывать более продвинутые сценарии, где агенты будут решать сложные задачи, такие как перемещение предметов с обходом препятствий или синхронное перемещение нескольких предметов с минимальными временными затратами. Также можно расширить функциональность среды для обучения агентов, взаимодействующих между собой, что позволит исследовать кооперативное и поведение в мультиагентных системах.

Кроме того, дальнейшие исследования могут включать интеграцию сенсоров и систем компьютерного зрения для более реалистичного восприятия среды агентами. Это позволит моделировать более сложные и реалистичные задачи, такие как сборка объектов, сортировка и упаковка в производственных процессах.

Дополнительно, возможна оптимизация вычислительных ресурсов и ускорение процесса обучения с помощью распределенных вычислений и параллельного обучения агентов. Исследования в этом направлении могут привести к созданию более эффективных и масштабируемых систем для управления робототехникой.

В совокупности, эти направления открывают широкие возможности для дальнейшего развития и применения методик обучения агентов ML-Agent в различных областях, включая промышленную автоматизацию, логистику, медицину и другие сферы, требующие интеллектуального управления робототехническими системами.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Volodymyr Mnih. Human-level control through deep reinforcement learning / Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. // Nature, 2015. ISSN 00280836. URL: http://dx.doi.org/10.1038/nature14236 (дата обращения 12.03.2024).

2. Marc G. Bellemare. The arcade learning environment: An evaluation platform for general agents / Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling // CoRR, abs/1207.4708, 2012. URL: http://arxiv.org/abs/1207.4708 (дата обращения 12.03.2024).

3. Alex Nichol. Gotta learn fast: A new benchmark for generalization in rl / Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, John Schulman // arXiv preprint arXiv:1804.03720, 2018.

4. OpenAI. Universe, Dec 2016. сайт: URL: https://openai.com/blog/universe/ (дата обращения 12.03.2024).

5. Karl Cobbe. Leveraging procedural generation to benchmark reinforcement learning / Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman // arXiv preprint arXiv:1912.01588, 2019.

6. Bowen Baker. Emergent tool use from multi-agent autocurricula / Bowen Baker, Ingmar Kanitscheider, Todor M. Markov et al. // CoRR, abs/1909.07528, 2019. сайт: URL: http://arxiv.org/abs/1909.07528 (дата обращения 12.03.2024).

7. Ryan Lowe. Multi-agent actor-critic for mixed cooperative-competitive environments / Ryan Lowe, Yi Wu, Aviv Tamar et al. // CoRR, abs/1706.02275, 2017. сайт: URL: http://arxiv.org/abs/1706.02275 (дата обращения 12.03.2024).

8. Igor Mordatch. Emergence of grounded compositional language in multi-agent populations / Igor Mordatch, Pieter Abbeel // CoRR, abs/1703.04908, 2017. сайт: URL: http://arxiv.org/abs/1703 (дата обращения 12.03.2024).

9. Charles Beattie. Deepmind lab / Charles Beattie, Joel Z. Leibo, Denis Teplyashin et al. // CoRR, abs/1612.03801, 2016. сайт: URL: http://arxiv.org/abs/1612.03801 (дата обращения 27.03.2024).

10. Yuval Tassa. Deepmind control suite / Yuval Tassa, Yotam Doron, Alistair Muldal // CoRR, abs/1801.00690, 2018. сайт: URL: http://arxiv.org/abs/1801.00690 (дата обращения 27.03.2024).

11. Arthur Juliani. Unity: A general platform for intelligent agents / Arthur Juliani, Vincent-Pierre Berges, Esh Vckay et al. // CoRR, abs/1809.02627, 2018. сайт: URL: http://arxiv.org/abs/1809.02627 (дата обращения 27.03.2024).

12. M. Wydmuch, M. Kempka & W. Jaśkowski, ViZDoom Competitions: Playing Doom from Pixels / M. Wydmuch, M. Kempka, W. Jaśkowski // IEEE Transactions on Games, vol. 11, no. 3, 2019. сайт: URL: http://vizdoom.cs.put.edu.pl/research (дата обращения 27.03.2024).

13. Cinjon Resnick. Pommerman: A multi-agent playground / Cinjon Resnick, Wes Eldridge, David Ha et al. // CoRR, abs/1809.07124, 2018. сайт: URL: http://arxiv.org/abs/1809.07124 (дата обращения 27.03.2024)..

14. Chris Bamford, Shengyi Huang, and Simon M. Lucas. Griddly: A platform for AI research in games / Chris Bamford, Shengyi Huang, Simon M. Lucas // CoRR, abs/2011.06363, 2020. сайт: URL: https://arxiv.org/abs/2011.06363 (дата обращения 27.03.2024).

15. William H. Guss. The minerl 2019 competition on sample efficient reinforcement learning using human priors / William H. Guss, Cayden Codel, Katja Hofmann et al. // CoRR, 2021.

16. David Silver. Mastering the game of Go with deep neural networks and tree search / David Silver, Aja Huang, Chris J. Maddison et al. // Nature, 2016. ISSN 0028-0836. doi: 10.1038/nature16961.

17. Christopher Berner. Dota 2 with large scale deep reinforcement learning / Christopher Berner, Greg Brockman, Brooke Chan et al. // CoRR, abs/1912.06680, 2019. сайт: URL: http://arxiv.org/abs/1912.06680 (дата обращения 27.03.2024).

18. Alphastar: Mastering the real-time strategy game starcraft ii. сайт: URL: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii (дата обращения 27.03.2024).

19. G. Michalos. Automotive assembly technologies review: challenges and outlook for a flexible and adaptive approach / G. Michalos, S. Makris, N. Papakostas et al. // CIRP Journal of Manufacturing Science and Technology, 2010. сайт: URL: http://www.sciencedirect.com/science/article/pii/S1755581709000467 (дата обращения 27.03.2024).

20. J. Kruger. Innovative control of assembly systems and lines / J. Kruger, L. Wang, A. Verl et al. // CIRP Annals 2017;66(2):707 – 730. сайт: URL: http://www.sciencedirect.com/science/article/pii/S000785061730149X (дата обращения 27.03.2024).

21. Digital Twin Market to Demonstrate a Robust Growth Over 2025. – сайт: URL: https://www.marketresearchfuture.com/press-release/digital-twin-industry (дата обращения 15.05.2024)

22. Prepare for the Impact of Digital Twins. – сайт: URL: https://www.gartner.com/smarterwithgartner/prepare-for-the-impact-of-digital-twins (дата обращения 15.05.2024)

23. Digital Twins and Digital Thread for Buildings and Infrastructure. – сайт: URL: https://www.idc.com/getdoc.jsp?containerId=US47451521 (дата обращения 15.05.2024)

24. Digital Twin Applications and Use Cases. – сайт: URL: https://unity.com/ru/solutions/digital-twin-applications-and-use-cases (дата обращения 15.05.2024)

25. I. Verner. Robot online learning through digital twin experiments: A weightlifting project / I. Verner, D. Cuperman, A. Fang et al. // Auer, ME, Zutin, DG, editors. Online Engineering & Internet of Things. Cham: Springer International Publishing. ISBN 978-3-319-64352-6, 2018.

26. T. Hassel, O. Hofmann. Reinforcement learning of robot behavior based on a digital twin. / T. Hassel, O. Hofmann // Marsico, MD, di Baja, GS, Fred, ALN. Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2020. SCITEPRESS. ISBN 978-989-758-397-1, 2020.

27. G. Chryssolouris, D. Mavrikios, N. Papakostas. Digital manufacturing: History, perspectives, and outlook / G. Chryssolouris, D. Mavrikios, N. Papakostas et al. // Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture 2009. – сайт: URL: https://doi.org/10.1243/09544054JEM1241 (дата обращения 27.03.2024)

28. Iqbal, Jamshed, R. Ul Islam, and Hamza Khan. Modeling and analysis of a 6 DOF robotic arm manipulator / Iqbal, Jamshed, R. Ul Islam, Hamza Khan // Canadian Journal on Electrical and Electronics Engineering 3, no. 6, 2012.

29. H. C. Nguyen. Research on the application of Neural Networks in identification and control of the robotic arms - A nonlinear dynamic object / H. C. Nguyen // Journal of Science and Technology – the University of Da Nang, vol. 5, 2016.

30. Meng, Wei, Quan Liu, Zude Zhou. Recent development of mechanisms and control strategies for robot-assisted lower limb rehabilitation / Meng, Wei, Quan Liu, Zude Zhou et al. // Mechatronics 31, 2015.

31. Leon Žlajpah. Simulation in robotics / Leon Žlajpah // Mathematics and Computers in Simulation 79, no. 4, 2008.

32. Metin Topaz and Serdar Kucuk. Dynamics simulation toolbox for industrial robot manipulators / Metin Topaz, Serdar Kucuk // Computer Applications in Engineering Education 18, no. 2, 2010.

33. Wang, Zhi-Xing. Kinematical analysis and simulation of an industrial robot based on Matlab / Wang, Zhi-Xing, Wen-Xin Fan, Bao-Cheng Zhang, Yuan-Yuan Shi// Jidian Gongcheng/Mechanical & Electrical Engineering Magazine 29, no. 1, 2012.

34. Hee-Jun Kang and Young-Shick Ro. Robot manipulator modeling in Matlab-SimMechanics with PD control and online gravity compensation / Hee-Jun Kang, Young-Shick Ro // In International Forum on Strategic Technology 2010, IEEE, 2010.

35. Kim Suseong, Seungwon Choi, and H. Jin Kim. Aerial manipulation using a quadrotor with a two DOF robotic arm / Kim Suseong, Seungwon Choi, H. Jin

Kim // In 2013 IEEE/RSJ International Conference Intelligent Robots and Systems. IEEE, 2013.

36. Mohammed Abu Qassem, I.M. Abuhadrous, H. Elaydi. Modeling and Simulation of 5 DOF educational robot arm / Mohammed Abu Qassem, I.M. Abuhadrous, H. Elaydi // In Conference: Advanced Computer Control (ICACC), 2010 2nd International Conference on, Volume. 5, 2010.

37. P.Hemalatha, C.K.Hemantha Lakshmi, and Dr.S.A.K.Jilani. Real-time Image Processing based Robotic Arm Control Standalone System using Raspberry pi / P.Hemalatha, C.K.Hemantha Lakshmi, Dr.S.A.K.Jilani // SSRG International Journal of Electronics and Communication Engineering 2.8, 2015.

38. Ramanideepthi Tanneeru, Ajay Jandrajupalli, B. Kiran Kumar. Design and Analysis of Robot Arm using Matlab & ANYSYS / Ramanideepthi Tanneeru, Ajay Jandrajupalli, B. Kiran Kumar // SSRG International Journal of Mechanical Engineering 2.3, 2015.

39. Annamareddy Srikanth. KINEMATIC ANALYSIS OF 3 D.O.F OF SERIAL ROBOT FOR INDUSTRIAL APPLICATIONS / Annamareddy Srikanth, M.Sravanth, V.Sreechand, K.Kishore Kumar // International Journal of Engineering Trends and Technology (IJETT), 2013.

40. B. Lakshmigandan. EEG Based Brain Controlled Mobile Arm Pick and Place robot / B. Lakshmigandan, V. Janakiram, S. Jothi Raj, V. Salai Selvam // International Journal of Engineering Trends and Technology (IJETT), 2017.

41. Wang Daina. A study of robot subjectivity. / Wang Daina // Chengdu: Chengdu University of Technology, 2018.

42. Chen Jun. Research on real-time simulation system for multi-degree-of-freedom robotic arm. / Chen Jun // Harbin: Harbin Engineering University, 2012.

43. Edelman, B. J. Noninvasive neuroimaging enhances continuous neural tracking for robotic device control / Edelman, B. J., Meng, J., Suma, D. et al. // Science Robotics, 2019. – сайт: URL: https://doi.org/10.1126/scirobotics.aaw6844 (дата обращения 12.04.2024)

44. Xu Kou. Inverse kinematic solution and trajectory planning of a six-degree-of-freedom robotic arm / Xu Kou // Guangzhou: Guangdong University of Technology, 2016.

45. Ma Yuhao. Six-degree-of-freedom robotic arm obstacle avoidance trajectory planning and control algorithm research / Ma Yuhao // University of Chinese Academy of Sciences, 2019.

46. Zuo Fuyong. SCARA robot trajectory planning and simulation based on MATLAB Robotics toolbox / Zuo Fuyong, Hu Xiaoping, Xie Ke, et al. // Haapasalo, Xiang Tan: Journal of Hunan University of Science and Technology, 2012.

47. Wang Yongjie. Optimisation of structural parameters for multi-degree-of-freedom robotic arms / Wang Yongjie // Beijing Institute of Technology, 2016.

48. Li Baofeng. Space robot workspace calculation based on Monte Carlo method / Li Baofeng, Sun Hanxu, Jia Qingxuan, et al // Journal of Spacecraft Engineering, 2012.

49. Microsoft Robotics Studio - робототехника для всех. – сайт: URL: https://www.bytemag.ru/articles/detail.php?ID=6470 (дата обращения 12.04.2024)

50. Hughes C. Robot Programming: A Guide to Controlling Autonomous Robots: Pearson Education / C. Hughes, T. Hughes // Pearson Education, 2016.

51. Пикалёв Я.С. Анализ существующих симуляторов робототехнических систем / Я.С. Пикалёв // Проблемы искусственного интеллекта, 2017. – 15 с.