ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

На правах рукописи
УДК xxx.xxx

Староверов Алексей Витальевич

Иерархические методы и алгоритмы визуальной навигации внутри помещений с обучаемыми навыками

Специальность 1.2.2—
«Математическое моделирование, численные методы и комплексы программ»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
кандидат физико-математических наук, доцент
Панов Александр Игоревич

Долгопрудный— 2023

Оглавление

Стр.

Введение

В последнее время у научного сообщества появился большой интерес к задачам воплощенного искусственного интеллекта (ВИИ/ Embodied AI) [1—4]. Особенность их заключается во взаимодействии агента (робота) с окружающими объектами в человеко-ориентированных средах. Одной из главной задачей ВИИ является способность агента свободно ориентироваться в новых для него средах и оперировать семантическими априорными знаниями на основе прошлого опыта [5]. Для этого мобильный робот должен уметь решать различные подзадачи, или, иными словами, обладать разными навыками, такими как: обнаружение и сегментирование объектов внешней среды [6—8], локализация и картографирование (SLAM, [9; 10]), навигация [11], планирование [12] и т.д. На практике, эти навыки объединяются в единую систему, где каждый модуль отвечает за свою подзадачу и может быть реализован как на основе нейронных сетей, так и классическим способом [13; 14]. Альтернативой данного подхода являются полностью самообучаемые системы [15; 16], которые могут быть сформулированы, как связанные с обучением с подкреплением [17]. Агент учится выполнять оптимальные последовательности действий для выполнения задачи с максимальной совокупной наградой, получая обратную связь от окружающей среды. При этом агент не получает прямых инструкций для решения задачи. Самообучающиеся системы, в которых используются агенты, основанные на обучении с подкреплением, достигли впечатляющих результатов во все более сложных областях [2; 15; 18; 19].

Одним из перспективных подходов для объединения навыков агента в единую архитектуру является иерархическое обучение с подкреплением (HRL) [20]. Иерархический подход позволяет разделить сложную задачу на множество подзадач. Для людей это является естественной процедурой. Однако, остается до конца не изученным вопрос – как именно человеку удается находить соответствующую иерархическую структуру. Поиск хорошей декомпозиции на подзадачи часто творческая задача, решение которой представляет серьезную проблему. Несмотря на то что в этом направлении получен ряд достижений [21], автоматическое построение иерархической структуры остается открытой проблемой в обучении с подкреплением. Методы HRL позволяют агентам раз-

ложить задачу на более простые подзадачи. HRL-подходы обучают агентов различным уровням стратегии, каждый из которых специализируется на принятии решений в различных временных масштабах.

Для успешного применения обучаемых алгоритмов, однако, требуется симуляционная среда, которая будет обеспечивать большие объемы данных и симулировать все внешние условия, под которые агент будет приспосабливаться. Особенно важно это при переносе агента из симулятора в реальность. Если в реальности условия будут сильно отличаться, то агент не будет способен их преодолеть, а обучение в реальном мире занимает непропорционально много времени и может быть небезопасным, так как на первых шагах агент будет предпринимать случайные действия, перед тем как выучить приемлемую стратегию поведения. Это мотивировало созданию таких симуляционных сред как Habitat [1] и BPS [22], которые будут использованы далее в работе.

В данных средах за последнее время было продемонстрировано много успешных алгоритмов, которые применяя модульные [14; 23—25] или иерархические подходы [23; 26] превосходили классические подходы [15], что послужило убедительным доказательством того, что системы искусственного интеллекта могут быть масштабированы для работы в сложных, динамичных средах и применяться на реальных робототехнических системах. Однако как показывает опыт соревнований Habitat Challenge [1], современные методы недостаточно хорошо справляются с задачами, где требуется семантическое понимание сцены, успешно завершая эпизоды только в половине случаев.

В следствии этого была выбрана тема диссертационной работы.

Целью данной работы является повышение автономности робототехнических систем в задаче навигации на основе разработки гибридных методов визуальной навигации с использованием обучаемых и необучаемых навыков с возможностью использования на реальном роботе.

Для достижения поставленной цели были определены и решены следующие **задачи:**

– Разработать иерархический обучаемый метод решения задачи навигации в 2D и визуальных 3D средах. Интегрировать семантическую сегментацию, картирование и локализацию в обучаемый метод поиска целевых объектов.

– Для задачи навигации к семантическим объектам разработать метод визуальной навигации с использованием минимально необходимых априорных знаний о структуре среды. Выделить в поведении агента предварительно обученные стратегии поведения, которые можно объединить и повторно использовать в различных навигационных задачах без каких-либо изменений. Исследовать методы 3D реконструкции сцен и возможность использования их как симулятора для дообучения навыков агента для применения на реальном мобильном роботе.
– Разработать гибридный алгоритм решения задачи навигации, объединяющий классические и обучаемые навыки агента с обучаемым модулем переключения стратегий. Адаптировать предложенный метод под реального мобильного робота в зашумленных условиях.

Научная новизна:

– Был предложен метод интеграции семантической сегментации, картирования, локализации и обучения с подкреплением для повышения эффективности исследования окружающей среды, поиска целевого объекта и быстрой навигации к нему. Для задачи навигации до точки был предложен иерархический метод с выделением подцелей. Особенностью данного метода является одновременное обучение всех уровней иерархий в условиях разряженной награды от среды.
– Задача поиска целевых объектов на карте была сформулирована через навыки агента и предложен вариант использования опорных областей для ускорения исследования сцен для мобильного робота в человеко-ориентированных помещениях. Был представлен метод иерархической стратегии с ориентирами HLPO, который использует доступную информацию о ориентирах и на основе нее выстраивает иерархию из заранее обученных навыков агента, что улучшает способность агента исследовать среду в два раза. Полученный метод был перенесен на реального робота путем дообучения стратегии в реконструированной среде реального помещения.
– Выполнено оригинальное исследование, в рамках которого был разработан объединяющий классические и на основе обучения с подкреплением навыки агента алгоритм SkillFusion, показавший в задаче навигации к целевым объектам свое преимущество перед только классическими или

обучаемыми стратегиями. Выбор навыков осуществляется на основе модуля оценки их полезности в каждый момент времени.

Теоретическая значимость работы заключается в следующем:

– Предложен метод, совмещающий классические алгоритмы планирования и методы обучения с подкреплением. Его главная особенность заключается в том, что он учитывает преимущества обоих парадигм и динамически выбирает в зависимости от состояния агента и оценки функции полезности каждого навыка, какой навык использовать в текущий момент.
– Предложен новый подход к решению задачи навигации поиска целевых объектов с использованием ориентиров. С обновленной формулировкой задачи была создана новая иерархическая архитектура, в которой используются навыки, которые можно комбинировать и повторно использовать в различных навигационных задачах без изменений.

Практическая значимость работы заключается в следующем:

– Для методов семантической сегментации, картографирования и локализации был собран и выложен в открытый доступ оригинальный набор данных HISNav.
– Отработан метод 3D реконструкции реального помещения и использование его в симуляторе для обучения алгоритмов на основе обучения с подкреплением.
– Предложенный способ использования методов обучения с подкреплением для задач навигации был испытан на мобильных роботах в реальных условиях и легко адаптируется на разные робототехнические платформы. В будущем данный подход может быть расширен на семантически более сложные постановки задачи, тем самым повышая степень автономности робототехнических систем.

Методология и методы исследования.

Разрабатываемые алгоритмы основываются на методах машинного обучения, теории графов, методах оптимизации и статистике. Основным методом оценки эффективности предложенных результатов в данном исследовании является численный эксперимент. Сравнительный анализ эффективности алгоритмов проводится на основе статистического анализа нескольких запусков каждого из алгоритмов. В дополнение к этому, в работе проводится абляционное

исследование, которое позволяет оценить вклад отдельных элементов дизайна нового предложенного решения в конечный результат. Реализация всех рассматриваемых алгоритмов и экспериментов осуществлена с использованием языков программирования Python3 и bash, а также дополнительных технологий, таких как библиотека машинного обучения PyTorch, программа для контейнеризации приложений docker, библиотека numpy и другие.

Положения, выносимые на публичное представление:

- Обучаемый метод выделения подцелей с интеграцией метода SLAM и семантической сегментации для задачи навигации до точки и поиска заданных объектов.
- Оригинальный алгоритм визуальной навигации с использованием опорных областей для ускорения исследования сцен для мобильного робота в человеко-ориентированных средах.
- Гибридный метод, объединяющий классические и обучаемые подходы для решения задачи поиска целевых объектов на основе функции полезности каждого навыка. Данный метод занял первое место на международном соревновании по навигации Habitat Challenge 2023 [1]

Достоверность результатов, полученных в ходе исследования, обеспечивается использованием методики численного эксперимента. Представленные алгоритмы описаны подробно, что позволяет повторить их результаты. Для каждого алгоритма представлено детальное описание и код выложен в открытый доступ. Многие из полученных данных согласуются и дополняют результаты, полученные в работах других исследователей.

Апробация работы. Основные результаты работы докладывались на:

- XXI Международная научно-техническая конференция “Нейроинформатика-2019”, 7-11 октября 2019, Москва
- XXII Международная научно-техническая конференция “Нейроинформатика-2020”, 12-16 окября 2020, Москва
- VI Всероссийский научно-практический семинар “Беспилотные транспортные средства с элементами искусственного интеллекта” (БТС-ИИ 2021), 16-19 ноября 2021, Москва
- Научно-практический семинар Центра когнитивного моделирования ФПМИ МФТИ, 11 мая 2023, Москва

[1] https://aihabitat.org/challenge/2023/

– IEEE/CVF Conference on Computer Vision and Pattern Recognition 2023, “CVPR-2023”, Embodied AI Workshop, 18-22 июня, Ванкувер

Содержание диссертации соответствует паспорту специальности 1.2.2. Математическое моделирование, численные методы и комплексы программ, в частности, пунктам:

– п. 2 – Разработка, обоснование и тестирование эффективных вычислительных методов с применением современных компьютерных технологий.

– п. 3 – Реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента.

– п. 5 – Разработка новых математических методов и алгоритмов валидации математических моделей объектов на основе данных натурного эксперимента или на основе анализа математических моделей.

– п. 8 – Комплексные исследования научных и технических проблем с применением современной технологии математического моделирования и вычислительного эксперимента.

Личный вклад. В работе [27] – разработка метода интеграции семантической сегментации, картирования, локализации и обучения с подкреплением для задачи навигации и иерархического метода выделения подцелей; В работе [28] автор предложил постановку задачи навигации с ориентирами, разработал метод иерархической стратегии с ориентирами и метод переноса обученной стартегии на реального робота. В работе [29] автор предложил и реализовал метод интеграции классических и обучаемых навыков.

Публикации. Основные результаты по теме диссертации изложены в 3 печатных изданиях [27—29], 3 в периодических научных журналах, индексируемых Scopus, в том числе 3 из которых опубликованы в журналах первого квартиля.

Объем и структура работы. Диссертация состоит из введения, четырёх глав, заключения и двух приложений.

Полный объём диссертации составляет 111 страниц с 35 рисунками и 10 таблицами. Список литературы содержит 116 наименований.

Глава 1. Методы решения задачи визуальной навигации

Разработка систем автоматизированного принятие решений стала одним из главных приоритетов в последнее время [17]. Данные системы уже встречаются во всех аспектах нашей жизни. От классических алгоритмов планирования [30], используемых в системах навигации [31—33], до сложных алгоритмов принятия решений, которые могут поддерживать полноценный мультимодальный диалог [2; 3] с пользователем на любые темы. По мере цифровизации нашей внешней среды и большим внедрением информационных технологий во все большее количество устройств с которыми мы взаимодействуем на постоянной основе, становятся актуальны системы которые могут оказывать и физическую помощь человеку. Автоматизированные системы в наше время уже начинают внедряться в сектора низкоквалифицированных рабочих мест и способны заменять как курьеров или кассиров, делать большую часть механических операции на производстве и обеспечивать складскую логистику. Однако большинство существующих решений справляются с задачей только в строго контролируемых и заранее описанных условиях, выполняют только одну конкретную задачу, но могут в ней превосходить человека по качеству и производительности. Но что является наибольшим ограничением, текущие решения требуют многолетнего детального проектирования командами инженеров. Одним из перспективных методов решения этих проблем является внедрение алгоритмов способных учиться на своих неизбежных ошибках. Уже существуют системы на основе обучения, которые в таких задачах как перевод текста, классификация фотографий или генерация речи превосходят все предыдущие написанные в явном виде алгоритмы. Такие системы называют узким ИИ и их особенностью является большой набор размеченных данных, которые модель учится обобщать. Робототехнические же системы относят к воплощенному ИИ, где агент выполняя действия в среде, меняет ее состояние. Для безопасного сбора данных в данных типах задач используют симуляционные среды. За последние годы появилось множество виртуальных сред, применимых для алгоритмов обучения с подкреплением, от напоминающих видеоигры до основанных на трехмерных моделях реальных объектах и окружений. Последние представляют наибольший

интерес, так как позволяют агенту получить данные приближенные к реальности, что важно при переносе стратегии в реальный мир.

Для воплощенного ИИ, как и для робототехнических систем в целом, одной из главных задач является навигация. В данной работе будут рассматриваться несколько модификаций к основной постановке задачи навигации. Это перемещение агента в ранее неизвестной ему среде от точки к точке и поиск ключевых объектов на сцене, задаваемых семантической категорией. Классические методы в робототехнике обычно решают данный класс задач путем создание карты окружения, и затем локализации и планирования в ней. Однако в такой постановке сложно учитывать априорные знания о структуре внешней среды, которые помогают в похожих условиях ориентироваться человеку. Это может быть как общее понимание планировки помещений, связь семантических классов объектов между собой и сходства одних помещений с другими. Для решение этих задач в последнее время стали успешно применяться методы компьютерного зрения (CV) и обучения с подкреплением (RL), которые могут в качестве входной информации принимать изображение с камер и на выходе выдавать действия, которые должен выполнить агент для максимизации своего вознаграждения в среде.

1.1 Постановка навигационной задачи

В данной работе будут рассмотрены две навигационные задачи, навигация до точки (PointGoal Navigation) и визуальная навигация до целевого объекта (Visual ObjectGoal Navigation). Обе этих подзадачи могут решаться как классическим подходом, с одновременным построением карты и локализацией (Simultaneous Localization and Mapping, SLAM), так и методами обучения с подкреплением (Reinforcement Learning, RL).

В случае подхода, основанного на SLAM, робот в процессе движения строит карту помещений и наносит на эту карту целевой объект, как только обнаружил его. После того, как целевой объект нанесен на карту, робот строит до него маршрут по полученной карте и движется по этому маршруту.

В случае подходов, основанных на RL, информация об окружающей среде вместо карты запоминается в скрытые слои рекуррентной нейронной сети (RNN). Нейросеть принимает на вход данные с робота и выдает действие, оптимальное в данный момент для решения задачи - проехать вперед, повернуть налево или направо, остановиться по достижении целевого объекта.

Обучаемая стратегия определяется через марковский процесс принятия решений (MDP) $< S,A,T,R,\gamma >$, где S - это набор состояний, A- это набор доступных действий, $T(s_{t+1}|s_t,a_t)$ - это функция перехода, R - это функция вознаграждения, а γ - это коэффициент дисконтирования. В предлагаемой постановке задачи, состояния s_t не полностью наблюдаемые. Агенту доступна только неполная информация о состоянии на каждом временном шаге - наблюдение o_t. Предполагается, что агент использует аппроксиматор f состояния s_t из истории наблюдений: $s_t \approx f(o_t,o_{t-1},\dots)$ (на практике это реализуется как нейронная сеть агента, которая отвечает за принятие решений).

Далее будут описаны особенности симуляционной среды Habitat, которые агенту необходимо учитывать при построении пути до цели.

Динамика столкновений. Некоторые симуляторы [34] используют грубую нерегулярную навигационную сетку, где агент "телепортируется" из одного места в другое (на расстоянии 1-2 м). Другие симуляторы [35] используют тонкую регулярную сетку (с разрешением 0,01 м), где агент перемещается по незанятым ячейкам без столкновений или промежуточных шагов. В симуляторе Habitat и экспериментах в данной работе, используется более реалистичная модель столкновений - агент перемещается в непрерывном пространстве состояний, и движение может вызывать столкновения, приводящие к частичному (или отсутствию) продвижения в заданном направлении. Что важно, агент может выбрать движение вперед (на 0,25 м) и оказаться в месте, которое не на 0,25 м впереди от того места, где он начал; таким образом, одометрия не является тривиальной даже при отсутствии шума актуаторов.

Спецификация эпизода. Агент при старте эпизода инициализируется в начальной позиции и ориентации, которые выбираются равномерно случайным образом из всех проходимых позиций на полу среды. Целевая позиция (или объект) выбирается таким образом, чтобы она находилась на том же этаже, и существовал проходимый путь от начальной позиции агента. В течение эпизода агенту разрешается совершить до 500 действий. Этот порог значитель-

но превышает количество шагов, которые требуются оптимальному агенту для достижения всех целей.

1.1.1 Задача навигации до точки

Задача навигации до точки состоит в следующем - агент инициализируется в случайной начальной позиции и ориентации в среде и ему предлагается переместиться к целевым координатам, которые предоставляются относительно положения агента; нет карты местности, и агент должен использовать только свои сенсорные данные для навигации.

Агент физически воплощен в виде цилиндрической примитивной формы с диаметром 0,2 м и высотой 1,5 м. Пространство действий состоит из четырех действий: повернуть налево, повернуть направо, двигаться вперед и остановиться. Эти действия отображаются на идеализированные актуаторы агента, которые обеспечивают поворот на 10 градусов или линейное перемещение на 0,25 м. Действие остановиться позволяет агенту сигнализировать о том, что он достиг цели.

Из доступных сенсоров, агент оснащен одной RGB камерой, расположенной на высоте 1,5 м от центра основания агента и ориентирована вперед. Камера имеет разрешение 256×256 и угл обзора 90 градусов. Кроме того, доступен бесшумный датчик глубины в той же позиции и ориентации, что и RGB камера. Угол обзора и разрешение датчика глубины соответствуют параметрам камеры. Агенты, использующие только RGB камеру, обозначаются как RGB, агенты, использующие датчик глубины - как Depth, а агенты, использующие оба датчика - как RGBD. Все агенты оснащены бесшумными GPS и компасом, то есть они имеют доступ к координатам своего местоположения и, неявно, к своей ориентации относительно целевой позиции.

Эпизод считается успешным, только если агент выполнит действие остановки в пределах 0,2 м от целевых координат, измеряемых по геодезическому расстоянию вдоль кратчайшего пути от позиции агента до целевой позиции. Если агент совершает 500 действий без выполнения вышеуказанного условия,

эпизод заканчивается и считается неудачным. Производительность измеряется с помощью метрики «Успех, взвешенный длиной пути» (SPL, [36]).

$$SPL = \frac{1}{N}\sum_{i=1}^{N}\frac{l_i}{max\,(p_i,l_i)}, \tag{1.1}$$

где l_i — длина кратчайшего пути между начальной точкой и целевой точкой. p_i — длина пути, пройденного агентом в течении эпизода.

1.1.2 Визуальная навигация до целевого объекта

Данная задача состоит в визуальной навигации до целевого объекта одного из шести типов: стул, диван, кровать, комнатное растение, телевизор и туалет.

Для выполнения этой навигационной задачи виртуальный робот (агент) получал на каждом шаге изображения и глубины с RGBD камеры с ограничением глубины от 0.5 до 5 метров, а также точные данные о местоположении и ориентации (GPS+Compass). Агент мог на каждом шаге выполнить действие одного из шести типов: проехать 25 см вперед, повернуться на 30 градусов влево, повернуться на 30 градусов вправо, наклонить камеру на 30 градусов вниз, поднять камеру на 30 градусов вверх, остановиться и завершить эпизод. Эпизод считался успешно завершенным, если агент выдал действие stop в радиусе 1 метра от целевого объекта, и целевой объект находился в прямой видимости. На выполнение давалось 500 секунд. В качестве метрики используется SPL (успех, взвешенный по длине пути) (2.4), где l_i — длина кратчайшего пути между начальной точкой и целевым объектом, наиболее близкого к точке, где заканчивается траектория агента. p_i — длина пути, пройденного агентом в эпизоде.

Агент оснащен камерой с горизонтальным углом обзора в 42 градуса на высоте 1.31 м. Чтобы охватить такой камерой все окружающее пространство, а также увидеть низкие объекты, агенту необходимо много вращаться на месте, наклонять и поднимать камеру. Также в пространство действий для данной задачи был добавлен шум - например, действие вперед двигало агента не ровно на 25 см вперед, а с небольшим случайным отклонением.

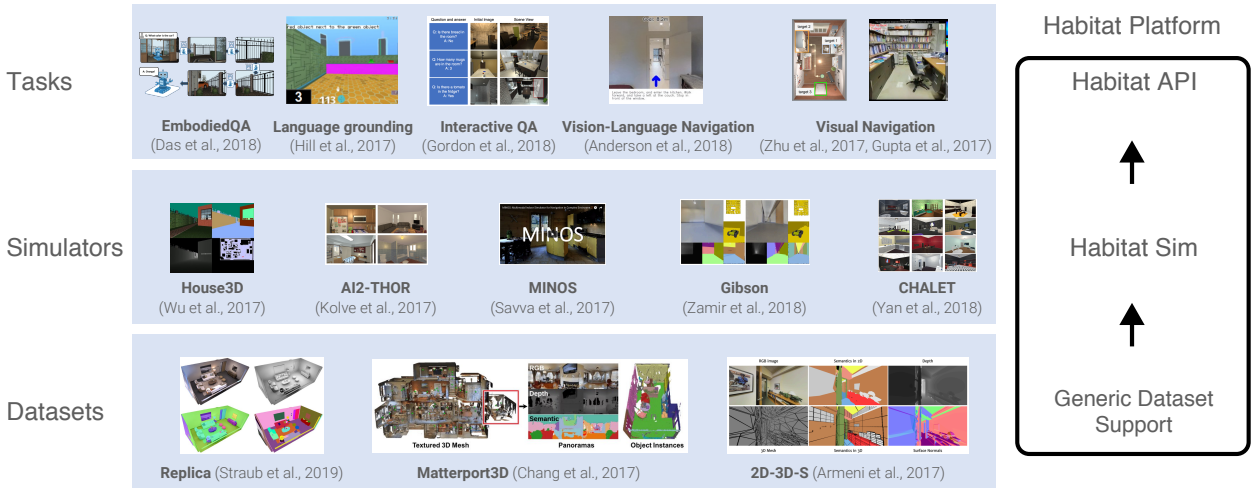

Рисунок 1.1 — Программное обеспечение Habitat для обучения воплощенных агентов включает в себя: (1) наборы данных, предоставляющие 3D-сцены с семантическими аннотациями, (2) симулятор, который отображает 3D-сцены, в которых может быть смоделирован воплощенный агент, и (3) постановка задач, которые определяют цель агента и параметры входных данных.

1.2 Среды для воплощенного ИИ

Для воплощенного ИИ, акценты делаются на на активном восприятии, долгосрочном планировании, обучения от взаимодействия со средой. Со всем этим робот может столкнуться в условиях реального мира и обучаться на собранных реальных данных возможно. Однако, обучение роботов в реальном мире будет происходить довольно медленно (реальный мир работает не быстрее реального времени и не может быть распараллелен), плохо обученные агенты могут невольно причинить вред себе, окружающей среде или другим, это требует больших ресурсов и затрудняет тестирование алгоритмов, поскольку точное воспроизведение условий между экспериментами сложно обеспечить.

В контексте воплощенного ИИ, симуляторы помогают преодолеть вышеупомянутые проблемы - они могут работать на порядки быстрее, чем в реальном времени так как могут быть распараллелены по кластеру; обучение в симуляции безопасно, дешево и позволяет проводить справедливое сравнение и оценку прогресса. Как только перспективный подход будет разработан и протестирован в симуляции, его можно перенести на физические платформы, работающие в реальном мире.

В последнее время, появилось много симуляторов, которые позволяют тестировать и собирать данные для RL алгоритмов. Наиболее известным набором сред, стал Gym [37], предоставляющий интерфейс для полностью наблюдаемых игр, таких как Arcade Learning Environment (ALE, [38]). Gym сосредоточен на эпизодическом обучении с подкреплением, где опыт агента разбивается на ряд эпизодов. В каждом эпизоде начальное состояние агента случайным образом выбирается из распределения, и взаимодействие продолжается до тех пор, пока среда не достигнет конечного состояния. Цель в эпизодическом обучении с подкреплением - максимизировать ожидание общего вознаграждения за эпизод. Для задач непрерывного управления стандартом стал набор сред Mujuco ([39]), из его преимуществ также является наличие исходного открытого кода. OpenSpiel ([40]) предоставляет подход для RL в таких играх, как шахматы, покер, го, крестики-нолики и др. Из частично наблюдаемых 3D-симуляторов, с более игровыми средами, можно отметить симулятор ViZDoom ([41]) и DeepMind-Lab ([42]).

В данной работе наибольший интерес представляют фотореалистичные среды, где можно задавать параметры агента, что бы он был близок к реальному роботу, а наблюдения с сенсоров робота и его действия также содержали шумы и были похожи на реальное поведение. В итоге, был выбран симулятор Habitat [1].

1.2.1 Симулятор Habitat

Симулятор Habitat [1] состоит из двух компонентов:

- Habitat-Sim: гибкий и высокопроизводительный 3D-симулятор с настраиваемыми агентами, несколькими датчиками и универсальной обработкой 3D-наборов данных (с встроенной поддержкой наборов данных Matterport3D [43], Gibson [44] и HM3D [45]).
- Habitat-API: модульная библиотека высокого уровня для конечной разработки алгоритмов воплощенного искусственного интеллекта - где определены задачи воплощенного искусственного интеллекта (например, навигация, выполнение инструкций, ответ на вопросы), настройка

и обучение воплощенных агентов (через имитацию или обучение с подкреплением или через классический SLAM) и оценка с использованием стандартных метрик.

Архитектура и реализация Habitat сочетают в себе модульность и высокую производительность (Рис. 1.1). При отображении сцены из набора данных Matterport3D [43], Habitat-Sim достигает нескольких тысяч кадров в секунду (fps) при однопоточном выполнении и может достигать более 10 000 fps при многопроцессном выполнении на одном графическом процессоре, что на порядки быстрее аналогов. Habitat-API позволил в данной работе обучать и оценивать воплощенных агентов с использованием различных классов методов и в различных наборах данных 3D-сцен.

1.2.2 Набор данных Habitat-Matterport 3D

Набор данных Habitat-Matterport 3D (HM3D, [45]) содержит 1000 3D-реконструкций зданий из самых разных мест реального мира. Каждая сцена в наборе данных состоит из текстурированной трехмерной реконструкции таких помещений как многоэтажные жилые дома, магазины и другие частные помещения. HM3D превосходит существующие наборы данных, доступные для академических исследований, по физическому масштабу, полноте реконструкции и визуальной точности. HM3D содержит 112,5 тыс. м2 проходимого пространства, что на 1,4 - 3,7 раза больше, чем в других наборах данных масштаба зданий, таких как MP3D [43] и Gibson [44] (Таб. 1). По сравнению с существующими фотореалистичными 3D-наборами данных, изображения из HM3D, имеют на 20 - 85% более высокую визуальную достоверность по отношению к аналогичным изображениям, полученным с помощью реальных камер, а сетки HM3D имеют на 34 - 91% меньше артефактов, которые встречаются из-за неполной реконструкции поверхности. Увеличенный масштаб, достоверность и разнообразие HM3D напрямую влияют на производительность воплощенных агентов, обученных с его использованием. Авторы данных пишут, что HM3D является «парето-оптимальным» в следующем смысле - агенты, обученные выполнять навигацию до точки (PointGoal) на HM3D, достигают наивысшей производи-

Dataset	Replica [46]	MP3D [43]	Gibson [44] (4+)	ScanNet [47]	HM3D [45]
Number of scenes	18	90	571 (106)	1613	1000
Floor area (m^2)	$2.19k$	$101.82k$	$217.99k$ ($17.74k$)	$39.98k$	$365.42k$
Navigable area (m^2)	$0.56k$	$30.22k$	$81.84k$ ($7.18k$)	$10.52k$	$112.50k$
Navigation complexity	5.99	17.09	14.25 (11.90)	3.78	13.31
Scene clutter	3.4	2.99	3.14 (3.04)	3.15	3.90

Таблица 1 — Сравнение HM3D с другими существующими наборами данных с 3D-реконструкциями [45].

тельности независимо от того, оцениваются ли они на данных HM3D, Gibson или MP3D. Подобное утверждение не может быть сделано про обучение на других наборах данных, где обученные агенты при переносятся на другие наборы данных теряют качество.

1.3 Обучение с подкреплением

Машинное обучение (ML) можно условно поделить на три класса алгоритмов: обучение с учителем (SL), обучение без учителя (UL) и обучение с подкреплением (RL). В обучении с учителем агент получает размеченные данные (X и Y), и его целью является найти отображение, которое по входным данным X находит Y для новых значений X, не входящих в обучающую выборку. Для примера, агенту могут быть даны рисунки животных (X), для каждого из которых есть соответствующая маркировка с указанием вида животного (Y). Затем агент изучает отображение, которое позволяет определить, какие виды животных изображены на фотографиях. В обучении без учителя агенту дается только набор данных X, и агент учится находить распределение данных $p(X)$. Это может быть группировка рисунков по типам животным.

Целью алгоритмов обучения с подкреплением является поиск стратегии π которая отображает пространство состояний S в пространство действий a в средах с заданной функцией вознаграждения, причем выбранные действия максимизируют накопленное вознаграждение, полученное при взаимодействии агента с окружающей средой [17; 48]. Агент в RL обычно выполняет последовательность действий в окружающей среде в течение эпизода или траектории, которые заканчивается условием завершения. Условие завершения обычно со-

ответствует состоянию, соответствующему успеху или неудаче, но может быть активировано, когда превышено максимальное количество действий за эпизод.

1.3.1 Марковский процесс принятия решения

В формальном виде, взаимодействие агента с окружающей средой (Рис. 1.2) описывается Марсковским процессом принятия решения (Markov Decision Process или MDP). Его главное свойство Марковости заключается в том, что выбор оптимального действия в текущем состоянии агента не зависит от предыдущих состояний и ранее принятых агентом действий. MDP определяется как кортеж из $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, где:

- $\mathcal{S}$ — множество состояний.
- $\mathcal{A}$ — множество действий.
- $\mathcal{P}$ — функция переходов, которая задает динамику среды, $P(s'|s,a)$.
- $\mathcal{R}$ — функция вознаграждения.
- γ — коэффициент дисконтирования, $(0 \leq \gamma \leq 1)$.

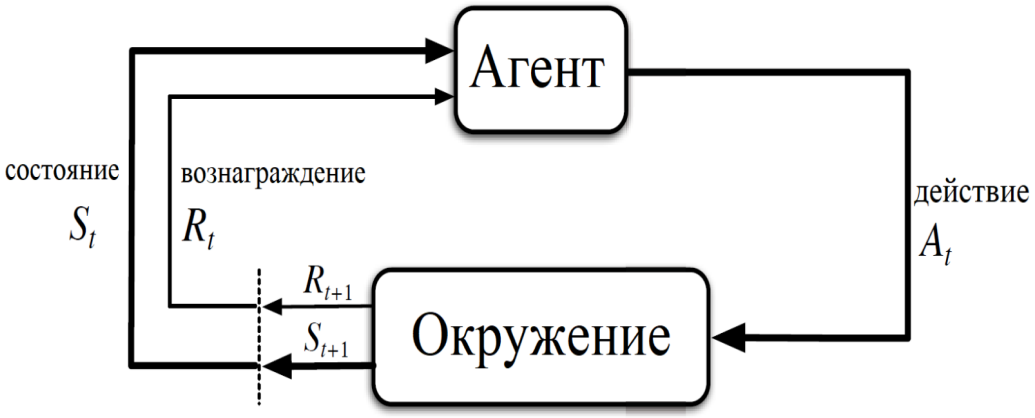

Рисунок 1.2 — Схема взаимодействия агента и среды в марковском процессе принятия решения [17].

Для поиска стратегии π, максимизирующей среднее суммарное вознаграждение, функционал для оптимизации задается как:

$$J(\pi) = \mathbb{E}_{\mathcal{T}\sim\pi} \sum_{t\geq 0} \mathcal{R}(\mathcal{T}) \to \max_{\pi}, \tag{1.2}$$

где коэффициент дисконтирования γ вводится в функцию вознаграждения $\mathcal{R}(\mathcal{T})$ для того что бы данный функционал не мог уходить в бесконечность или не существовать вообще. Это даёт приоритет получению вознаграждения в ближайшее время перед получением того же вознаграждения через некоторое время.

$$\mathcal{R}(\mathcal{T}) \coloneqq \sum_{t \geq 0} \gamma^t r_t \tag{1.3}$$

Функция полезности состояний (V функция) также может быть определена как показатель среднего дальнейшего вознаграждения агента, находящегося в состоянии S.

$$V^{\pi}(s) \coloneqq \mathbb{E}_{\mathcal{T} \sim \pi | s_0 = s} \mathcal{R}\left(\mathcal{T}\right) \tag{1.4}$$

Для одновременной оценки полезности состояния агента в текущий момент и предпринимаемого из него действия, вводят Q-функцию, которая определяется аналогично значениям V.

$$Q^{\pi}(s, a) \coloneqq \mathbb{E}_{\mathcal{T} \sim \pi | s_0 = s, a_0 = a} \mathcal{R}\left(\mathcal{T}\right) \tag{1.5}$$

Один из базовых алгоритмов поиска оптимальной стратегии в том случае, когда агенту известна вся информация о переходах среды, состоит в том, что происходит оценка полезности каждого состояния среды и использование этой информации для выбора оптимального действия в каждом состоянии. Так как наибольшую полезность агент имеет, следуя оптимальной стратегии $\pi*$, то истинное значение полезности состояния S определяется как $V^{\pi_*}(s)$.

С другой стороны, зная значения полезности состояний, можно использовать принцип максимальной ожидаемой полезности и выбирать действия, которые максимизируют ожидаемую полезность следующего состояния. Таким образом, полезность состояния находится через уравнение Беллмана как сумма текущего вознаграждения, полученного при совершении выбранного по стратегии π действия, и ожидаемой полезности очередного состояния, в которое агент попадает в зависимости от модели переходов $\mathcal{P}$:

$$V^{\pi}(s) = \mathbb{E}_a \left[r(s, a) + \gamma \mathbb{E}_{s'} V^{\pi}(s') \right] \tag{1.6}$$

Для Q-функции уравнение Беллмана записывается как:

$$Q^{\pi}(s,a) = r(s,a) + \gamma \mathbb{E}_{s'} \mathbb{E}_{a'} Q^{\pi}(s',a') \tag{1.7}$$

1.3.2 Алгоритмы обучения с подкреплением

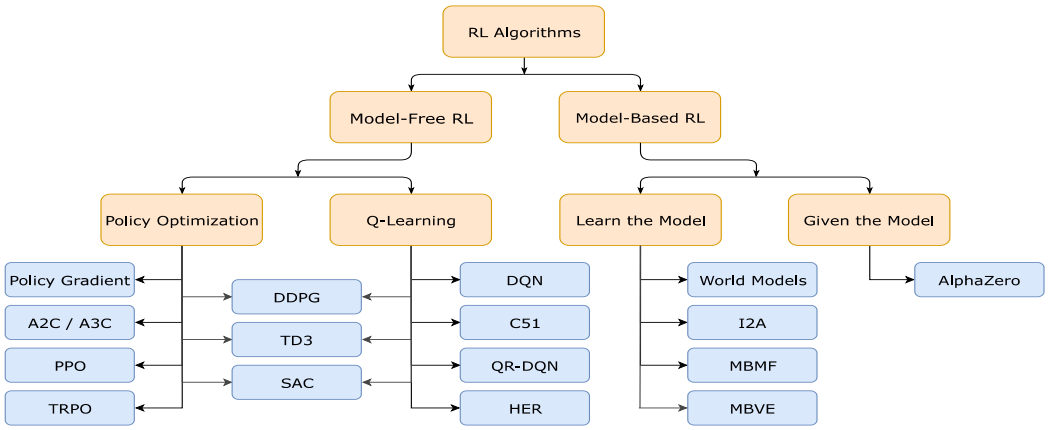

Рисунок 1.3 — Классификация алгоритмов с подкреплением. [49].

Составить полный список современных алгоритмов обучения с подкреплением довольно сложно, так же как и составить их всеобъемлющую систематику (Рис. 1.3). В данном разделе будут подчеркнуты наиболее фундаментальные принципы алгоритмов обучения с подкреплением и рассмотрены несколько современных подходов, успешно показавших себя в задачах навигации.

Одним из важных определяющих свойств алгоритма RL является наличие доступа к модели окружающей среды, если его нет то алгоритм относится к классу свободных от модели (model-free), иначе основанным на модели (model-based). Модель окружающей среды представляет из себя функцию, которая предсказывает переходы между состояниями агента и функцию вознаграждения.

Основным преимуществом наличия модели является то, что она позволяет агенту планировать, предвидеть, что произойдет при ряде возможных выборов, и явно выбирать между своими вариантами. Затем агенты могут обобщить результаты планирования в выученную стратегию. Особенно известным примером этого подхода является AlphaZero [50]. В 2015 усовершенствованная версия

данного алгоритма, AlphaGo [51], стала первым ИИ, обыгравшим профессионального игрока в Го.

Основным недостатком model-based подходов является то, что агенту обычно не доступна истинная модель окружающей среды. Если агент хочет использовать модель в этом случае, он должен выучить модель самостоятельно, исключительно на основе собранного опыта, что создает несколько проблем. Самой большой проблемой является то, что агент может собрать недостаточное количество опыта и переобучиться на нем, что приведет к тому, что агент в немного отличающихся условиях уже будет вести себя не оптимально, в том числе при переносе в реальную среду, где обычно присутствуют шумы. Также обучение модели по своей природе сложно, поэтому даже интенсивные усилия в этом направлении и готовность потратить много времени и вычислительных ресурсов могут не оправдаться.

Хотя безмодельные (model-free) методы отказываются от потенциальных преимуществ в эффективности использования собранных агентом данных, они обычно проще в реализации и настройке. Из-за этого методы свободные от модели стали более популярными и были более широко исследованы и протестированы, чем методы основанные на модели. В данной работе так же будут использоваться методы данного типа.

В безмодельных алгоритмах сбор данных становится важной составной частью: определяя стратегию взаимодействия со средой (behavior policy), происходит влияние на то, для каких состояний s, можно получить следующее состояние s' из функции переходов. Собираемые данные — траектории — алгоритм может запоминать, например, в памяти. Но не каждый алгоритм RL сможет пользоваться такими сохранёнными данными, и поэтому возникает ещё одна важная классификация RL алгоритмов: по текущему опыту (on-policy) и по отложенному опыту (off-policy).

Если алгоритм умеет улучшать свою стратегию, используя данные произвольной стратегии μ, то обучение проводится по отложенному опыту (off-policy). On-policy алгоритмам будет необходимо отправлять в среду конкретную стратегию, поскольку они будут способны улучшать стратегию π лишь по состояниям из неё же самой, что является существенным ограничением.

Важно, что off-policy алгоритм сможет на данных произвольного эксперта провести «полное» обучение, то есть условно сойтись к оптимуму при доста-

точном объёме и разнообразии экспертной информации, не потребовав вообще никакого дополнительного взаимодействия со средой. А если алгоритм может переиспользовать опыт, но с ограничениями (например, только с недавних итераций, или только из наилучших траекторий), то он всё равно будет относиться к on-policy, поскольку для каждой новой итерации алгоритма нужно будет снова собирать сколько-то данных из оптимизируемой стратегии.

Глубокое Q-Обучение

Глубокое Q-Обучение (DQN [52]) расширяет алгоритм табличного Q-Обучения, делая его применимым в средах с большим пространством состояний и действий, путем аппроксимации Q функции. DQN относится к семейству алгоритмов безмодельных (model-free) и по отложенному опыту (off-policy). Как и в случае табличного Q обучения, целью в DQN является обучение Q функции, которая возвращает ожидаемое вознаграждение от пары состояние-действие, $Q(s, a)$. В случае DQN, функция $Q(s, a)$ аппроксимируется глубокой нейронной сетью (NN), которая может быть многослойным персептроном (MLP) или сверточной нейронной сетью (CNN), в зависимости от представления состояния среды. Оптимизация глубокой нейронной сети с постоянно меняющимся распределением данных представляет собой несколько проблем, которые могут привести к катастрофическому забыванию и общей нестабильности во время обучения. Одним из ключевых способов исправить эти проблемы является большой буфер накопленного опыта, который обычно содержит более миллиона переходов состояний агента. Во время оптимизации, переходы равномерно выбираются из буфера повторения, так что для выполнения обновления параметров сети используется разнообразный собранный опыт. Функция потерь в DQN основана на правиле обновления таблицы в Q-Обучении:

$$\mathcal{L}(\theta) = [r + \gamma \max_a Q(s_{t+1},a;\theta) - Q(s_t,a_t;\theta)]^2 \tag{1.8}$$

Как и в случае табличного Q обучения, стратегия исследования является эпсилон-жадной (ϵ-greedy), с вероятностью ϵ, уменьшающейся со временем во время обучения. Недостатком метода DQN является то, что в целом они при-

менимы только в средах с дискретным пространством действий. Однако, это решает, основанный на DQN, алгоритм Soft Actor-Critic [53]. Тем не менее, алгоритм DQN был первым подходом Deep RL, который достиг уровня человека в среде Atari-57 и убедил исследователей в применимости Deep RL в областях с огромными пространствами состояний.

Алгоритм Актор-Критик

Альтернативным подходом к формированию стратегии агента является формирование ее в явном виде. Таким образом, нейросеть может отображать состояние агента сразу в распределение по возможным действиям. Такие алгоритмы в основном относятся к типу Актор-Критик (Actor-Critic), где актор отвечает за обучение непосредственно стратегии агента $\pi(s;\theta)$, а критик апроксимирует функцию вознаграждения $V(s;\theta)$. Одним из базовых алгоритмов данного вида является A2C (Advantage Actor-Critic, [54]). Данный алгоритм относится к виду model-free и on-policy. Функция преимущества (advantage) описывает насколько выгоднее выбрать действие a в состоянии s по сравнению с обычным поведением стратегии π в состоянии s. Определяется функция преимущества как разность между Q и V функциями: $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$.

На каждом шаге обучения A2C, для каждого из M количества параллельных сред, агент собирает набор траекторий длины N, используя стратегию π_θ. Затем, для каждой пары s_t,a_t вычисляется оценка Q-функции без учета зависимости от ϕ по формуле:

$$Q(s_t, a_t) \coloneqq \sum_{\hat{t}=t}^{N-1} \gamma^{\hat{t}-t} r_{\hat{t}} + \gamma^{N-t} V_{\phi}(s_N) \tag{1.9}$$

Функция потерь для критика при этом определяется как:

$$Loss^{critic}(\phi) \coloneqq \frac{1}{MN} \sum_{s_t,a_t} \left(Q(s_t, a_t) - V_{\phi}(s_t)\right)^2 \tag{1.10}$$

Далее делается шаг градиентного спуска по ϕ, используя $\nabla_\phi Loss^{critic}(\phi)$. Вычисление градиента для исполнителя при этом происходит по следующей формуле:

$$\nabla_\theta^{actor} := \frac{1}{MN} \sum_{s_t, a_t} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \left(Q(s_t, a_t) - V_\phi(s_t)\right) \tag{1.11}$$

По сравнению с алгоритмами, основанными на итерации по полезности (Value Iteration), такими как DQN, где авторы полностью полагались на этап оценки стратегии, то есть на обучение критика: модели исполнителя в явном виде даже не было в алгоритме, поскольку текущая стратегия всё время считалась жадной по отношению к текущему критику. Это означало, что качество стратегии упиралось в качество критика: пока Q-функция не научится адекватно приближать истинную Q^*, стратегия хорошей не будет.

Алгоритмы же основанные на policy gradient, предоставляет возможность обучать исполнителя напрямую, минуя промежуточный этап в виде критика: так, в алгоритме REINFORCE [17] используется Монте-Карло оценка Q-функции, что позволяет обходиться без обучения модели критика в явном виде. Таким образом, DQN и REINFORCE — это два «крайних случая», первый алгоритм полностью опирается на критика, а второй алгоритм — на исполнителя. Только у первого недостатки компенсируются возможностью обучаться в off-policy режиме и использовать буфер всех накопленных траекторий, а вот недостатки алгоритма REINFORCE — высокая дисперсия и необходимость играть целые эпизоды — не имеют аналогичного противовеса.

Важно, что в Policy Gradient алгоритмах при расчёте градиента, значение Q-функции может заменяться на оценку заглядывающую в будущее. И насколько сильно при обучении актёра опираться на критика — это и есть bias-variance trade-off, который в on-policy алгоритмах возможно разрешать. За счёт этого A2C в отличие от DQN может использовать и в качестве таргета для обучения критика, и в качестве оценки для обучения Q-функции GAE-оценку [17].

Алгоритм оптимизации ближайшей стратегии

Дальнейшим развитием on-policy алгоритма A2C является алгоритм оптимизации ближайшей стратегии (Proximal Policy Optimization, [48]). Главной целью при его создании было увеличить эффективность использования собранного опыта. Главный недостаток A2C подхода: неэффективность по объему взаимодействия со средой. Нам всё время нужны данные, сгенерированные при помощи текущей стратегии с параметрами θ, чтобы посчитать оценку градиента в точке θ и сделать шаг оптимизации. После этого будут нужны уже данные из новой стратегии, а старые удаляются. Решить данную проблему можно двумя способами: использованием методов оптимизации более высокого порядка, которые делают меньше итерации за счет их большей эффективности и вычислительной сложности, и использование данных, полученных от прошлых версий стратегий. Первый метод часто в результате оказывается слишком вычислительно затратным и сложно настраиваемым. Со вторым методом, полностью перейти в off-policy режим с сохранением преимуществ on-policy не получится, поэтому накладываются ограничения на стратегию сбора данных, она должна быть схожа с текущей стратегией, то есть, быть недавней версией актуальной стратегии. При использовании сильно отличающейся стратегии при сборе данных, проблема будет заключаться в оценке выражения $A^{\pi}(s,a)$. Если оценивать функцию преимущества (advantage) для одной стратегии по данным из другой, то такая оценка будет сильно смещенной.

Чтобы решить эту проблему, алгоритм PPO определяет зону близости, которая ограничивает размер обновлений, которые могут быть внесены в стратегию. Как только данные агента выходят за пределы зоны близости, их вклады в градиенты обновления обрезаются до нуля. PPO вдохновлен аналогичным алгоритмом, оптимизацией стратегии доверительного региона (TRPO, [55]), но его гораздо проще реализовать и он показывает сопоставимую эмпирическую производительность. PPO обеспечивает эту зону близости путем изменения функции потерь в A2C, на следующую:

$$L(s,a;\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a), clip\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)},1-\epsilon,1+\epsilon\right)A^{\pi_{\theta_k}}(s,a)\right) \tag{1.12}$$

Такая функция потерь ограничивает максимальное изменение в соотношении вероятностей текущей и прошлой стратегии через гиперпараметр ϵ, который обычно задается в пределах $10-20\%$. Получается, что когда преимущество отрицательно, выражение ограничивает переходы, где отношение вероятностей действий ниже $1-\epsilon$. Когда оно положительно, обратное верно, и целевая функция обрезается, когда отношение выше $1+\epsilon$. Это ограничивает обновления таким образом, чтобы они оставались в зоне вокруг прошлых параметров стратегии.

На данный момент, PPO является одним из самых часто применяемых алгоритмов в обучении с подкреплением, в частности, одна из его вариаций для задач навигации, DDPPO [15], была выбрана в качестве основы для далее предлагаемых в работе алгоритмов.

1.4 Методы иерархического обучения с подкреплением

Иерархическое обучение с подкреплением (HRL) является перспективным подходом для решения задач с очень большим пространством состояний и отсутствием немедленного подкрепляющего сигнала [56]. Иерархический подход позволяет разделить сложную задачу на множество подзадач. Для людей это является естественной процедурой [57]. Однако, остается до конца не изученным вопрос – как именно человеку удается находить соответствующую иерархическую структуру. Поиск хорошей декомпозиции на подзадачи часто творческая задача, решение которой представляет серьезную проблему. Несмотря на то, что в этом направлении получен ряд достижений [58], автоматическое построение иерархической структуры остается открытой проблемой в обучении с подкреплением.

Одной из первый работ в этой области является алгоритм выделения опций для представления иерархического поведения [17]. Пусть у нас есть множе-

ство опций G, то есть даны или несколько разных стратегий π_g, или универсальная стратегия $\pi(a \mid s, g)$. Эти стратегии, работающие на уровне примитивных действий, также называются рабочими (workers). Также имеется высокоуровневая стратегия, которая обычно называется менеджером (manager) $\overset{\star}{\pi}(g \mid s)$. Высокоуровневые действия также ещё называют макро-действиями, а примитивные действия — микро-действиями.

На очередном шаге менеджер, исходя из текущего состояния s_t, выбирает опцию, которая будет далее работать в среде: $g_t \sim \overset{\star}{\pi}(g_t \mid s_t)$. Выбранный рабочий генерирует примитивное действие: $a_t \sim \pi(a_t \mid s_t, g_t)$. Среда переходит в новое состояние $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$. И в этот момент вызывается стратегия терминальности опции g: с вероятностью $\beta_{t+1} \sim \text{Bernoulli}(\beta_g(s_{t+1}))$ рабочий завершает свою работу и снова передаёт решение менеджеру (тот снова выбирает следующего рабочего, и так далее). Если же стратегия терминальности не срабатывает, менеджер не вызывается, и взаимодействовать со средой продолжает рабочий π_{g_t}.

В рамках подхода с опциями, однако, нет гарантий что выучим какие-то разумные подзадачи; скорее рабочие как-то распределят между собой области среды, за которые будут ответственны. Однако, деля сложную задачу на последовательность более мелких, можно переставать думать об исходной задаче на уровне выполнения подзадач. Общая схема иерархического RL с двумя и более уровнями уровнями, к которой хотелось бы прийти, выглядит так. Определяется стратегия-менеджер и стратегии-рабочие для каждой подзадачи. Когда менеджер выбирает подзадачу g, запускается стратегия соответствующего рабочего π_g . Она делает несколько шагов в исходной среде, пока не решит задачу или не сдастся; критерий остановки процесса выполнения стратегии рабочего остаётся открытым. Далее менеджер снова принимает решение о следующей решаемой подзадаче, оптимизируя исходное вознаграждение, а рабочие стремятся решить свои подзадачи.

Строить такую идеальную схему, чтобы алгоритм сам смог выучить и набор подзадач, и механизмы определения моментов передачи управления менеджеру нетривиальная задача из-за возникающих нестабильностей. В теории опций авторы абстрагировались от идей подзадач и оптимизировали всю иерархическую конструкцию на максимизацию исходной функции вознаграждения.

1.4.1 Нестабильность обучения иерархических методов с подкреплением

Наиболее известные существующие алгоритмы HRL, которые могут изучать многоуровневые иерархии, не могут эффективно изучать уровни стратегий одновременно, особенно в непрерывном пространстве состояний и действий. Чтобы решить эту проблему, был представлен подход Hierarchical Actor-Critic (HAC, [23]).

Иерархии, созданные методом HAC, имеют архитектуру, состоящую из набора вложенных стратегий, зависимых от цели, которые используют пространство состояний как пространство действий для разложения задачи на набор подзадач. Одна из основных задач в обучении нескольких уровней стратегий заключается в нестационарных функциях перехода и вознаграждения. Начиная с верхнего уровня, каждый уровень будет предлагать действие для более нижней стратегии, которая имеет определенное количество попыток для его достижения. Таким образом, для любого действия верхнего уровня следующее состояние и вознаграждение зависит от ниже стоящих стратегий. В результате, когда все уровни в иерархии изучаются одновременно, функция перехода и функция вознаграждения для любого уровня выше базового могут продолжать меняться до тех пор, пока стратегии ниже этого уровня продолжают меняться.

В качестве примера нестационарной функции перехода можно рассмотреть робота на рисунке 1.4. Верхний уровень стратегии предлагает состояние B из текущего состояния робота A. Однако из-за не детерминированной стратегии ниже стоящего уровня робот будет оказываться в состояниях C,E,D. Из-за этого стратегия верхнего уровня будет считать действие B недостижимым и избегать его, когда как с улучшением ниже стоящих уровней оно может достигаться с большей точностью.

Так же будет нестационарной функция вознаграждения при одновременном обучении нескольких уровней иерархий. Рисунок 1.5 показывает пример где двухуровневая стратегия достигает одного состояния, но получает различные вознаграждения из-за разного пути к этому состоянию. Для этого случая в алгоритме предполагается, что функция вознаграждения равна -1 для любого действия, которое не достигает цели задачи, и 0 в противном случае.

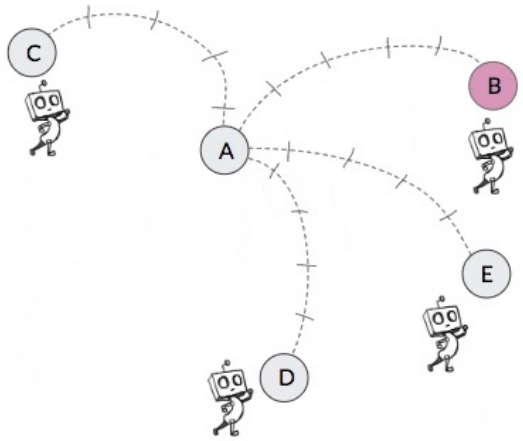

Рисунок 1.4 — Пример нестационарной функции перехода. Когда агент предлагает подцель состояния B, находясь в состоянии A, следующее состояние, которое зависит от этого действия, меняется со временем по мере изменения низкоуровневой стратегии.

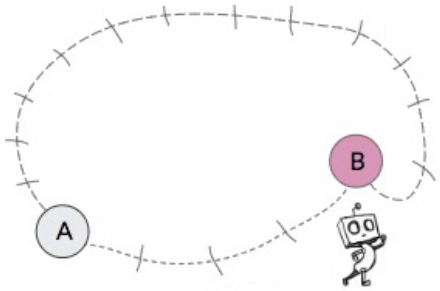

Рисунок 1.5 — Пример нестационарной функции вознаграждения. Хотя в обоих итерациях агент может достичь подцели B, низкоуровневая стратегия выбирает разные пути, поэтому одно и то же действие подцели может давать разные вознаграждения.

Нестационарные функции переходов и вознаграждений являются значительной проблемой, так как они затрудняют обучение стратегий на всех уровнях. Алгоритмы RL обычно оценивают ожидаемую долгосрочную стоимость действия a_t в состоянии s_t (т.е. $Q_{Target}(s_t,a_t)$) как сумму текущего вознаграждения r_{t+1} и дисконтированного значения текущей стратегии π в последующем состоянии s_{t+1} (т.е. $\gamma V_\pi(s_{t+1})$).

$$Q_{Target}(s_t,a_t) = r_{t+1} + \gamma V_\pi(s_{t+1}) \tag{1.13}$$

Если r_{t+1} и s_{t+1} не имеют стационарных распределений для одной и той же пары состояние-действие, Q значения не будут стабилизироваться, и будет трудно оценить отдельные действия, что, в свою очередь, затруднит обучение эффективной стратегии. Таким образом, чтобы иерархический агент мог параллельно изучить все свои стратегии и реализовать преимущества HRL, необходимо преодолеть нестационарные функции перехода и вознаграждения, которые возникают на всех уровнях выше базового уровня.

1.4.2 Алгоритм иерархического актора-критика

Ключевая проблема, описанная выше, заключается в том, что если все уровни иерархии должны обучаться параллельно, действия на любом уровне не могут быть оценены по отношению к текущей иерархии стратегий ниже этого уровня. Эта иерархия нижнего уровня будет продолжать меняться до тех пор, пока стратегия нижнего уровня будет учиться на собранном опыте и исследовать среду. Изменения в стратегии более низкого уровня, в свою очередь, вызовут нестационарные переходы и функции вознаграждения на более высоких уровнях, что затруднит обучение эффективной стратегии на этих более высоких уровнях. Центральная идея подхода, «Иерархический актор-критик» (HAC), заключается в том, что вместо оценки текущих действий по отношению к текущим стратегиям нижних уровней, действия оцениваются относительно оптимальной иерархии нижнего уровня. Оптимальная иерархия нижнего уровня, состоящая из оптимальных версий всех стратегий нижнего уровня, не меняется со временем. В результате, распределение последующих состояний и вознагра-

дений за любое растянутое во времени действие будет стабильным, что позволит иерархическому агенту параллельно изучать свои многочисленные уровни стратегий. Агенты, которые обучаются с помощью структуры HAC, могут обучать каждый небазовый уровень иерархии относительно оптимальных версий стратегий нижнего уровня без фактической необходимости в оптимальной иерархии стратегий нижнего уровня в результате двух основных компонентов структуры: (i) архитектуры иерархических стратегий, которую изучают агенты HAC, и (ii) ретроспективных переходов, которые агент использует для оценки действий. Пример подобных переходов схематично показан на (Рис. 1.6).

Переходы ретроспективных действий. Чтобы эффективно изучать несколько стратегий параллельно, небазовые уровни иерархии нуждаются в переходах, которые оценивают действия так, как будто стратегии более низкого уровня уже оптимальны. В этом заключается цель первого набора переходов HAC, которые называются переходами ретроспективных действий. Переходы ретроспективных действий реализуются с помощью простой процедуры: заменой предложенного действия действием, которое фактически было выполнено агентом. Для небазовых уровней агента, которые предлагают подцели, это означает, что всякий раз, когда уровень предлагает какое-то состояние подцели, но уровень ниже пропускает это состояние подцели и заканчивает в каком-то другом состоянии после H шагов, переход ретроспективного действия будет использовать состояние, в котором агент завершил как исходное подцелевое действие. Благодаря этому изменению, компоненты действия и следующего состояния в переходе будут одинаковыми, и, таким образом, ретроспективные переходы действий могут моделировать, как будет действовать оптимальная иерархия стратегий нижнего уровня. Кроме того, для компонента перехода вознаграждение будет зависеть только от состояния s_{t+1}, достигнутого после H попыток, и целевого состояния g для рассматриваемого уровня. Вознаграждение не будет учитывать точный путь к состоянию достигнутому после H попыток, поскольку цель состоит в том, чтобы создать переходы, моделирующие оптимальную иерархию стратегий нижнего уровня. Вознаграждение также будет разреженным, чтобы избежать проблем, возникающих при разработке функций вознаграждения вручную. В частности, вознаграждение за каждый уровень, $r_{t+1}(s_{t+1},g)$, будет равна 0 за любое действие, в котором $s_{t+1} \in g$, и -1 в противном случае.

Переходы ретроспективных целей. Хотя ретроспективные переходы действий устраняют проблемы нестационарных функций перехода и вознаграждения, они создают одну проблему — как изучить несколько уровней стратегий только с разреженными функциями вознаграждения. Структура HAC решает эту проблему, дополняя архитектуру вторым типом перехода. Переходы ретроспективных целей по своей сути расширяют идею ретроспективного воспроизведения опыта (HER) [59] на иерархические стратегии. HER помогает агентам научиться достигать различных целевых состояний в областях с редким вознаграждением, обобщая опыт агента по достижению других целевых состояний. HER реализуется путем создания копий исходных переходов [, , ,] и замены исходного целевого состояния состоянием, которое фактически было достигнуто во время эпизода. Исходное вознаграждение также заменяется соответствующим вознаграждением с учетом нового состояния цели.

Иерархия вложенных стратегий работает следующим образом. Стратегия наивысшего уровня принимает на вход текущее состояние и целевое состояние, предоставленное задачей, и выдает действие в виде подцели в пространстве целевого состояния. Это действие используется как целевое состояние для стратегии нижнего уровня. Стратегия на этом уровне принимает на вход текущее состояние и целевое состояние, предоставленное уровнем выше, и выдает свою собственную подцель для достижения следующего уровня. Этот процесс продолжается до тех пор, пока не будет достигнут самый низкий уровень. Затем самый низкий уровень принимает на вход текущее состояние и целевое состояние, предоставленное уровнем выше, и выдает примитивное действие. Кроме того, каждый уровень имеет определенное количество попыток достичь своего целевого состояния. Когда уровень либо исчерпывает попытки, либо достигает своего целевого состояния, выполнение на этом уровне прекращается и уровень выше выдает другую подцель. Цель метода HAC - эффективно изучить k-уровневую иерархию Π_{k-1}, состоящую из k отдельных стратегий π_0,....π_{k-1} где k - это гиперпараметр. Чтобы изучить π_0,....π_{k-1} параллельно, метод HAC преобразует универсальный исходный марковский процесс принятия решений (UMDP) $U_{original} = (S_i,G_i,A_i,T_i,R_i,\gamma_i)$.

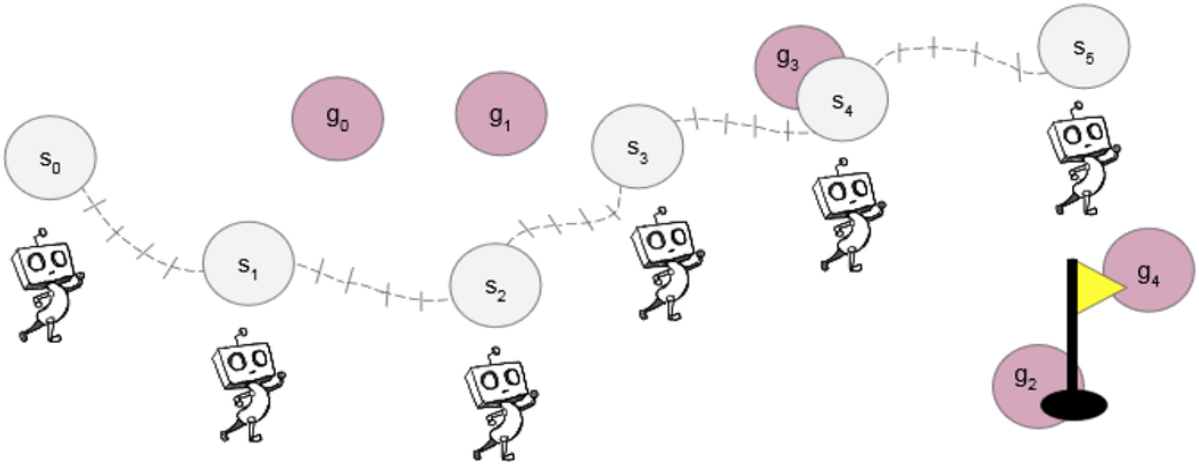

Рисунок 1.6 — Схематичное изображение hindsight переходов. Стратегия верхнего уровня из состояния агента s_i ставит подцель g_i на пути к целевому состоянию (желтый флаг). Из-за не оптимальной стратегии нижнего уровня, агент оказывается в состоянии s_{i+1}, которое при обучении стратегии верхнего уровня заменяет g_i, делая переход нижнего уровня оптимальным.

1.5 Состояние исследований

1.5.1 Навигация методами обучения с подкреплением

Большим прорывом в подходах с использованием обучения с подкреплением в навигационных задачах стал метод DDPPO [15], в основе которого лежит Proximal Policy Optimization [48]. Без каких-либо модулей отображения или планирования, DDPPO в задаче PointNav смог выполнить 2,5 миллиарда шагов в среде и решить задачу с производительностью на уровне человека. Для оптимизации процесса обучения авторы исследовали наиболее эффективную архитектуру нейронной сети [60] с использованием ResNet [61] и рекуррентными GRU [62] слоями, которая также была использована в предлагаемых в этой работе подходах.

Подходы, основанные на обучении с подкреплением, обычно аппроксимируют функцию полезности или прямое отображение состояния в действие с использованием алгоритмов на основе градиента стратегии [17]. Эти алгорит-

мы извлекают признаки в виде представления, релевантного для конкретной задачи, с использованием функций аппроксимации.

Один из подходов к получению более устойчивого и переносимого представления окружающей среды - это многозадачность. Этот подход хорошо подошел для предлагаемого метода SkillFusion [29] с несколькими навигационными навыками, поскольку он позволяет агенту отслеживать и запоминать различную информацию из одного и того же состояния [63]. Недавние успехи в многозадачности продемонстрировали способность одной нейронной сети выполнять широкий спектр задач [2]. Проведенные в данной работе эксперименты также показали, что совместное обучение нескольких навыков в ранней фазе слияния [64] дает лучшие результаты по сравнению с базовыми решениями [15].

Другой подход к улучшению представления - это разделение визуального кодирования состояния от непосредственной стратегии управления движением. Это можно достичь, обучая кодировщики на более крупных и разнообразных наборах данных, что приводит к менее специфичным для домена представлениям. Затем эти параметры кодировщика можно заморозить, что позволяет алгоритму RL обучать меньше параметров и сходиться быстрее [65] [66]. Использование предварительно обученных моделей, таких как CLIP [67], в качестве кодировщиков изображений в задачах навигации также оказалось эффективным [68] [69].

1.5.2 Навигация методами SLAM и планирования

Классический подход к навигации в ранее не виденных агентом окружающих средах предполагает декомпозицию задачи навигации на следующие подзадачи: локализация, картографирование, планирование пути и следование по пути. Задачи локализации и картирования часто решаются парно, используя подход одновременной локализации и картирования (SLAM). Большинство методов SLAM ([70], [71], [9]) извлекают признаки из изображений камеры или облаков lidar и отслеживают движение робота с использованием соответствия признаков. Один из самых популярных методов - RTAB-MAP [72]. Он спосо-

бен работать с различными датчиками (RGB-D камера, стереокамера, lidar) и строит как 2D карту занятости, так и 3D карту облака точек.

Для планирования пути, наиболее популярным подходом является использование алгоритмов поиска пути по графу, таких как A* [30] или Theta* [12], на карте занятости, построенной с помощью SLAM. Для следования по пути существует множество методов управления, которые обычно адаптированы к конкретной робототехнической платформе [31—33].

Еще одна важная задача, связанная с навигацией, - это исследование неизвестной среды. Главным направлением в классических методах исследования является использование границ [73; 74]. Методы на основе границ ищут границы между свободными и неисследованными ячейками на карте занятости и выбирают одну из этих границ в качестве цели. Удобная и эффективная реализация алгоритма исследования на основе границ представлена в работе [75].

1.5.3 Объединение классической и обучаемой навигации

Многие работы показали, что агенты, обученные на основе RL, уступают в избегании столкновений и управлении памятью при навигации, но превосходят в обработке неоднозначных ситуациях и лучше ориентируются в зашумленных условиях [76] [77]. Одно из направлений слияния обучаемых и необучаемых подходов - это разработка модулей навигации на основе обучения, которые могут быть интегрированы в единый пайплайн с классическими методами планирования. Например, SLAM и классическое планирование пути могут быть сформулированы в дифференциальной форме и обучены как одна система [78]. Для задач исследования среды и ObjectGoal, [14] и [13] продемонстрировали превосходство модульных подходов над полностью обучаемыми и чисто классическими. Авторы реализовали модуль глобальной стратегии на основе RL, который предсказывает для задачи подцель, а классическое локальное планирование выдает путь к этим подцелям. В предлагаемом подходе SkillFusion [29] навыки не делятся на только классический или обучаемый, а вместо этого реализуем каждый навык обоими подходами. Это позволяет агенту определять,

какой тип навыка должен быть выполнен в каждый момент времени на основе его функции полезности.

В методе PONI [79], алгоритм выбирает промежуточную цель на построенной карте с помощью потенциальной функции, предсказанной нейросетью. Потенциальная функция точки состоит из двух компонент - оценка площади, которую можно картировать с данной точки, и оценка вероятности наличия цели рядом с данной точкой. Для предсказания второй компоненты на карту наносятся объекты 16 типов - помимо целевых объектов, также стол, плита, раковина и т.д., и полученная 16-слойная карта подается на вход нейросети. Нейросеть, обученная на сотнях тысяч примеров подобных карт, оценивает вероятность наличия цели по взаимному расположению объектов на построенном участке карты. Промежуточная цель для агента выбирается как точка с наибольшим значением потенциальной функции. Для движения к промежуточной цели использовался алгоритм Fast Marching Method (FMM).

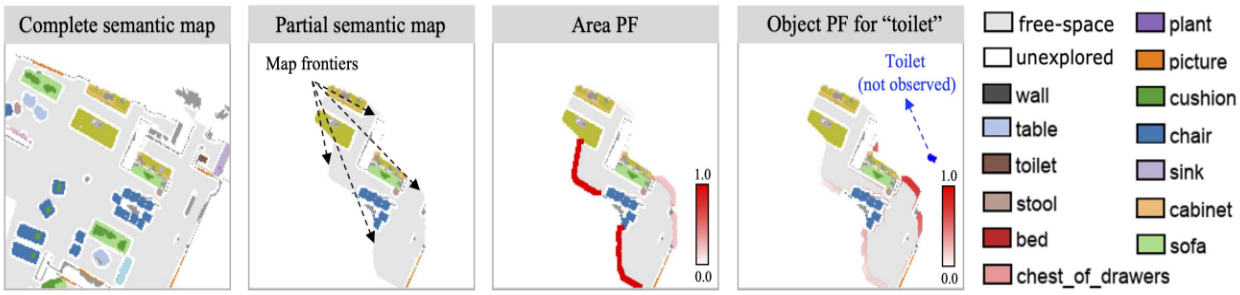

Рисунок 1.7 — Пример потенциальных функции в методе PONI, которые помогают найти целевой объект [79].

После того, как на карту был нанесен один или несколько целевых объектов, агент строил по карте путь до ближайшего из них с помощью метода поиска пути A* и двигался к цели по построенному пути. Так как истинные метки объектов агенту не давались, то объекты на входных изображениях размечались с помощью нейросетевой семантической сегментации, которая имеет значительный процент ошибки. Из-за ошибки нейросети, агент мог нанести на карту ложный целевой объект (например, в случае, когда нейросеть приняла край дивана за стул, или картину на стене за телевизор). По мере приближения к такой ложной цели, она становилась лучше видна камерой агента, и нейросеть переставала помечать ее как цель, однако она уже была нанесена на карту, и агент доезжал до нее и завершал эпизод с нулевым результатом.

Другая гибридная стратегия управления - это Байесовское слияние контроллеров (BCF) [80], которое сочетает стохастическую RL стратегию, учитывающую неопределенность, с алгоритмическим контроллером. Байесовская формулировка позволяет методу с наименьшей неопределенностью доминировать в управлении. В состояниях высокой неопределенности стратегии, BCF смещает композитное распределение действий в сторону более рискованного априорного распределения, уменьшая вероятность катастрофического сбоя. Однако, этот подход больше подходит для задач манипуляции с большим пространством действий, где отсутствие риска столкновения важнее, чем метрики задач. Предлагаемый подход SkillFusion [29], с другой стороны, при выборе навыка больше полагается на оценку будущего вознаграждения, а не на неопределенность.

Идея использования ориентиров в качестве области интереса в навигационных задачах также описана в [81] [26]. Первый метод, HIGL, также делил стратегию на высокоуровневую и низкоуровневую, где высокий уровень генерирует подцель в отношении ориентиров, а низкий уровень достигает ее. Выборка ориентиров была основана на критериях «охват» и «новизна» из ранее посещенных областей. В предлагаемом методе HLPO [28], был использован аналогичный подход, но вместо выборки из предыдущих траекторий (в постановке задачи агент должен ориентироваться в ранее невиданных сценах) была использована семантическая структура сцены и ориентиры определялись как координаты центра комнаты. Стратегия выбора ориентиров основывается на расстоянии до ориентиров и статистике объектов по типам помещений.

Применение обучения с подкреплением для задачи визуальной навигации в трехмерных симуляторах и на реальных роботах еще недостаточно широко изучено. В задаче навигации до точки агент должен быть устойчив к внешним шумам сенсоров и актуаторов и уметь разбивать задачу на подцели для планирования на более далекие горизонты. Для задачи поиска целевого объекта, агент должен быть способен полагаться как на явное представление окружающей среды в виде карт, так и полагаться только на визуальную информацию, когда осложнено выполнение задачи SLAM с необходимой точностью.

Глава 2. Навигация в реальном времени с иерархическим обучением с подкреплением

Навигация в реальном времени, планирование маршрута, локализация и избегание препятствий являются ключевыми задачами мобильных роботов [82].Часто эти задачи решаются методами, которые не используют машинное обучение в своей основе [83—85]. Такие подходы часто сталкиваются с ограниченной адаптивностью к различным условиям, в которых работает роботизированная платформа. С одной стороны, это обеспечивает относительно высокую производительность решений, но с другой стороны, требует знания динамических моделей движения робота и объектов окружающей среды, моделей шума в данных датчиков, а также наличия других априорных данных о среде. В реальных сценариях функционирования робота такие априорные данные могут отсутствовать. Поэтому важно, чтобы робот имел возможность обучаться в процессе взаимодействия со средой. Таким образом, робот может рассматриваться как интеллектуальный агент [86], поведение которого определяется архитектурой, включающей обучаемые модули.

В последние годы методы глубокого обучения и обучения с подкреплением (RL) привнесли значительные улучшения в методы навигации мобильных роботов. В данной главе рассматривается ключевая основанная на зрении задача в области мобильной робототехники - навигация внутри помещений с использованием RGB-D изображений [27] [24] [25]. Решение этой задачи предполагает анализ трех подзадач: семантическая сегментация объектов на сцене, локализация и картографирование, исследование сцены и планирование маршрута. Однако применение этих методов сталкивается с рядом сложностей из-за низкой вычислительной эффективности современных архитектур нейронных сетей и их ограниченной адаптивности к специфике роботизированных экспериментов.

В данной главе основное внимание уделяется критически важным визуальным задачам в области мобильной робототехники, таким как навигация внутри помещений к объекту с использованием данных с RGB-D камеры. Решение этой задачи предполагает анализ следующих подзадач:

1) семантическая сегментация целевого объекта,
2) оценка одометрии и локализация на карте,

3) автоматическое исследование и планирование пути.

В каждой из вышеуказанных задач применяются различные классические и нейросетевые методы, которые обеспечивают достаточное качество решений, однако их вычислительная эффективность может быть недостаточной для использования в энергоэффективных встраиваемых системах управления мобильными роботами. В данной главе представлены методы, способные работать в режиме реального времени и решать указанные проблемы. Эти методы интегрированы в единую обучаемую структуру управления поведением для мобильной роботизированной платформы, способной автоматически исследовать окружающую среду и перемещаться к объектам указанного класса. Общепринято, что качество решения напрямую зависит от качества симулятора, его возможностей рандомизации и обобщающей способности применяемых нейронных сетей.

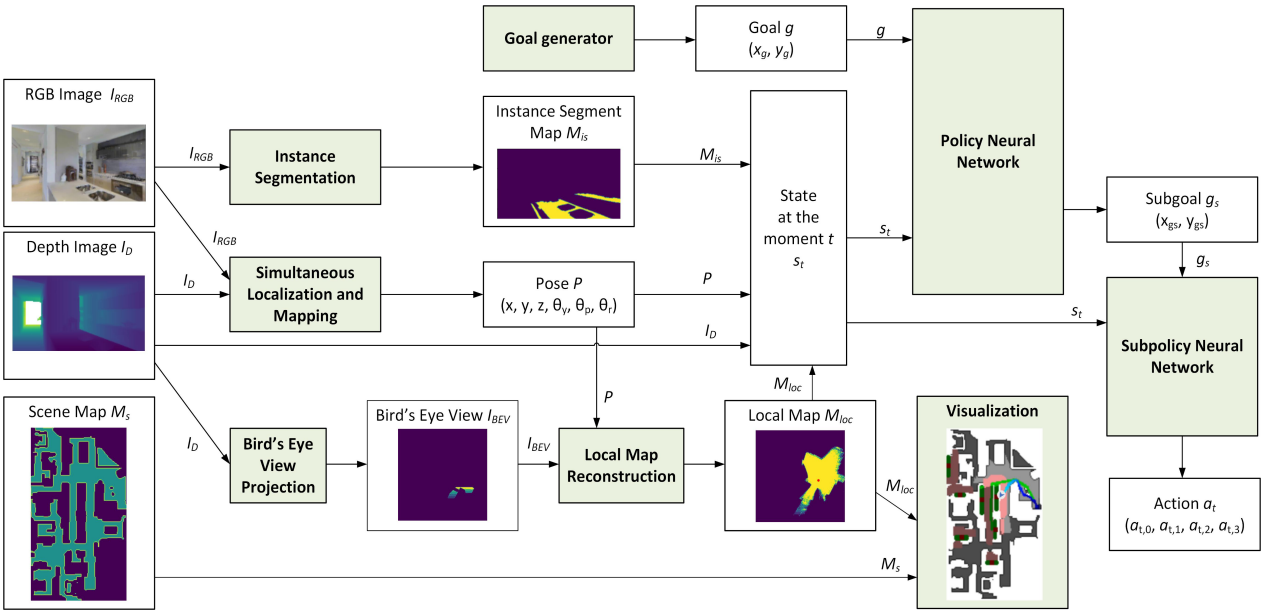

Рисунок 2.1 — Структура предлагаемой архитектуры HISNav для сегментации, SLAM и навигации.

Многие известные методы, применяемые для решения задачи навигации к объектам, обладают низкой устойчивостью к шуму. Это было особенно заметно в ходе международного соревнования Habitat Challenge 2020 [1] [87], где ни одному из участников не удалось эффективно справиться с одновременным шумом камер и актуаторов. Следовательно, разработка унифицированных нейросетевых архитектур управления роботами в режиме реального времени для решения задачи навигации к объектам, устойчивых к различным видам помех,

[1] https://aihabitat.org/challenge/2020/

остается актуальной и нерешенной проблемой с практической точки зрения. Разрабатываемая нами архитектура представляет собой шаг вперед в решении этой глобальной проблемы.

Основной вклад предложенного метода в решение данной проблемы заключается в следующем:

- Разработка новой нейросетевой архитектуры для управления роботом, которая обладает высокой скоростью работы и устойчивостью к возможному шуму в датчиках и исполнительных механизмах,
- Интеграция методов семантической сегментации, картирования, локализации и обучения с подкреплением для повышения эффективности исследования окружающей среды, поиска целевого объекта и быстрой навигации к нему,
- Внедрение нового подхода к управлению интеграцией данных в алгоритм одновременной локализации и картирования для повышения метрик SPL и доли успешных эпизодов,
- Создание нового набора данных HISNav, основанного на виртуальной среде Habitat и включающего изображения RGB-D, их метки сегментации экземпляров, позы камеры робота и действия по управлению роботом в различных сценах.

Предлагаемое решение основано на известной парадигме sim2real, когда нейронные сети предварительно обучаются с помощью данных из симулятора, а затем полученные модели используются на реальном роботе. Для этого были собрали уникальные наборы данных на основе доступной симуляционной среды Habitat, которые позволили нам построить эффективные модели для решения проблем сегментации объектов и локализации на карте. Предложенный метод HISNav и собранные для него наборы данных доступны публично по ссылке [2] под лицензией MIT.

В качестве среды, в которой функционирует робот, выбран фотореалистичный симулятор внутренних помещений Habitat с набором фотореалистичных 3D-сцен Matterport3D [43]. Структура предлагаемого подхода для сегментации экземпляров, одновременной локализации и картирования, а также навигации (HISNav Framework), представлена на Рис. 2.1.

[2] https://github.com/cds-mipt/HISNav

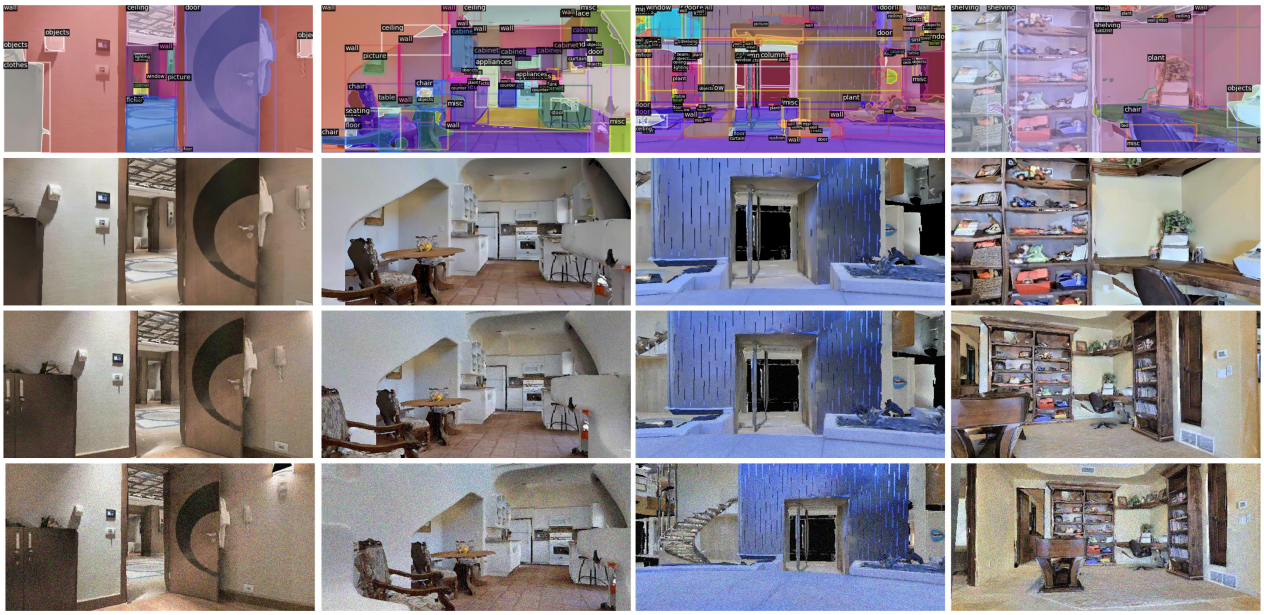

Рисунок 2.2 — Примеры изображений из набора данных HISNav с тремя уровнями шума: первый ряд содержит визуализацию истинной сегментации, второй ряд демонстрирует изображения без шума, третий ряд включает изображения с легким гауссовским шумом ($mean = 0, \sigma = 1, \ = 0{,}05$), нижний ряд содержит изображения с сильным гауссовским шумом ($mean = 0, \sigma = 1, \ = 0{,}1$)

2.1 Сбор данных в симуляторе для картирования и навигации

Для изучения качественных характеристик ключевых элементов подхода, таких как сегментация экземпляров и SLAM, с помощью симулятора был подготовлен специальный, достаточно разнообразный набор данных.

В данном разделе описывается предлагаемый автором набор данных HISNav Dataset, который состоит из различных траекторий движения робота, записанных в виртуальной среде Habitat. Траектории основаны на 49 уникальных сценах из Matterport3D [43], представляющих собой помещения различных стилей. Для каждой сцены предусмотрено не более 5 траекторий с тремя различными типами шума в изображениях камеры и шума в действиях.

Для исследования устойчивости разработанного нами подхода к воздействию шума, были проведены эксперименты с использованием трех типов искажений на изображениях: без шума, слабый Гауссовский шум ($mean = 0, \sigma =$

$1, intensity = 0.05$), сильный Гауссовский шум ($mean = 0, \sigma = 1, intensity = 0.1$). Примеры изображений из набора данных показаны на Рис. 2.2.

Вместе с шумом в изображениях использовался также шум в действиях робота: в условиях отсутствия шума в действиях, при наличии незначительного шума в действиях ($mean = 0, \sigma = 1, intensity = 0.2$), и сильном шуме в действиях ($mean = 0, \sigma = 1, intensity = 0.5$).

Источником данных для набора данных служат RGB-D изображения, которые включают информацию о цвете пикселей I_{RGB}, и плотную карту глубин наблюдаемой сцены I_D. Эти данные могут быть получены с использованием современных высокоточных RGB-D камер, которые широко применяются в области мобильной робототехники.

Каждое цветное изображение и данные с его глубиной имеют разрешение 640x320. Каждый пиксель изображения с глубиной содержит значение расстояния в метрах в диапазоне от 0 до 100 м. Каждому изображению соответствует набор данных с семантической разметкой объектов, включающий 40 классов (таких как стена, пол, стул, дверь, стол, диван и т.д.).

Для удобства исследования алгоритмов визуального SLAM, набор данных HISNav также включает в себя истинные положения камеры (ground truth camera poses) в формате TUM [88], а также информацию об управляющих командах, представленную в виде первых трех строк матриц управления $C_i, 1 \leq i \leq 4$, записанных построчно для каждого действия агента вместе с моментом совершения действия.

Весь набор данных состоит из 135962 изображений и разделен на три части: обучающую (train), валидационную (val) и тестовую (test) выборки. Детали разбиения представлены в Таб. 2. При создании набора данных было достигнуто достаточное разнообразие сцен, а также обеспечен баланс между обучающей, валидационной и тестовой выборками.

Таблица 2 — Подробности собранного набора данных HISNav

	HISNav-train	HISNav-val	HISNav-test	total
Number of images	72626	27952	35384	135962
Ratio, %	53,4	20,6	26,0	100
Number of unique scenes	49	35	43	
Number of tracks	88	35	43	
Number of instances per class (40 classes)				
wall	420782	171545	231628	823955
floor	181355	73785	94042	349182
chair	153759	54164	59711	267634
door	228804	83998	103098	415900
table	76787	33727	37339	147853
picture	80842	31947	39932	152721
cabinet	37649	18275	21674	77598
cushion	47449	25205	23442	96096
window	110639	35525	54558	200722
sofa	24254	10277	10999	45530
bed	19651	8117	10491	38259
curtain	45786	14236	16187	76209
chest of drawers	16311	5817	8023	30151
plant	36286	19104	27240	82630
sink	7002	3014	3275	13291
stairs	19337	6359	11485	37181
ceiling	139829	55721	81929	277479
toilet	1948	1219	1444	4611
stool	14721	6572	9056	30349
towel	6082	3201	4642	13925
mirror	14857	5987	6692	27536
tv monitor	13440	5357	8471	27268
shower	5987	1166	1726	8879
column	31210	11130	10720	53060
bathtub	3854	1465	1571	6890
counter	15917	6626	9077	31620
fireplace	7218	2337	1960	11515
lighting	33907	9153	15949	59009
beam	5514	1490	3183	10187
railing	23927	9405	12778	46110
shelving	22927	11515	14981	49423
blinds	2150	782	1061	3993
gym equipment	868	372	495	1735
seating	19521	4747	7178	31446
board panel	1498	67	398	1963
furniture	3140	967	1277	5384
appliances	7839	2691	3745	14275
clothes	586	561	634	1781
objects	113931	45902	59614	219447
misc	223810	87047	126577	437434

2.2 Метод HISNav для визуальной навигации

2.2.1 Семантическая сегментация

Описанный набор данных обладает рядом особенностей. Объекты в нем могут быть как небольшими, так и занимать значительную часть кадра. При этом требуется сегментировать как перемещаемые элементы интерьера (например, стул, стол, полотенце и т.д.), так и неподвижные, фоновые объекты (стена, пол, окно, лестница и т.д.). В таких условиях двухстадийные методы сегментации, которые сначала обнаруживают ограничивающие прямоугольники, а затем их сегментируют, могут работать неэффективно. Для проверки этого были рассмотрены различные архитектуры популярной модели MaskRCNN [89] с основной сетью с малым числом параметров, а также их современный аналог YOLACT++ [90]. Рассмотрение моделей типа DeepSnake [91], PolarMask [92], PolyTransform [93] для данной задачи нецелесообразно, поскольку они предназначены для уточнения выпуклых контуров сегментов без полостей. Однако в подготовленном наборе данных сегменты объектов могут содержать “дырки”. Наиболее перспективными кажутся легкие версии современных одностадийных полностью сверточных подходов к сегментации объектов Blendmask [7] и SOLOv2 [8]. Для сравнения качества целесообразно провести анализ получаемых метрик средней точности обнаружения объектов (mAP) [94] с различными порогами метрики коэффициента перекрытия окон (IoU).

2.2.2 Локализация и картирование

Для решения задачи локализации робота в пространстве с использованием RGB-D изображений в качестве базовых были выбраны два современных метода с открытым исходным кодом: OpenVSLAM [9] и DXSLAM [3] [95]. Оба эти метода относятся к косвенным разреженным подходам, которые оценивают

[3] https://github.com/ivipsourcecode/dxslam

позу камеры (робота) на основе найденных ключевых точек и их дескрипторов: в первом случае используется метод ORB, во втором - нейросетевой метод HF-Net [96]. Эти методы представляют собой улучшенную версию популярного метода визуальной одометрии ORB-SLAM2 [10].

Собственную позу P робота в среде необходимо определять как положение его центра x, y, z, так и ориентацию $\theta_y, \theta_p, \theta_r$ (углы yaw, pitch и roll соответственно). Определение позы возможно с помощью данных одометрии, получаемых на выходе метода SLAM, который использует на входе RGB-D-изображения (пары I_{RGB}, I_D).

Вычисление позиции текущего кадра относительно предыдущего включает в себя сопоставление ключевых точек и последующее решение задачи блочного уравнивания (bundle adjustment). Сопоставление ключевых точек осуществляется на основе их дескрипторов, и для ускорения этого процесса обычно используется метод “мешка слов” (bag of words). Кроме того, современные методы для дополнительного ускорения этапа сопоставления также используют модель движения.

При использовании модели движения метод предсказывает перемещение агента на основе предыдущих перемещений. В частности, строится матрица скорости V, а позиция следующего кадра P_{cw}^{i+1} оценивается на основе позы предыдущего кадра P_{cw}^{i} по следующей формуле: $P_{cw}^{i+1} = VP_{cw}^{i}$. После оценки P_{cw}^{i+1} производится проектирование ключевых точек предыдущего кадра на текущий и ищется их сопоставление с ключевыми точками текущего кадра на основе их расстояния на плоскости изображения.

При непрерывном и гладком движении робота использование модели движения оправдано, и сопоставление ключевых точек на основе расстояния на плоскости изображения происходит корректно. Однако, при резких дискретных движениях предположения модели движения неверны, и ее использование приводит к быстрой потере ключевых точек и существенному падению качества. В то же время, если перемещения агента в среде являются результатом совершения им определенных и известных действий, а также если известно, к каким перемещениям приводят эти действия, то они могут быть использованы в качестве модели движения.

Агент в среде Habitat может совершать 4 действия: продвинуться вперед на d метров, повернуться на $\pm\alpha$, а также завершить эпизод. Для этих действий

вводятся матрицы управления $C_i, 1 \leq i \leq 4$, соответствующие этим действиям и имеющие следующую форму::

$$C(\alpha, d) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.1}$$

Эта матрица задает перемещение вдоль оси z на d и поворот вокруг оси y на α. Тогда действие перемещения вперед задается матрицей $C_1 = C(0, d)$, повороты - матрицами $C_{2,3} = C(\pm\alpha, 0)$, а завершение эпизода - матрицей $C_0 = C(0, 0)$. Отметим, что несмотря на то, что действия агента известны, его перемещение зашумлено, и фактически является случайной матрицей $\hat{C} = C(\alpha + \xi_\alpha, d + \xi_d)$, где ξ_α и ξ_d - случайные величины, шумы в угле поворота и перемещения соответственно. Поэтому знание действий агента в среде не позволяет восстановить его перемещение без использования методов визуальной одометрии. Далее по матрице управления строится матрица скорости для использования в модели движения: $V = C_i^{-1}$.

Предлагаемый подход с заменой модели движения на модель управления позволяет интегрировать априорные знания о перемещении в методы SLAM и улучшает результаты визуальной локализации агента в среде.

Для исследования были выбраны следующие метрики SLAM:

1) Ошибки относительного смещения (T_{KITTI}, %) и вращения (R_{KITTI}, град/м), применяемые в тесте KITTI Odometry Benchmark [97], [98]. В связи с короткими внутренними траекториями в наборе данных HISNav, были использованы подпоследовательности с длиной (0,25, 0,5, 1, 2, 4, 8, 16, 20) метров вместо обычных (100 200,..., 800).

2) Для смещения (APE_T, м) и поворота (APE_R, град) также были использованы абсолютные ошибки позиции, введенные в работе [99].

Абсолютная ошибка позиции для смещения APE_T вычисляется как медиана массива ошибок E_T. Каждый элемент массива E_T вычисляется следующим образом:

$$E_T^i = ||P_{pred}^i - P_{GT}^i||_2, \tag{2.2}$$

где P^i_{GT} — истинное смещение для i-го кадра
$(x^i_{GT}, y^i_{GT}, z^i_{GT})$, P^i_{pred} — предсказанное смещение для i-го кадра $(x^i_{pred}, y^i_{pred}, z^i_{pred})$, $||.||_2$ — норма $L2$, соответствующая абсолютной ошибке между двумя точками.

Абсолютная ошибка положения для вращения APE_R вычисляется как медиана массива ошибок E_R.

Для каждого кадра i ошибка позиции P^i_R вычисляется следующим образом:

$$P^i_E = (P^i_{pred})^{-1} \cdot P^i_{GT}, \tag{2.3}$$

где P^i_{pred} — предсказываемая позиция (4×4) на кадре i, P^i_{GT} — истинная позиция (4×4) на кадре i.

Затем из каждой ошибки позиции (4×4) извлекается матрица вращения P^i_E (3×3). Каждая матрица вращения преобразуется в трехмерный вектор, который сонаправлен оси вращения Эйлера, а его норма определяет угол поворота (в радианах). Затем из каждого вектора вращения вычисляется норма. Норма полученного вектора поворота преобразуется из радиан в градусы и используется в качестве угловой ошибки E^i_R.

3) Процент потерянных траекторий, %. Эта метрика показывает, как часто метод SLAM терпит неудачу при отслеживании в начале или в конце траектории.

Метод SLAM считается неотслеживаемым, если выполняется хотя бы одно из следующих условий:

- , Если первая успешно предсказанная поза находится дальше, чем на 0,2 секунды от первой позы GT.
- , Если последняя предсказанная поза стоит дальше, чем на 0,2 секунды от последней позы GT.

2.2.3 Иерархический подход к задачи визуальной навигации

После решения задач сегментации объектов и локализации агента на карте, ключевой подзадачей является навигация к целевому объекту. Определение целевого объекта происходит путем указания целевого класса объекта (напри-

мер, стул, стол и т.д.). Предполагается, что обнаружение целевого объекта будет осуществляться путем его сегментации на цветном изображении с получением карты семантических экземпляров M_{is}.

Для этого был применен модульный иерархический подход, который позволяет разделить задачу навигации на планирование пути на коротком горизонте и определение подцелей на длинном горизонте. Дополнительным преимуществом предлагаемого подхода является отсутствие сложной функции вознаграждения в процессе обучения, которая обычно формируется вручную и не может полностью отразить все особенности среды.

Основная цель разрабатываемого метода — обучение стратегии Π_{k-1} k-го уровня. Где k был взят равным 2. Каждый уровень стратегии представляет из себя $\pi_i : S_i \times G_i \rightarrow A_i$. Чтобы изучить эти стратегии π_0,π_1, был использован универсальный марковский процесс принятия решений (UMDP) U_0,U_1, где $U_k = (S,G,A,R,\gamma)$, где γ – коэффициент дисконтирования. Пространство подцелей G состоит из координат. Пространство действий первого верхнего слоя A представляет собой координаты подцели, необходимые для достижения цели задачи. Стоит отметить, что подцель первого уровня определяется относительно текущего положения агента и ограничена определенной областью $Targ$ вокруг агента, чтобы быть достижимой в любой ситуации. Пространство действий второго нижнего уровня представляет собой действия, которые агент должен выполнить для достижения подцели, определенной первым уровнем. После того, как подцель установлена первым уровнем стратегии, второму уровню стратегии необходимо определить последовательность действи N^a_{max} для достижения заданной подцели. Если эта подцель была достигнута, агент получает вознаграждение R равное 0, в противном случае -1. Первый слой имеет максимум N^t_{max} попыток для достижения конечной цели.

Робот в момент времени t может выбирать действие a_t, принадлежащее множеству $A = \{a_{t,0},a_{t,1},a_{t,2},a_{t,3},\}$, где $a_{t,0}$ - оставаться на месте, $a_{t,1}$ - повернуться влево на угол Δ_α, $a_{t,2}$ - повернуться вправо на угол Δ_α, $a_{t,3}$ - сместиться прямо на расстояние Δ_d.

Подобный тип движения реализован у роботов, имеющих возможность поворота на месте, в частности роботов-пылесосов, роботов-курьеров и т.п. В качестве метрики используется SPL (успех, взвешенный по длине пути) (2.4).

$$SPL = \frac{1}{N}\sum_{i=1}^{N}\frac{l_i}{max\,(p_i,l_i)}, \tag{2.4}$$

где l_i — длина кратчайшего пути между начальной точкой и целевым объектом, наиболее близкого к точке, где заканчивается траектория агента. p_i — длина пути, пройденного агентом в эпизоде.

Нижний уровень стратегии за счет дискретного пространства действия основан на DDDQN[100], высший уровень стратегии имеет непрерывное пространство действия (координаты подцели) и основан на Twin Delayed DDPG (TD3) [101] алгоритме. Структура нейронной сети показана на Рис. 2.3.

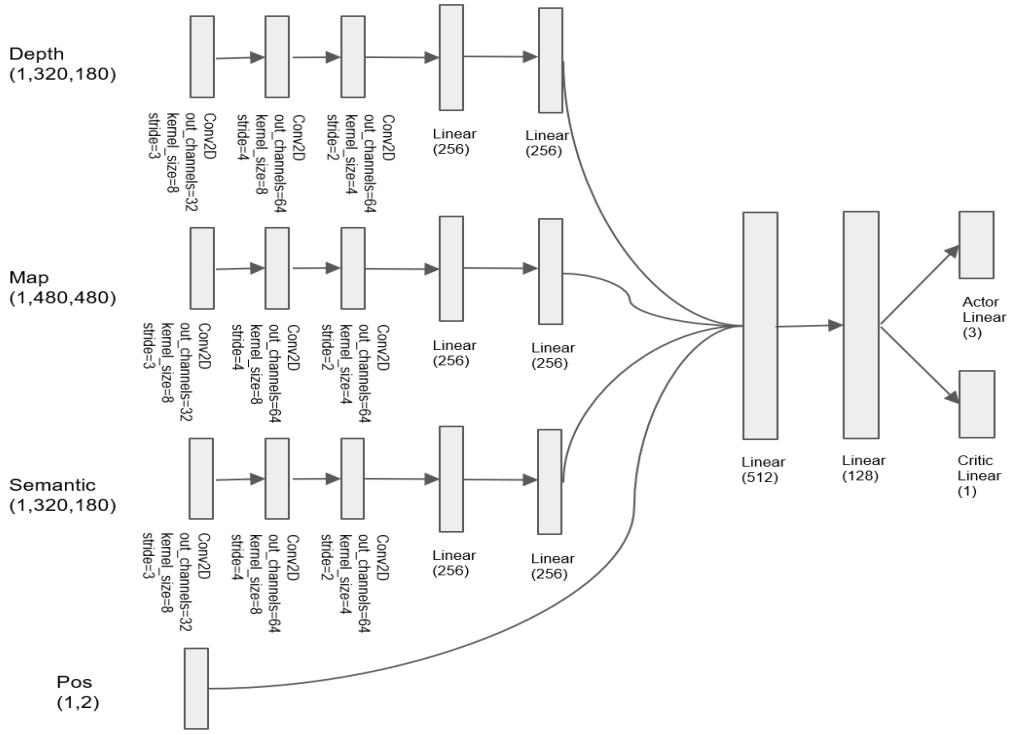

Рисунок 2.3 — Структура нейронной сети для аппроксимации стратегии агента в методе HISNav

Чтобы обучать все уровни стратегий одновременно, была использована основная идея иерархического актора-критика (Hierarchical Actor-Critic, [23]). Вместо того, чтобы оценивать действия по отношению к более низкому уровню стратегий, в данном методе оцениваются действия относительно того, куда в действительности пришел нижний уровень — оптимальная иерархия нижнего уровня. Оптимальная иерархия нижнего уровня, состоящая из оптимальных версий всех стратегий нижнего уровня, не меняется со временем. В результате

состояния и вознаграждение за любое действие будут стабильными, что позволит иерархическому агенту параллельно изучать несколько уровней стратегий.

2.3 Эксперименты

Аппаратная платформа. Для оценки эффективности предложенных подходов была использована следующая аппаратная платформа:

1) Серверная платформа, оснащенная графическим процессором NVidia Tesla V100 32Gb, процессором Intel Xeon Gold 6154 (16 ядер 3GHz), и оперативной памятью объемом 128Gb,
2) Наземная робототехническая платформа на базе шасси Clearpath Husky, оснащенная камерой ZED 1200x600 и бортовым компьютером с графическим процессором GTX1050Ti 4Gb, процессором Intel Core i5-4570TE (2 ядра 3.3GHz), и оперативной памятью объемом 8Gb.

Семантичесмкая сегментация. В ходе вычислительных экспериментов на обучающей выборке из набора данных HISNav, был обучен ряд современных моделей сегментации:

- SOLOv2 with ResNet34 backbone,
- Blendmask with DLA34 backbone,
- YOLOACT++ with ResNet50 and FPN backbone,
- Mask R-CNN implementation in Detectron2 [102] with Resnet50 and FPN,
- Mask R-CNN implementation in MMdetection [103] with the same backbone.

Оценка качества сегментации экземпляров в процессе обучения на валидационной выборке представлена на рисунке 2.4. На основании этих данных были выбраны наиболее эффективные модели по итогам обучения.

Используя отобранные модели, была составлена таблица 3, которая подробно описывает качество сегментации при различных пороговых значениях IoU на наборе данных HISNav-test.

На основании проведенных экспериментов можно сделать вывод о том, что наиболее высокие показатели качества демонстрирует сеть с архитектурой Blendmask, в то время как результаты сегментации объектов у модели SOLOv2

Таблица 3 — Качество сегментации экземпляров на наборе данных HISNav-test.

Model	mAP (IoU=0.50:0.95)	mAP (IoU=0.5)
Без шума на изображениях		
BlendMask	**0,2408**	0,3648
SOLOv2	0,2391	**0,3778**
YOLACT++	0,1691	0,2739
Mask R-CNN$_{det2}$	0,2198	0,34834
Mask R-CNN$_{mmdet}$	0,2267	0,3593
Со слабым шумом на изображениях		
BlendMask	**0,2317**	0,3564
SOLOv2	0,2311	**0,3703**
YOLACT++	0,1631	0,2655
Mask R-CNN$_{det2}$	0,2097	0,3377
Mask R-CNN$_{mmdet}$	0,2174	0,3501
С сильным шумом на изображениях		
BlendMask	**0,2175**	0,3427
SOLOv2	0,2154	**0,3527**
YOLACT++	0,1513	0,2526
Mask R-CNN$_{det2}$	0,1953	0,3204
Mask R-CNN$_{mmdet}$	0,2026	0,3363

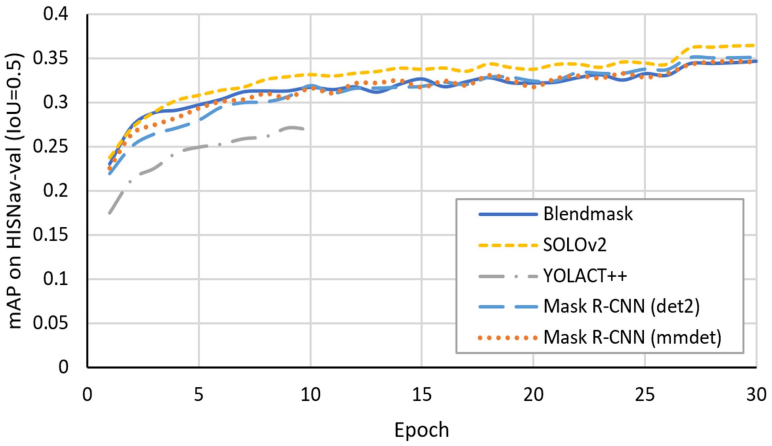

Рисунок 2.4 — Результаты обучения моделей сегментации экземпляров.

немного хуже. Именно эти две модели могут быть рассмотрены как базовые для интеграции в предлагаемый метод HISNav.

Рисунок 2.5 — Примеры сегментации экземпляров с использованием различных моделей нейронных сетей на изображениях из набора данных HISNav-test.

Визуализация сегментации экземпляров изображения для всех обученных нейронных сетей представлена на рисунке 2.5. Этот рисунок демонстрирует сравнение различных моделей с истинной сегментацией. Модель SOLOv2 кажется более точной, чем другие, что подтверждается ее качественными показателями.

Локализация и картирование. Таблица 4 демонстрирует, что предложенный метод DXSLAM (именуемый CDXSLAM) обеспечивает превосходную

Таблица 4 — Результаты валидации методов SLAM на данных с различным режимом движения.

Metrics	OpenVSLAM			CDXSLAM		
	motion	bow	control	motion	bow	control
Эксперименты на данных с быстрым движением ($\Delta_\alpha = 10°$ and $\Delta_d = 25cm$)						
T_{KITTI}, %	14.59	12.78	12.55	23.21	13.91	**12.40**
R_{KITTI},°/m	5.06	4.73	4.54	7.53	4.45	**3.77**
APE_T, m	0.79	0.72	**0.67**	1.13	0.79	0.77
APE_R,°	11.35	10.42	**9.71**	12.24	10.40	10.45
Lost, %	48.1	30.2	**27.1**	82.2	65.9	46.5
Эксперименты на данных с медленным движением ($\Delta_\alpha = 5°$ and $\Delta_d = 10cm$)						
T_{KITTI}, %	15.99	13.08	13.28	34.63	13.66	**11.48**
R_{KITTI},°/m	4.80	4.49	4.43	16.54	4.18	**3.41**
APE_T, m	0.77	0.77	**0.70**	1.42	0.78	0.78
APE_R,°	10.84	11.36	10.77	27.84	10.79	**9.53**
Lost, %	38.8	**24.8**	27.1	89.9	45.7	28.9

Таблица 5 — Результаты валидации методов SLAM на данных с различным шумом.

Metrics	OpenVSLAM			CDXSLAM		
	motion	bow	control	motion	bow	control
Эксперименты на данных без шума						
T_{KITTI}, %	15.16	13.89	14.15	23.12	15.20	**11.90**
R_{KITTI},°/m	4.94	4.63	4.59	8.65	4.54	**3.57**
APE_T, m	0.74	0.72	**0.68**	0.89	0.74	0.78
APE_R,°	11.37	10.52	**9.95**	14.18	10.51	9.96
Lost, %	28.2	17.6	15.3	59.3	27.9	**15.2**
Эксперименты со слабым Гауссовским шумом (mean = 0, $\sigma = 1, intensity = 0.05$)						
T_{KITTI}, %	16.14	13.15	**11.29**	175.05	12.10	–
R_{KITTI},°/m	4.96	4.75	4.13	85.42	**3.97**	–
APE_T, m	0.86	0.76	**0.68**	8.09	0.82	–
APE_R,°	11.29	10.98	**9.44**	147.33	10.57	–
Lost, %	37.2	23.2	**17.4**	98.8	55.8	100.0
Эксперименты с сильным Гауссовским шумом (mean = 0, $\sigma = 1, intensity = 0{,}1$)						
T_{KITTI}, %	14.34	**11.33**	14.52	–	11.88	–
R_{KITTI},°/m	4.80	4.39	4.94	–	**3.98**	–
APE_T, m	0.70	0.75	**0.66**	–	0.89	–
APE_R,°	**10.12**	11.34	12.23	–	11.36	–
Lost, %	64.0	**40.7**	50.0	100.0	83.7	100.0

производительность с точки зрения показателей T_{KITTI} и R_{KITTI}. Можно сделать вывод, что интеграция данных в DXSLAM на основе нейронных сетей и в OpenVSLAM приводит к значительному увеличению относительных и абсолютных показателей качества. Визуализация одного из треков из набора данных HISNav представлена на рисунке 2.7.

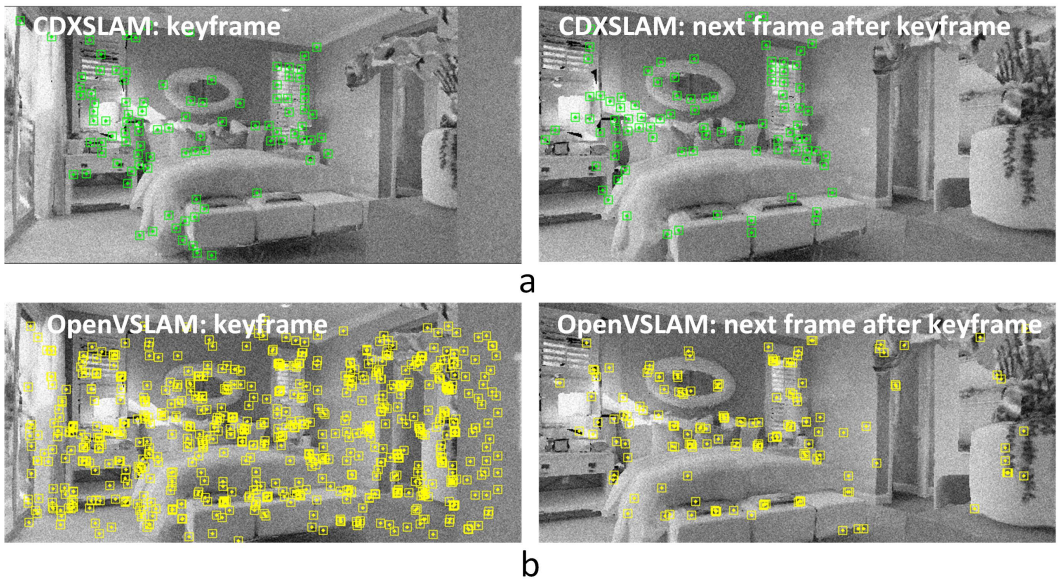

Рисунок 2.6 — Детали обнаружения и сопоставления ключевых точек в зашумленных изображениях: а - для метода CDXSLAM, б - для метода OpenVSLAM.

Рисунок 2.6 иллюстрирует эффективность обнаружения ключевых точек с использованием рассмотренных методов SLAM. Как видно на рисунке, на ключевом и последующем кадре при применении нейросетевого CDXSLAM отсутствуют шумовые точки. OpenVSLAM добавляет множество шумовых точек в кадр, которые не совпадают со следующим кадром.

Согласно таблице 5, нейросетевой метод CDXSLAM часто теряет позицию агента на траектории в условиях зашумленных датчиков. Возможная причина заключается в том, что нейросетевые ключевые точки менее универсальны, и их дескрипторы менее надежны, чем сформированные методом ORB. Базовая нейронная сеть HF-Net была обучена на датасете с реальными сценами OpenLORIS [104], которые не содержали такого типа шумов, влияние которого анализируется в данной статье.

Результаты экспериментов CDXSLAM и OpenVSLAM на роботизированной платформе в условиях университета представлены на рисунке 2.8. Главное

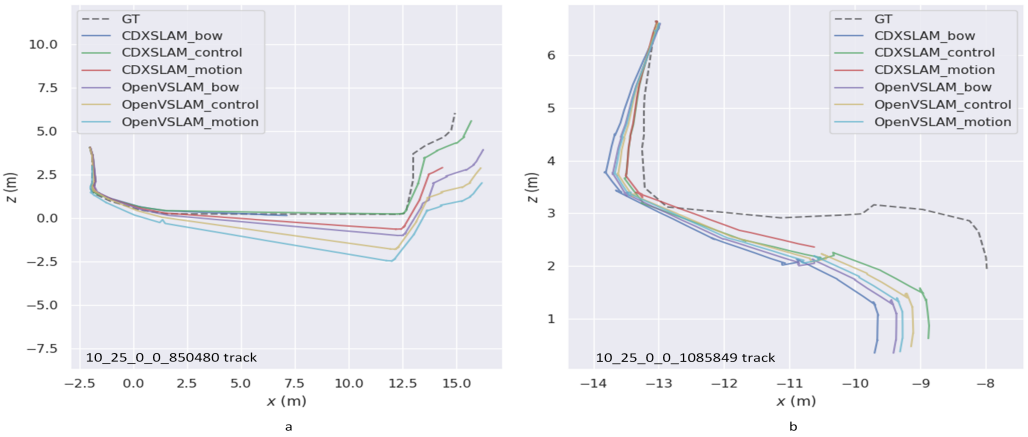

Рисунок 2.7 — Результаты изученных методов CDXSLAM и OpenVSLAM на некоторых траекториях набора данных HISNav. Интеграция в модель движения данных управления повышает качество обоих методов.

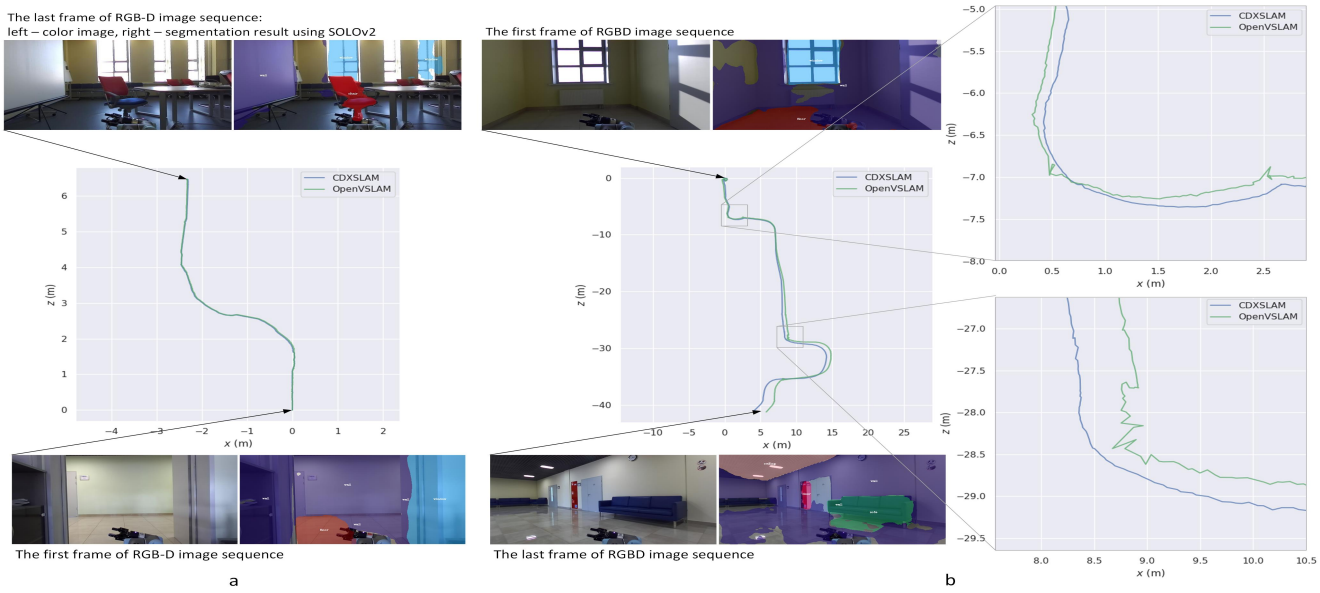

Рисунок 2.8 — Результаты SLAM на роботизированной платформе.

преимущество CDXSLAM заключается в том, что его траектория более плавная (см. рисунок 2.8 б). Это является важным параметром для таких приложений, как реконструкция карты.

Обучение с подкреплением в задаче навигации. Методы глубокого обучения с подкреплением часто сталкиваются с проблемой недостатка вознаграждения за исследование, что замедляет процесс обучения на больших сценах. Один из подходов, решающих эту проблему, — это метод случайной дистилляции сети (Random Network Distillation, [105]), который был выбран для сравнения с HISNav методом. В общем случае, задача навигации к целевому объекту сводится к двум подзадачам: навигации к точке и исследованию сцены.

Предлагаемая архитектура успешно справляется с задачей навигации к точке, а для решения задачи исследования был использован модуль глобальной стратегии SemExp [106]. Для ускорения процесса обучения, во всех приведенных ниже экспериментах была использована одна сцена с несколькими эпизодами, имеющими различные начальные и конечные точки.

В следующем эксперименте, приводится сравнение PPO, RND и HISNav методов в задаче навигации до точки с наличием Гауссова шума сенсоров ($mean = 0$, $\sigma = 1$, $intensity = 0.05$) и Гауссова шума в действиях агента ($mean = 0$, $\sigma = 1$, $intensity = 0{,}2$).

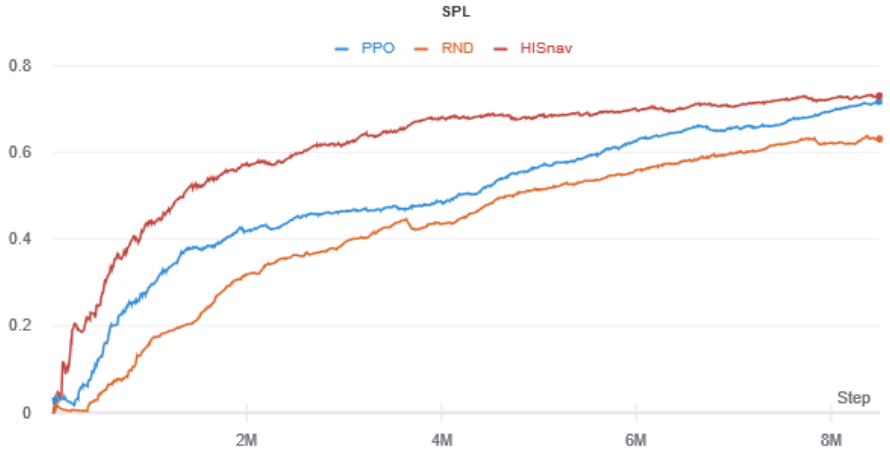

Рисунок 2.9 — PPO vs RND vs HISNav в задаче навигации к точке.

Как показано на рисунке 2.9, метод RND демонстрирует результаты, которые немного уступают PPO. Внутреннее вознаграждение RND не обеспечивает прироста в начале процесса обучения, и этот прирост снижается до минимума к концу. Предлагаемый метод в конечном итоге сходится к значению SPL, равному 0,7, для PPO и RND, однако он достигает порога в SPL 0,6 в два раза быстрее, на 3M, а не на 6M.

2.4 Выводы

В данной работе предлагается новый подход HISNav для решения задачи навигации внутри помещений с использованием RGB-D камеры в условиях шума, который может быть вызван сенсорами и актуаторами робота.

Предлагаемый подход реализует парадигму sim2real [107], в соответствии с которой, модули предварительно обучаются с использованием симуляционной среды, а затем полученные модели переносятся на реального робота.

Предлагаемый подход включает современные нейросетевые методы сегментации объектов, локализации и картирования, которые позволяют решать свои подзадачи в режиме реального времени. Для ускорения исследования среды и поиска объектов был разработан новый иерархический метод обучения с подкреплением с автоматической генерацией подцелей, что позволяет нам избежать создания сложной функции вознаграждения.

Для предварительного обучения всех подсистем был разработан новый датасет HISNav с использованием фотореалистичной среды Habitat. На основе этого датасета были обучены модели сегментации, локализации и стратегии агента, которые проявляют устойчивость к внешним шумам.

Экспериментальные исследования показали перспективность HISNav подхода. С необходимыми доработками было продемонстрировано, что современные архитектуры нейронных сетей и методы обучения с подкреплением подходят для решения важных задач робототехники в режиме реального времени. Однако следует отметить, что необходимы дальнейшие исследования в этом направлении. Несмотря на то, что были продемонстрированы преимущества предлагаемой архитектуры перед существующими обучаемыми методами решения задачи навигации, с точки зрения работы и скорости обучения, качество решения должно быть значительно улучшено.

Существуют методы, которые менее устойчивы к шуму и менее вычислительно эффективны, но в некоторых случаях позволяют сократить пройденное расстояние при поиске объектов. Увеличение показателя SPL и необходимая для этого доработка предлагаемого подхода — это направление будущих исследований. В ходе данной работы было определено, что критическим моментом здесь должна быть более глубокая интеграция методов RL и SLAM, что было продемонстрировано, путем интеграции модели динамики в алгоритм локализации.

Глава 3. Визуальная навигация с использованием ориентиров

В данной главе рассматривается проблема визуальной навигации внутри помещения к объекту, определенному его семантической категорией. С появлением высокопроизводительных симуляторов [4] [108] [1], больших фотореалистичных наборов 3D-данных [44] [43] [109] и благодаря последним достижениям в области изучения обучаемых методов, многие исследования продемонстрировали значительные успехи в сквозном (end-to-end) подходе к обучению с подкреплением и модульных системах, объединяющих обучаемые и классические подходы. Применение обучаемых методов позволяет агенту адаптироваться практически к любым внешним условиям в симуляторе. Однако недостатком методов RL является то, что их необходимо специально обучать каждой постановке задачи с нуля, и эти алгоритмы после обучения сложно развернуть в реальном сценарии навигации [110].

В ходе соревнования Habitat 2021 [1] в задаче навигации к объекту (ObjectNav), алгоритм сквозного обучения с подкреплением (RL) зарекомендовал себя как передовое решение, продемонстрировав 23% эффективности [18]. Это свидетельствует о том, что задача ObjectNav еще далека от окончательного решения или находится в стадии стагнации. Дальнейшее увеличение показателей метрики невозможно, поскольку нерешенные эпизоды представляют собой обширные области, исследование которых без предварительной информации о их семантической структуре представляется затруднительным. Предполагается, что такая информация может быть легко аннотирована человеком и она должна оставаться неизменной в областях с часто перемещающимися объектами. Следовательно, агент должен обладать иерархической структурой для эффективного использования пространственного понимания. Пример работы предложенного алгоритма представлен на рисунке 3.1. Чтобы решить проблему недостаточного исследования сцен и сделать исследование более семантически значимым, предлагается расширить стандартную постановку задачи и дать агенту легкодоступные ориентиры в виде расположения комнат и их тип. Наличие ориентиров позволяет агенту построить иерархическую структуру стратегии и добиться успеха в 63% на валидационных сценах в фотореалистичном

[1] https://aihabitat.org/challenge/2021/

симуляторе Habitat. В используемой иерархии, низкий уровень состоит из отдельно обучаемых навыков, а высокий уровень решает, какой навык необходим в данный момент. Также в данной главе показывается возможность переноса обученного алгоритма на реального робота. После небольшого дообучения на реконструированной реальной сцене, робот показывает до 79% SPL при решении задачи навигации к произвольному объекту.

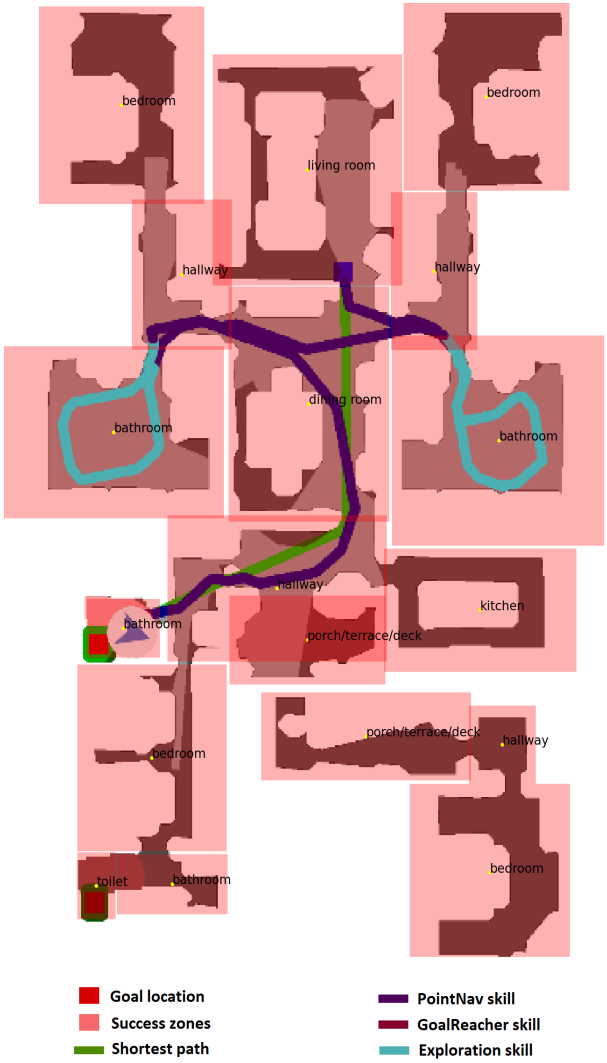

Рисунок 3.1 — Пример полученной траектории движения агента с визуализацией ориентиров.

В результате, данный подход включает в себя следующие аспекты:

– **Постановка задачи с ориентирами**. Когда люди решают задачи по исследованию помещений, чтобы найти объект, они в значительной степени полагаются на понимание взаимного расположения комнат конкретной сцены и могут предсказать тип объектов находящихся внутри. Чтобы агент мог обучиться этой концепции, агенту были даны ориенти-

ры ($\mathcal{G}$) в виде координат центра всех комнат и их тип. Эта информация может быть легко аннотирована человеком, в отличие от карты. Стоит обратить внимание, что карта препятствий или объекты внутри любых комнат по-прежнему неизвестны агенту.

– **Разделение стратегии агента на набор навыков**. В качестве стратегии агента были выделили три основных навыка: навигация к заданным координатам, исследование близлежащей местности и достижение увиденного объекта. Агент был отдельно обучен всем этим навыкам с использованием подхода оптимизации стратегии.
– **Иерархическая структура стратегии агента**. Чтобы эффективно использовать данный список ориентиров, был реализован модуль выбора навыков, который каждый раз направляет агента в наиболее перспективную зону ориентиров, выбирая один из представленных навыков.
– **Плавный перенос стратегии на новые реальные сцены**. Чтобы сделать трансфер стратегии возможным и предсказуемым, сначала была реконструирована сцена в 3D симулятор Habitat. После небольшого этапа адаптации к нему нейросетей и убедившись, что агент в нем безопасно перемещается, были проведены реальные тесты и было получено поведение агента, похожее на поведение в симуляторе. Поскольку реальный робот использует только RGB-изображение, глубина и семантика были получены нейросетевыми методами.

3.1 Метод иерархической стратегии с ориентирами

Рассматриваемая задача навигации определяется как задача навигации к объекту (указанному семантической меткой) в ранее невидимой среде [111]. На практике, агент инициализируется в произвольной точке в среде, и стремится найти экземпляр категории объектов $I_{goal} \in \{c_1,c_2,...,c_{20}\}$ (например, *диван*), перейдя к нему.

В процессе выполнения задачи агент может использовать только входные данные с камеры RGB-D (I_{RGBD}), датчика GPS+Компас (I_{GPS}) и списка ориентиров ($\mathcal{G}$) для навигации. Список ориентиров содержит все центральные коор-

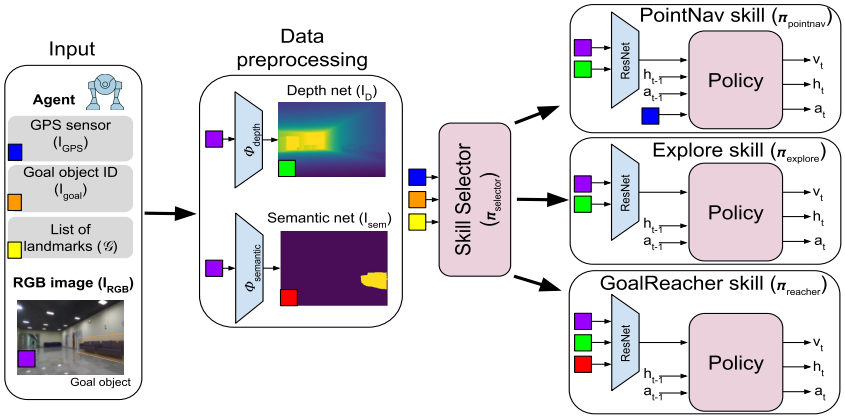

Рисунок 3.2 — Схема метода иерархической стратегии с ориентирами (HLPO). Предлагаемый нами подход состоит из трех основных блоков: предварительная обработка данных, селектор навыков и стратегии навыков. Разноцветные квадраты внизу элементов означают, какие данные отдают модули на выходе (внизу слева) и принимают в качестве входа (внизу справа).

динаты комнат и их тип, без информации о карте или какие объекты находятся внутри или как добраться до этих комнат.

Подсчет вознаграждения происходит, когда агент выбирает действие *stop*. В качестве метрики используются показатель успеха, взвешенный по длине пути (SPL), и показатель успеха (Success). SPL вычисляется для экземпляра объекта, ближайшего к начальному местоположению агента.

Таким образом, если агент появляется очень близко к *chair1*, но останавливается на удаленном *chair2*, он достигает 100% успеха (потому что он нашел “кресло”), но довольно низкий уровень SPL (потому что путь к *chair2* намного длиннее, чем к *chair1*). В частности, эпизод считается успешным, если агент вызывает действие *stop* в пределах евклидова расстояния 1,0 м от любого экземпляра категории целевого объекта, и агент может увидеть объект из этого положения остановки, повернув камеру вверх/вниз/влево/вправо.

В данной главе предлагается основанная на ориентирах модульная структура (рис. 3.2) для перехода к цели (I_{goal}) – Иерархическая оптимизация ориентиров (HLPO). Предлагаемый подход состоит из трех основных модулей: модуль выбора навыков $\pi_{selector}$, модуль предварительной обработки данных $\Phi_{semantic}$ и Φ_{depth} и стратегии навыков $\{\pi_{explore}, \pi_{reacher}, \pi_{pointnav}\}$.

Модуль выбора навыков анализирует все тренировочные сцены, связывая типы объектов с типами комнат, и составляет статистику. Для сбора дан-

Algorithm 1 Псевдокод валидации иерархической стратегии с ориентирами

Given:
$\pi_{selector}$: Skill selector,
$\pi_{explore}$: Explore policy,
$\pi_{reacher}$: GoalReacher policy,
$\pi_{pointnav}$: PointNav policy,
$\Phi_{semantic}$: Semantic segmentation model,
Φ_{depth} : Depth model,

Input:
I_{RGBD} : RGBD image from the camera,
I_{GPS} : X,Y coordinates relative to the start point,
I_{goal} Goal type of object,
$\mathcal{G}$: List of landmarks.

while episode episode not ended **do**
    $I_{skill\_type}, I_{room\_cord} \leftarrow \pi_{selector}(\mathcal{G}, I_{goal}, I_{GPS})$
    $I_{sem} \leftarrow \Phi_{semantic}(I_{RGB}, I_{goal})$
    $I_D \leftarrow \Phi_{depth}(I_{RGB})$
    if $I_{skill\_type} = PointNav$ **then**
        $a \leftarrow \pi_{pointnav}(I_{RGBD}, I_{GPS}, I_{room\_cord}, a_{prev})$
    end if
    if $I_{skill\_type} = Explore$ **then**
        $a \leftarrow \pi_{explore}(I_{RGBD}, a_{prev})$
    end if
    if $I_{skill\_type} = GoalReacher$ **then**
        $a \leftarrow \pi_{reacher}(I_{RGBD}, I_{sem}, a_{prev})$
    end if
    Execute action a in the environment
end while

ной статистики, информация о том, какие объекты принадлежат к какому типу комнаты была собрана со всех сцен набора данных Matterport. Основываясь на этой статистике, текущем типе цели объекта и расстояниях до комнат в текущий момент, модуль селектора умений ранжирует комнаты (как вероятность нахождения объекта в комнате/расстояние до комнаты) в порядке необходимости посещения для поиска целевого объекта.

Модуль предварительной обработки данных представляет собой две нейронные сети, выполняющие семантическую сегментацию и реконструкцию глубины.

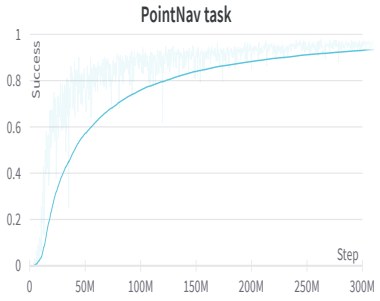

Рисунок 3.3 — Доля успешных выполненных эпизодов при выполнении навыка PointNav.

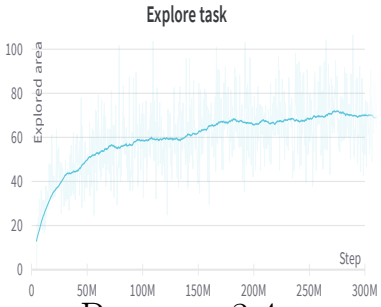

Рисунок 3.4 — Исследованная область (m^2) при выполнении навыка Exploration.

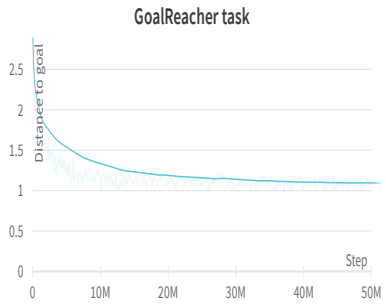

Рисунок 3.5 — Расстояние до целевого объекта (m) при выполнении навыка GoalReacher.

Модуль стратегий навыков принимает в качестве входных данных наблюдение агента и выводит действие, направленное на получение текущего необходимого навыка. Для сформулированной задачи были определены три навыка: навигация до заданных координат (PointNav), исследование сцены (Exploration) и достижение цели (GoalReacher). В проведенных экспериментах, навык PointNav выполнялся в 50% случаев во время эпизода, навык Exploration — в 22%, а навык GoalReacher — в 28%.

Навык PointNav обучался для достижения ориентира (комнаты представляющий наибольший интерес исходя из типа цели) (Рис. 3.3). Вознаграждение давалось пропорционально сокращенному расстоянию до цели. Стратегия принимает изображение RGBD и координаты относительно ориентира в качестве входных данных.

Навык GoalReacher был обучен навигироваться к объекту на заданное расстояние, когда его видит семантический сенсор (Рис. 3.4). Чтобы избежать

переобучения, агент во время инициализации эпизода появлялся в случайном месте, а затем ему задавалась цель достичь самого дальнего видимого объекта. Вознаграждение также пропорционально сокращению расстояния до ближайшего объекта целевого типа.

Навык Exploration больше других определяет успех задачи, особенно когда у агента нет ориентиров. В данной работе, для этого навыка агента была обучена стратегия на основе обучения с подкреплением, которая эффективно исследует близлежащую область (Рис. 3.5), поэтому она идеально подходит для полного исследования комнаты, но имеет более низкий процент охвата в больших сценах. В качестве вознаграждения агент получает +1 каждый раз, когда входит в новую зону размером в один квадратный метр.

Процесс управления навыками можно рассматривать как стратегию и реализовать как модуль глобальной стратегии. Как правило, методы HRL могут обучаться двухуровневой стратегии Π_1. Каждый уровень стратегии изучает $\pi_i : S_i,G_i \rightarrow A_i$, где G_i — множество возможных подцелей. Для изучения этих стратегий π_i используется множество MDP U_0,U_1, в которых $U_k = (S,G,A,T,R,\gamma)$. Для ObjectGoal задачи последовательность навыков может быть сформулирована в явном виде. Агент перемещается к первому ориентиру (координате интересующей комнаты, заданной глобальной стратегией) с помощью навыка PointNav. Затем агент исследует комнату, пока не покинет ее с помощью навыка Exploration. Если модель семантической сегментации видит целевой тип объекта в интересующей комнате, активируется навык GoalReacher и осуществляется переход к этому объекту.

Цель агента — найти стратегию π, которая максимизирует ожидаемое вознаграждение $\mathbb{E}_\pi[R_0 \mid s_0])$. Ожидание принимается в отношении начального распределения состояний, стратегии и переходов среды в соответствии с динамикой, указанной выше. Функция полезности состояния и действия (Q-функция) данной стратегии π определяется как $Q^\pi(s_t,a_t) = \mathbb{E}_\pi[R_t \mid s_t,a_t]$. Функция полезности состояния (V-функция) определяется как $V^\pi(s_t) = \mathbb{E}_\pi[R_t \mid s_t]$. Преимущество определяется как $A^\pi(s_t,a_t) = Q^\pi(s_t,a_t) - V^\pi(s_t)$ и сообщает, лучше ли действие a_t, чем среднее действие, которое стратегия π предпринимает в состоянии s_t.

В действительности, стратегия агента определяется в форме нейронной сети с двумя выходами. Один выход используется для определения распреде-

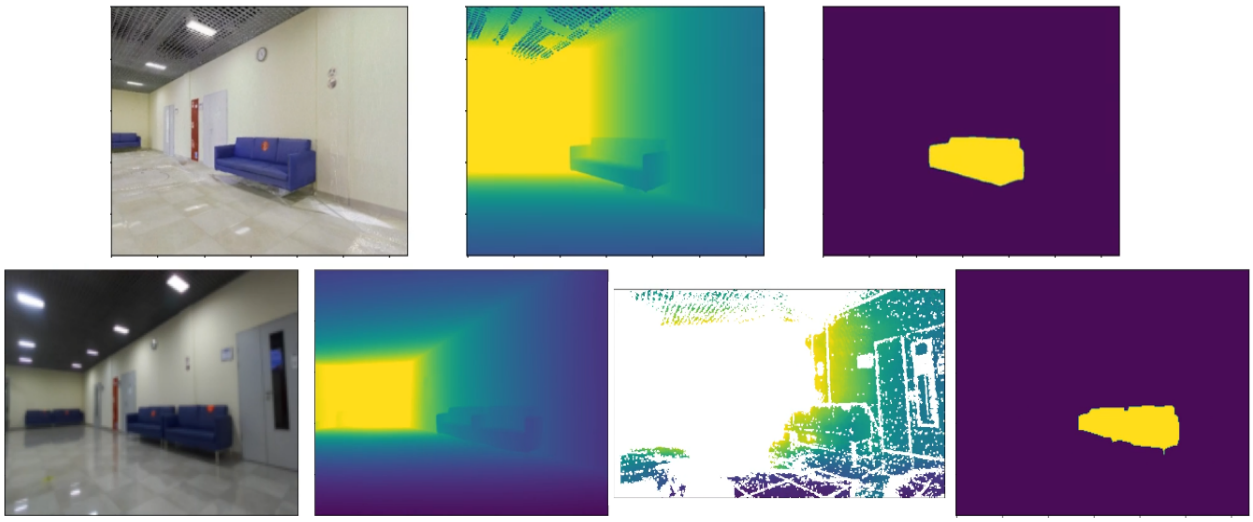

Рисунок 3.6 — Верхний ряд — данные, подаваемые на вход агенту в симуляторе. Изображение по центру верхнего ряда — это реальная глубина, которая была ограничена пятью метрами. Нижний ряд — то что видит агент в реальности. Второе изображение нижнего ряда — это глубина, полученная модулем реконструкции глубины. На третьем изображении нижнего ряда показано сравнение качества глубины нейронной сети с глубиной камеры ZED. Оба правых изображения представляют собой семантическую маску класса дивана, полученную модулем семантической реконструкции.

ления действий (актор) π, в то время как другой аппроксимирует функцию полезности текущей стратегии (критик) $V \approx V^{\pi}$.

Актор представляет собой полносвязный слой (FC), который генерирует логиты для каждого из четырех возможных действий. Выбор действия осуществляется путем применения категориального распределения к этим логитам. Критик, с другой стороны, это полносвязный слой, который генерирует оценку для данного состояния.

Для расчета вознаграждения используется обобщенная оценка преимуществ (GAE) с $\gamma = 0{,}99$ и $\tau = 0{,}95$.

В качестве метода решения для стратегии навыков, был применен метод оптимизации ближайшей стратегии (PPO) [48], который до настоящего времени остается передовым решением в обучении с подкреплением (RL) для большого количества задач. При наличии стратегии π_θ, параметризованной θ, и набора траекторий, собранных с помощью этой стратегии, агент обновляет π_θ следующим образом. Пусть $\hat{A}_t = R_t - \hat{V}_t$ - оценка преимущества, где $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, а $\hat{V}_t$ — ожидаемое значение R_t, и $r_t(\theta)$ — отношение вероятности действия в

соответствии с текущей стратегией и стратегией, используемой для сбора траекторий. Затем параметры обновляются путем максимизации выражения:

$$J^{PPO}(\theta) = E_t \left[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t) \right] \tag{3.1}$$

.

В качестве метода параллелизации был применен метод децентрализованной распределенной проксимальной оптимизации стратегии (DD-PPO) [15]. DD-PPO распределен (использует несколько машин), децентрализован (не имеет централизованного сервера) и синхронен (никакие вычисления не устаревают), что делает его концептуально простым и легким для реализации. Отличие синхронного метода от асинхронного заключается в том, что в синхронном методе каждый рабочий процесс собирает опыт из окружения и одновременно обновляет свои веса для общей модели со всеми рабочими процессами. Рабочие процессы, которые занимают слишком много времени для выполнения своей задачи, прерываются, чтобы остальные не ждали их. В общем виде этот метод реализует следующую абстракцию: на шаге k рабочий процесс n имеет копию параметров, θ_n^k, вычисляет градиент, $\partial\theta_n^k$ и обновляет θ следующим образом:

$$\theta_n^{k+1} = PU\left(\theta_n^k, AR(\nabla_\theta J^{PPO}(\theta_1^k), ..., \nabla_\theta J^{PPO}(\theta_N^k))\right) \tag{3.2}$$

где PU — любой метод оптимизации первого порядка (градиентный спуск), а AR выполняет редукцию (усреднение) по всем копиям переменной и возвращает результат всем рабочим процессам.

3.2 Перенос обученной стратегии на реального робота

Существует несколько способов обучения робота задаче навигации: от обучения в реальных сценариях до обучения в полностью смоделированных средах с последующим переносом полученной стратегии на реального робота. Первый слишком неэффективен, так как требует много ресурсов и может принести много вреда, пока стратегия не оптимальна. С последним удобно работать, и можно получить высокие результаты в симуляционных средах, но общая задача — сде-

лать так, что бы агент мог ориентироваться в реальных сценах. В этом случае процесс перехода от смоделированной среды к реальному роботу может быть даже более сложным, чем само обучение, так как в симуляторе агент не сталкивается с многими недостатками датчиков реального мира.

В качестве промежуточного подхода используется фотореалистичная среда, где сцена воссоздается с использованием лидаров и камер.

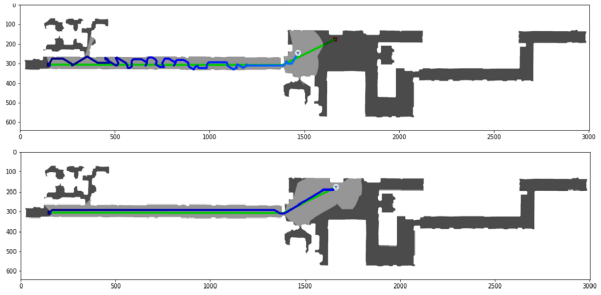

Рисунок 3.7 — Сравнение агента HLPO до и после адаптации.

Рисунок 3.8 — На нижнем изображении показано, как выглядит облако точек до нанесения текстур. Верхнее изображение это облако точек после нанесения текстур в программе RealityCapture.

Агент был обучен на максимально возможном количестве сцен, полученных из набора данных Matterport. Однако разнообразие сцен во всех наборах данных настолько велико, что 60 сцен недостаточно для учета всех характеристик любой сцены. Это и произошло в случае проводимых экспериментов в лабораторной зоне. Длинные одинаковые коридоры практически отсутствовали в обучающем наборе данных, и поведение агента внутри них значительно отклонялось от оптимального пути (Рис. 3.7). Поскольку дополнить набор данных большим количеством подобных сцен затруднительно, было решено перенастроить агента на реконструированную сцену перед проведением реальных тестов. Данный подход является подходящим для внутренней навигации в подобных случаях, поскольку такая калибровка требуется один раз для каждого типа

сцены, и дальнейший пользователь системы может загрузить соответствующие веса для своего типа сцены.

Создатели исходного набора данных mp3d [43] использовали камеру Matterport Pro2 (134 мегапикселя без лидара) и фирменное программное обеспечение Matterport, которое позволяет загружать фотографии для автоматической обработки в окончательный файл .obj. Этот файл затем можно использовать в симуляторе Habitat. Этот подход работает хорошо, если сцена не содержит мелких деталей. Однако в самом наборе данных mp3d было много пробелов и несоответствий текстур. В данной работе было стремление к более высокому качеству сцены, особенно к более точной реконструкции глубины, чтобы агент мог предсказуемо ориентироваться в узком пространстве комнаты. Предлагаемое решение заключается в использовании лазерного сканера Leica RTC360 3D. Дополнительным преимуществом является возможность предварительного редактирования полученных кадров, удаления проходящих людей и проверки качества сборки всей сцены. Для нанесения текстур на конечное облако точек была использована программа RealityCapture [2] (Рис. 3.8).

Рисунок 3.9 — Вероятность успеха во время адаптации сцены.

Кроме того, реконструированная сцена позволила откалибровать нейронную сеть по восстановлению глубины (Рис. 3.6) и установить правильные ограничения. Таким образом, глубина, полученная нейронной сетью, получилась очень близка к глубине, полученной из симулятора. Затем на полученной сцене агент был дообучен в течении одного миллиона шагов (Рис. 3.9)). Благодаря этому были учтены все особенности сцены.

После адаптации метода HLPO под особенности валидационной сцены, вероятность успеха выполнения эпизода в реконструированной сцене увеличилась с 0,6 до 0,9. А на реальном тесте с роботом Clearpath Husky был получен

[2]https://www.capturingreality.com/

SPL равный 0,79, что говорит о том, что предлагаемый метод HLPO одинаково хорошо работает как на симуляторе, так и в реальных тестах.

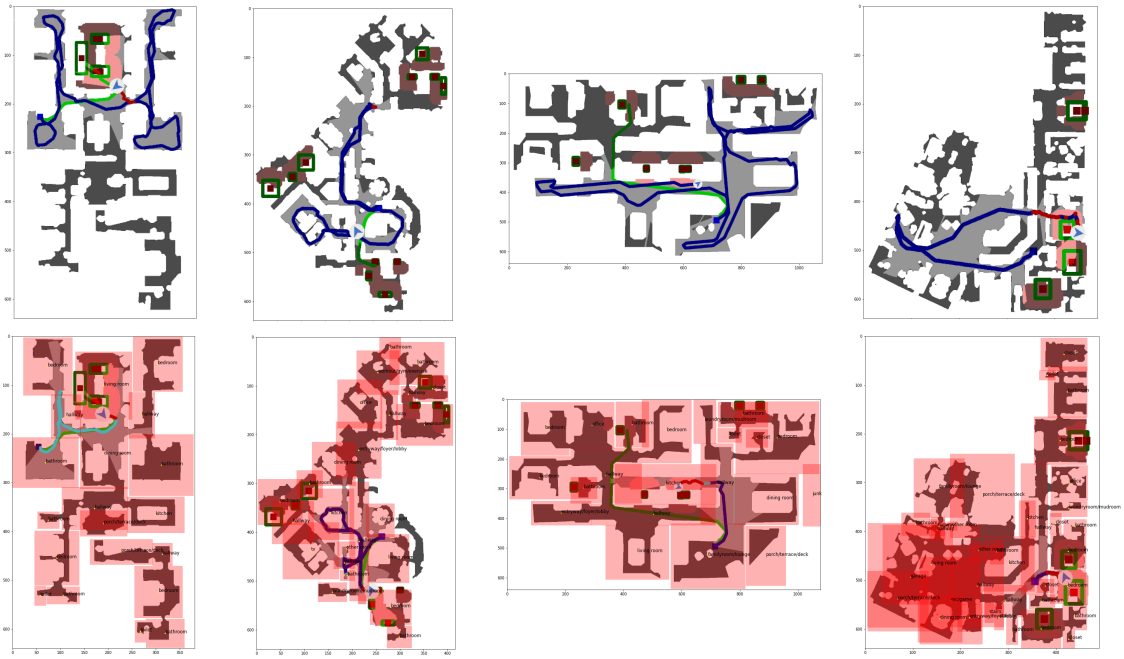

Рисунок 3.10 — Сравнение ExploreTillSeen (верхний ряд) и агента HLPO (нижний ряд)

3.3 Эксперименты

Для демонстрации эффективности предлагаемого метода HLPO в симуляторе Habitat были выделены два набора данных: для обучения агента и для тестирования. Для тестового набора данных были вручную отобраны 100 эпизодов из 11 сцен, которые достаточно сложны и обширны, чтобы продемонстрировать, что их изучение без использования информации об ориентирах приводит к низкой производительности. Эти эпизоды имеют среднее геодезическое расстояние до ближайшей цели не менее 5 метров. Что касается категорий предметов интерьера, то для проводимых экспериментов были выбраны: кровать, шкаф, стул, комод, прилавок, подушка, раковина, диван, стол и унитаз. Все эксперименты были проведены трижды, и дисперсия результатов составила не более 0,05.

Для сравнения метода HLPO с существующими современными подходами были выбраны пять базовых решений, которые не используют информацию об ориентирах. Однако на этапе обучения у них была поставлена цель изучить семантическую структуру сцен.

- **DDPPO** - Алгоритм DDPPO [15] был обучен со входом в виде датчика глубины + GPS + GT семантическим сенсором с вознаграждением, пропорциональным геодезическому расстоянию до целевого объекта, ближайшего к старту агента. За основу нейронных сетей были взяты ResNet50 и LSTM.
- **SemExp** - SemExp [106] представляет собой алгоритм SOTA с модульной структурой, который включает в себя как RL, так и планирование, который показал наилучшие результаты на Habitat Challenge 2020. Для проводимого эксперимента были использованы обученные авторами веса, но из-за различия в классах целевых объектов, агенту подавался на вход только целевой класс, а остальные устанавливались равными нулю.
- **Планирование** — комбинация модулей планирования, которые на каждом этапе обновляют карту препятствий и исследуют ее. Когда семантический модуль видит целевой объект, он отображается на карте, и агент переходит к нему посредством планирования.
- **Auxiliary RL** — общая обучаемая стратегия [18] с дополнительными обучающими задачами и вознаграждением за исследование. Именно сквозной алгоритм показал наилучшую производительность на Habitat Challenge 2021.
- **ExploreTillSeen** - Комбинация двух навыков на основе RL: один исследует местность (навык Explore) до тех пор, пока семантический сенсор не увидит цель, а другой (навык GoalReacher) следует за увиденным целевым объектом.

В результате применения подхода, основанного только на навыках Explore и GoalReacher, были получены результаты, сопоставимые с SOTA методами. Использование полного метода HLPO почти удвоило количество успешных эпизодов при валидации, что указывает на правильность предлагаемого выбора информации об ориентирах в качестве дополнительной необходимой информа-

Method	**GT semantic**		**Learned semantic**	
	Success	SoftSPL	Success	SoftSPL
E2E RL	0.18	0.35	0.11	0.24
SemExp	0.24	0.26	0.11	0.17
Planning	0.31	0.26	0.15	0.18
Auxiliary RL	0.51	0.34	0.19	0.19
ExploreTillSeen	0.46	0.33	0.20	0.21
HLPO	**0.86**	**0.52**	**0.46**	**0.30**
HLPO (Noise)	0.85	0.46	0.45	0.28
HLPO (Map)	0.90	0.54	0.51	0.42

Таблица 6 — Сравнение различных агентов на тренировочных эпизодах.

ции о сцене, и на то, что end-to-end подходы не могут самостоятельно изучить ее.

Для доказательства того, что шумы датчиков и действий не существенно ухудшают результаты навигации, шумы были добавлены и на этапе обучения стратегий, благодаря чему агент научился не обращать на них большого внимания. Проведенные эксперименты (строка HLPO (Noise) в таблице 6) подтвердили это. Эксперимент с добавлением полной карты (строка HLPO (Map) в таблице 6) показал, что общая эффективность метода может быть повышена, если у агента есть доступ к полной карте препятствий. Однако выигрыш не так заметен по сравнению со сложностью сбора полной карты для каждой сцены.

Для проверки необходимости использования ориентиров приведен пример траекторий методов ExploreTillSeen и HLPO (Рис. 3.10). Пример ExploreTillSeen показывает, что его возможности исследования сцены находятся на высоком уровне, но его траектории далеки от оптимальных. Если бы реальный робот каждый раз перемещался по каждой комнате квартиры в поисках раковины, такое поведение было бы нежелательным.

3.4 Выводы

В данной главе был предложен новый подход к навигационной задаче ObjectGoal. Стандартная постановка задачи ограничивает существующие методы, поскольку исследование больших сцен без информации о сцене занимает неоправданно много времени. Для решения этой проблемы было предложено использовать ориентиры в виде списка координат комнат и их типа.

С обновленной формулировкой задачи был разработан новый метод с иерархической стратегией, который использует навыки, комбинируемые и повторно используемые в различных навигационных задачах без изменений. Показатель успеха для предлагаемого метода увеличился в два раза: с 20% для современного метода до 46% с изученной семантикой и с 51% до 86% для истинной семантики из симулятора. Для выполнения задачи ObjectGoal агент был обучен трем навыкам: PointNav, Exploration и GoalReacher.

Чтобы обеспечить возможность и предсказуемость процесса переноса в реальность, был описан процесс реконструкции сцены в симуляторе с достаточным фотореалистичным качеством реконструкции с использованием профессионального сканера Leica RTC360.

Глава 4. Интеграция обучаемых и необучаемых навыков в задаче визуальной навигации

В данной главе для решения задачи навигации к объектам (ObjectNav) представлена гибридная схема SkillFusion, включающая модули, основанные на обучаемых и необучаемых методах, а также метод переключения между ними.

Классические подходы к решению задач навигации обычно включают в себя модульную систему, состоящую из модулей восприятия, картирования, локализации, планирования и управления движением. Эффективность этих подходов зависит от множества факторов, таких как качество данных с датчиков, сложность окружающей среды и количество приложенных инженерных усилий. Данный подход позволяет решать задачи навигации как в симуляции, так и на различных робототехнических платформах [112; 113].

В последние годы стала широко распространена ортогональная методика, основанная на сквозном (end-to-end) обучении с подкреплением. Современные обучаемые методы способны не только свободно перемещаться по сложным сценам без явной реконструкции карты [11; 15; 16], но также обладают способностями к манипуляции [114], долгосрочному планированию [115], построению моделей мира [19] и общению с людьми на естественном языке [3]. Успех этих методов в основном обусловлен большим объемом данных, используемых для обучения. В контексте робототехнических приложений для генерации данных обычно используются различные симуляторы. Это приводит к снижению производительности в реальных условиях, которые могут отличаться от тех, на которых обучалась система. Другой недостаток таких методов заключается в том, что обучаемые методы обычно полагаются только на неявное представление окружающей среды. На практике эти представления не очень подходят для задач с долгим горизонтом. В отличие от этого, классические, необучаемые методы используют явное представление окружающей среды через одновременную локализацию и картографирование (SLAM), и эти представления позволяют агенту проводить эффективное планирование на долгий срок в сложных сценах.

В данной главе предлагается метод, который преодолевает различия между классическими и обучаемыми подходами, предлагая гибридную архитектуру

управления агентом, называемую SkillFusion. Данная концепция объединения обучаемых и необучаемых методов не является новой сама по себе. Однако, в отличие от других методов слияния, которые опираются на модульные архитектуры [13], предлагаемый подход учитывает специфические преимущества обоих парадигм и динамически выбирает, какой навигационный навык выполнять, основываясь на оценке его полезности. Также, все навыки агента были реализованы с использованием как классических, так и обучаемых методов, что позволяет агенту адаптивно выбирать наиболее подходящую парадигму на лету.

В результате, данный подход включает в себя следующие аспекты:

– Задача ObjectNav была поделена на два отдельных навыка агнта. Исследование среды, где агент должен искать целевой объект в заданной области, и GoalReacher, где агент должен навигироваться к обнаруженному целевому объекту и остановиться рядом с ним. Затем была построена модульная архитектура, которая включает в себя как классические, так и обучаемые реализации каждого навыка. Во время выполнения эпизодов, агент управляет всеми навыками в зависимости от их внутренней оценки вознаграждения и внешних условий.
– Чтобы увеличить устойчивость обучаемой стратегии к внешним условиям, был использован предварительно обученный кодировщик визуальных наблюдений. Также были выбраны задачи навигации, которые требуют разного поведения от агента, но имеют одинаковый вход. Эти задачи были обучены с использованием одной нейронной сети, используя архитектуру раннего слияния [64], что позволило использовать одну нейронную сеть с общими рекуррентными слоями для всех задач и отдельные головы для каждой из задачи.
– Чтобы решить проблему шумов при семантической сегментации изображений с камеры (ложное обнаружение целевых объектов, выбросы и т.д.), были включены ряд техник как для необучаемых, так и для обучаемых модулей. Для необучаемых модулей была использована эрозия и затухание карты. Для обучаемых модулей было введено специальное действие “не уверен”, которое предотвращает направление обучаемой стратегии к ложно обнаруженному объекту, переключаясь на навык исследования.

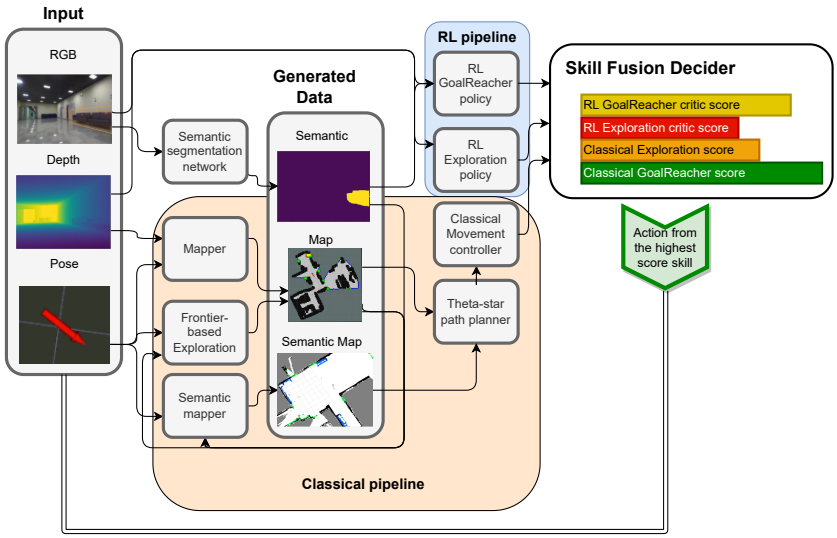

Рисунок 4.1 — Архитектура предлагаемого алгоритма для навигации SkillFusion, использованная на реальном роботе. Схема состоит из классической и обучаемой части, и каждая из которых имеет навыки исследования и достижения цели. Чтобы в каждый момент выбрать подходящий навык, был реализован механизм слияния навыков, который выбирает действие из доступных навыков с наибольшим значением полезности в данный момент.

4.1 Навигация с помощью классических навыков планирования агента

Реализация классических навыков Exploration и GoalReacher состоит из следующих частей: постановка промежуточной цели, локализация, картографирование, планирование пути и управление движением.

Модуль постановки промежуточной цели. Модуль постановки промежуточной цели принимает текущую оценку положения робота и карту, построенную с помощью SLAM, и выбирает цель, куда должен направиться робот. Если целевой объект отображен на карте SLAM, то выбирается ближайшая к нему точка.

В качестве модуля постановки промежуточной цели, была использована реализация подхода к исследованию среды на основе границ, которая была предложена в работе [75]. Алгоритм ищет границы между свободным и неизвестным пространством на 2D карте, построенной с помощью SLAM. Для поиска этих границ используется поиск в ширину (BFS).

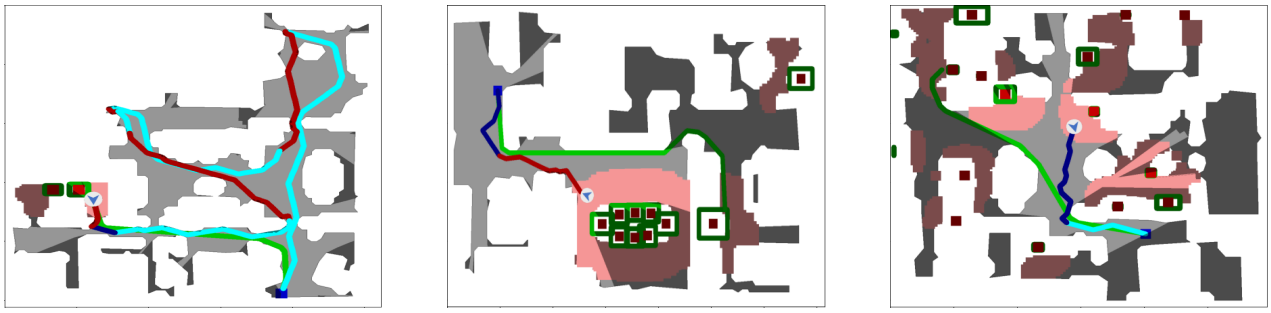

Рисунок 4.2 — Примеры работы SkillFusion в симуляторе. Красная линия - траектория, полученная классическим методом, светло-синяя линия - обучаемым навыком Exploration, а темно-синяя - обучаемым навыком GoalReacher.

Всем границам присваиваются функции стоимости, а центроид границы с наименьшей функцией стоимости отмечается как цель для робота. Функция стоимости границы учитывает расстояние между агентом и центроидом границы, размер границы и угол поворота агента перед тем, как он начнет движение к границе. Формально это записывается через следующее выражение.

Пусть $q \in \mathbb{R}^2$ - это вектор ориентации робота, а $\pi = (p_0, p_1, \ldots, p_k)$ это путь от робота к центроиду i-го фронта размера n_i. В пути π, p_0 это позиция робота, а p_k - это центроид $i - th$ фронта. Функция стоимости при этом определяется следующим образом:

$$cost_i = \alpha \sum_{j=0}^{k} ||p_{j+1} - p_j||_2 - \beta n_i + \gamma |\angle(q, p_1 - p_0)| \tag{4.1}$$

где α - это коэффициент для длины пути, β - это коэффициент для размера фронта, а γ - это коэффициент для угла поворота между ориентацией робота и направлением к фронту.

Модуль локализации. В симуляции используются данные о точном положении робота, поэтому внешний модуль локализации не требуется. На реальном роботе, для максимальной точности локализации и избежания накопления ошибок используется алгоритм лидарной одометрии LOL-odom [116] с дополнительной коррекцией положения через предварительно построенную 3D карту (эта карта используется только для коррекции положения).

Модуль картирования. Модуль картирования строит карту занятости сетки для навигации на этапе исследования сцены и семантическую карту для навигации к целевому объекту. В симуляции, где доступны данные о точном

положении, для построения карты занятости используется обратную проекцию карты глубины. Для построения семантической карты, используется обратная проекция предсказанной семантической маски. Все объекты высотой менее 0,2 м отображаются как ячейки пола, а все объекты выше 0,2 м отмечаются как препятствия. На реальном роботе используется алгоритм SLAM RTAB-MAP [72] с лидаром в качестве источника данных. Семантическая информация добавляется на карту RTAB-MAP из семантических масок, предсказанных с бортовой RGBD-камеры.

Как в симуляции, так и на реальном роботе, карта строится с разрешением 5 см. Чтобы закрыть маленькие “дыры” на карте и уменьшить количество избыточных фронтов, возникающих из-за пробелов на карте, размер ячейки увеличивается в два раза, используя метод максимального объединения (max pooling) с порядком значений ячеек “unknown < free < obstacle“: карта разбивается на квадраты 2x2, и каждый квадрат формирует одну ячейку на увеличенной карте. Эта ячейка помечается как препятствие, если квадрат содержит ячейку препятствия. Если квадрат 2x2 содержит ячейку свободного пространства и не содержит ячейку препятствия, то результирующая ячейка на увеличенной карте помечается как ячейка свободного пространства. И если квадрат 2x2 содержит только неизвестные ячейки, то результирующая ячейка увеличенной карты помечается как неизвестная. Пример такого максимального объединения показан на рис. 4.3.

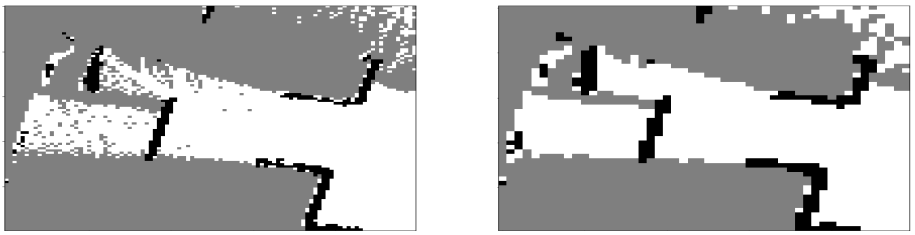

Рисунок 4.3 — Пример объединения построенной карты. Серый цвет обозначает неизвестную область, черный - ячейки препятствий, а белый - ячейки свободного пространства

Для предотвращения выбросов при картировании в предсказанных семантических масках, была реализована эрозия семантической карты и накопление семантической информации. Кроме того, используются непрерывные значения ячеек семантической карты вместо бинарных. Если семантический объект проецируется на ячейку карты, то значение в этой ячейке увеличивается на 1, в

противном случае значение умножается на коэффициент $\alpha < 1$. Ячейка помечается как ячейка целевого объекта, если ее значение превышает порог T. В проводимых экспериментах, были использованы значения $\alpha = 0.9$ и $T = 2$.

Модуль планирования. Для планирования пути был использован алгоритм Theta* [12]. Это алгоритм поиска пути на графе, который поддерживает перемещение агента под любым углом. По умолчанию Theta* планирует путь для агента нулевого размера, поэтому он был модифицирован для учета размера робота. В предлагаемой модификации, робот моделируется как диск радиуса r. На практике размер диска устанавливается больше, чем фактический размер робота, чтобы был запас по безопасности. Если путь не найден, то безопасный радиус уменьшается до $0.8r$ и повторяется планирование пути. Для проводимых экспериментов, $r = 0.15$ для симуляции и $r = 0.6$ для робота.

Модуль контроллера. В симуляторе выбранное действие выполнялось без дополнительных обработок в дискретном виде, а на реальном роботе была использована реализация метода следования траектории [1], которая получает путь и положение робота, и выдает команды скорости для робота. Это следование траектории похоже на подход к управлению в симуляции, но оно адаптировано для непрерывного движения и имеет некоторые эвристики для компенсации ошибок одометрии.

Для навыка GoalReacher были использованы те же модули локализации, планирования, картирования и контроллера, которые описаны выше. Для достижения нанесенной на карту цели был использован подход на основе границ, но вместо границ между свободным и неисследованным пространством карты используются границы целевых объектов.

4.2 Навигация с помощью обучаемых навыков агента

Так же, как и классические, обучаемые навыки были разделены на две части: стратегия исследования среды (Exploration) и стратегия достижения цели (GoalReacher). Для обучения этих навыков была использована децентрализованная распределенная оптимизация ближайшей стратегии (DD-PPO), так как

[1] https://github.com/andrey1908/strl_robotics/tree/main/control

она показывает одни из лучших результатов в аналогичных задачах визуальной навигации [15]. Архитектура для обучения предложенных обучаемых навыков показана на рис. 4.4. Ключевым фактором для этого результата было огромное количество шагов обучения. Это требует быстродействующего симулятора, который также должен быть фотореалистичным, для возможности переноса полученной стратегии в реальный мир. В этой связи в данной работе также был использован самый быстрый из доступных фотореалистичных симуляторов, BPS, [22] с самым большим набором данных из 1 000 сцен, HM3D [45]. Для обучения обучаемого навыка Exploration использовалась обучающая часть набора данных HM3D (800 сцен) и 145 сцен для обучаемого навыка GoalReacher, так как в HM3D есть только это количество сцен с доступной семантикой.

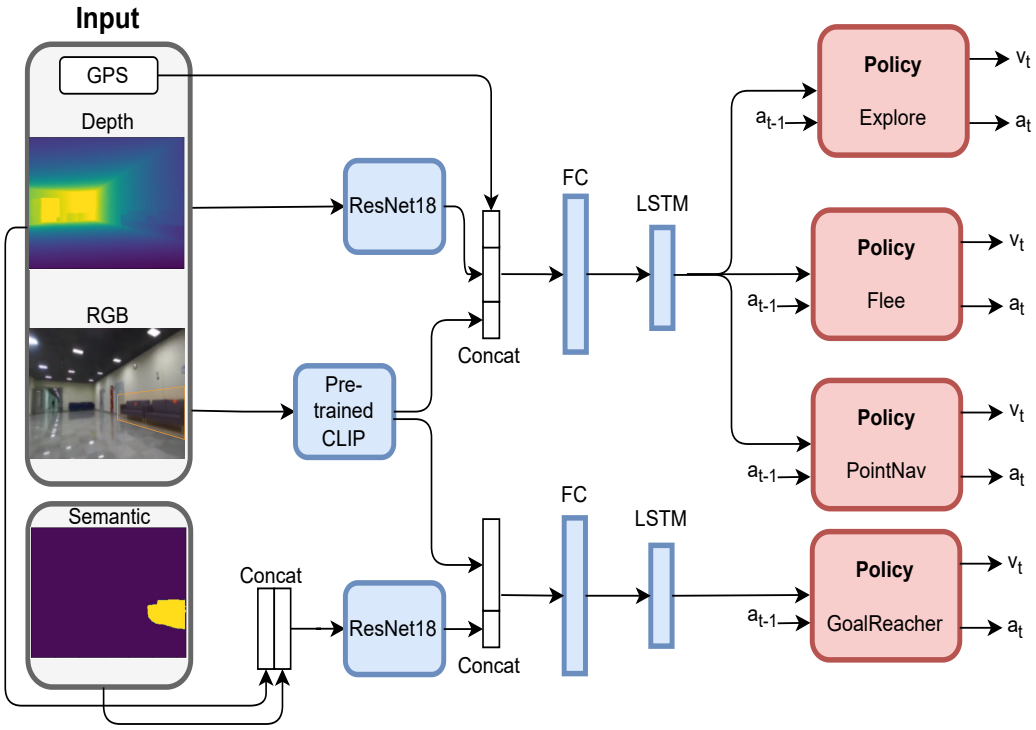

Рисунок 4.4 — Предложенная архитектура нейронной сети метода SkillFusion для синтеза задач навигации.

Навык Exploration. Исследование сцены является сложной задачей, которая требует от стратегии агента точности, способности запоминать прошлый опыт и планировать будущее. В реальных и фотореалистичных средах, эта задача усложняется значительным разнообразием размеров, планировки, обстановки, цвета и других параметров внутренних сред. Чтобы обобщить эти особенности, агенту необходим мощный кодировщик, чтобы извлечь всю необходимую информацию из шумного RGB-датчика. Решение этой проблемы состояло

из двух частей: получение визуально устойчивого кодировщика изображений и разработка динамически устойчивого RNN-кодировщика.

В качестве кодировщика изображений была использована предварительно обученная модель CLIP [67] и заморозили ее во время фазы обучения. Этот подход использовался в предыдущих работах [68] [69], и проведенные в данной работе эксперименты также демонстрируют способность модели навигации исключительно на основе RGB-датчика с замороженным кодировщиком CLIP.

Чтобы решить проблему динамически устойчивого RNN-кодировщика, все навигационные навыки были объедены и обучены с помощью одной нейронной сети (Рис. 4.4) с общими визуальными кодировщиками и слоями RNN и несколькими головами [64]. Навык Exploration должен быть осведомлен о пространственной структуре помещения, чтобы избежать исследования уже исследованных областей, в то время как навык Flee фокусируется на измерениях расстояний и ориентаций, навык PointNav фокусируется на поиске кратчайшего возможного пути к точке. Функция вознаграждения каждой задачи мотивирует эти поведения.

- Вознаграждение за навык Exploration устанавливается как $+1$, когда агент посещает ранее неисследованную ячейку размером $1m^2$, в противном случае 0.
- Вознаграждение за навык Flee пропорционально расстоянию от начальной точки.
- Вознаграждение за навык PointNav пропорционально сокращенному расстоянию до цели, плюс вознаграждение, если агент выполняет действие остановки в пределах 1 метра от целевой точки.

Навык GoalReacher. Важным аспектом навыка GoalReacher является способность различать целевые и нецелевые объекты. Чтобы передавать модели информацию об объекте, используется бинарная маска сегментации объекта, а не его идентификатор. Этот подход позволил обучать модель семантической сегментации независимо от стратегии агента. Недостатком этого является то, что нужно обучать стратегию с истинным датчиком семантики в симуляторе, чтобы иметь возможность давать правильные вознаграждения.

Во время проверочных эпизодов модель семантической сегментации может производить шум и ложноположительные объекты, которые появляются с некоторой периодичностью. Навык GoalReacher обучается с большим положи-

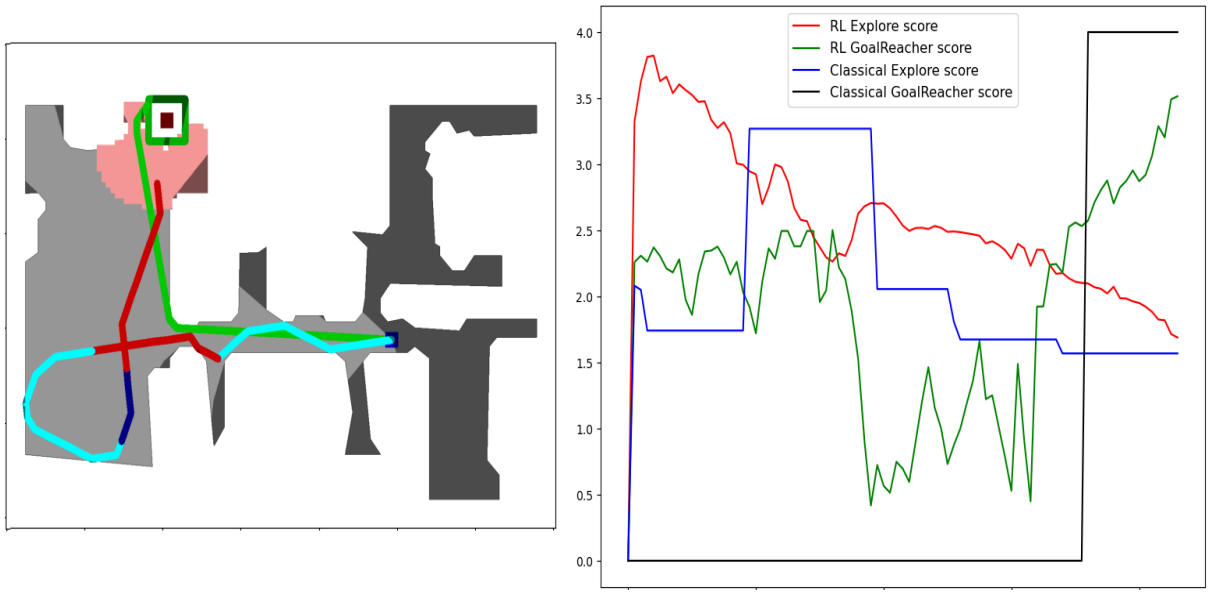

Рисунок 4.5 — Пример управления навыками во время эпизода. Синий цвет траектории обозначает выполнение обучаемого навыка Exploration, темно-синий обозначает обучаемый навык GoalReacher, а красный цвет обозначает классические навыки.

тельным вознаграждением за успешное завершение эпизода и малым отрицательным вознаграждением за каждый выполненный шаг. В результате шумные объекты могут заставить агента завершить эпизод в месте, где появляется шумный объект, в надежде, что это настоящий, что приведет к большому отрицательному вознаграждению, если это не так. Или же агент может попытаться найти другой более далекий объект и накопить большую сумму малых отрицательных вознаграждений, выполнив слишком много действий.

Чтобы позволить агенту различать шумные и реальные объекты и избежать попадания в неблагоприятные ситуации принятия решений, предлагается использовать новое действие “not sure". Когда агент не уверен в достоверности маски сегментации, он может выполнить это действие, чтобы вернуться к навыку Exploration, а не преследовать воспринимаемый шумный объект.

Во время фазы обучения навыка GoalReacher, эпизоды с ложной семантической сегментацией объекта выбираются с вероятностью 0,2, остальные эпизоды содержат истинную сегментацию. В шумных эпизодах агент может получить небольшое отрицательное вознаграждение, если он выбирает действие “not sure когда завершает эпизод, и большое отрицательное вознаграждение, если он выполняет действие “stop"в любом другом месте. Цель агента - быстро распознать, является ли семантическая сегментация ложной, и завершить эпи-

зод с действием “not sure или преследовать цель, убедиться в ее достоверности и выполнить действие “stop"рядом с ее положением.

4.3 Объединение классических и обучаемых подходов для задач навигации.

Чтобы объединить преимущества классических и обучаемых подходов, предлагается метод SkillFusion, так как, будучи отдельными, эти подходы имеют свои ограничения.

Классические методы исследования на основе границ на 2D карте надежны и устойчивы к изменениям сцены и камеры; они требуют только точных источников одометрии и карты. Однако, обучаемые подходы могут быть более эффективными при решении конкретных сложных задач, так как они могут быть обучены напрямую решать эти задачи, например, используя методы обучения с подкреплением. В задаче навигации к целевым объектам, подходы на основе RL показывают большую эффективность в сценах, где цель можно найти за короткое время, но на больших расстояниях рекуррентные нейронные сети начинают забывать информацию с первых шагов, в следствии чего агент начинает посещать уже исследованные области. Классические методы, с другой стороны, хранят всю информацию в виде 2D карты и могут исследовать новые места без повторений в течение всего времени работы. Кроме того, подходы обучения с подкреплением хороши в поиске цели и прицеливании на нее, но им трудно остановиться вблизи объекта точно в заданном радиусе.

Чтобы эффективно использовать преимущества каждого подхода, подход SkillFusion должен знать о тех преимуществах, которые каждый навык может предоставить в любой момент времени. В стратегии агента уже есть функции стоимости для каждого из навыков: функция критика для обучаемых навыков и функция стоимости границ и пути для классических навыков (Рис. 4.5). Данные функции нормируются до одного порядка на основе собранного опыта, и на каждом шаге агент выполняет навык с максимальным значением функции полезности. Так как целью задачи ObjectNav является выполнение действия остановки, когда агент находится рядом с целью, классическому навыку

GoalReacher присваивается высокое постоянное значение, когда цель появляется на карте после всех фильтров, так как этот навык наиболее полезен для выполнения задачи, чем любые исследования сцены.

В результате, агент начинает исследование сцены с помощью обучаемых навыков для эффективного покрытия больших областей вокруг стартовой позиции. Спустя некоторое время после старта, обучаемые навыки забывают информацию в своих рекуррентных слоях и начинают исследовать уже исследованные области. Когда это происходит, оценка обучаемого навыка Exploration становится меньше оценки полезности классического навыка Exploration, и агент переключается на классический навык. В этот момент классический навык Exploration на основе текущей карты направляет агента к границе с наивысшей стоимостью, и после этого оценка обучаемого навыка Exploration становится больше классической, и стратегия возвращается к обучаемому навыку (Рис. 4.2 слева). Следует отметить, что рекуррентные слои обучаемых навыков обновляются на каждом шаге, даже когда агент действует на основе классических навыков.

Если целевой объект становится видимым для семантической сегментации, оценка обучаемого навыка GoalReacher резко повышается и берет на себя контроль агентом, пока целевой объект не будет нанесен на семантическую карту. При выполнении обучаемого навыка GoalReacher, агент имеет два варианта: завершить эпизод самостоятельно, выполнив действие “стоп"(Рис. 4.2 справа), или вернуть контроль навыкам Exploration, выполнив действие “не уверен если он решит, что целевой объект является шумом. Когда целевой объект уже находится на карте сегментации, оценка классического навыка GoalReacher устанавливается на постоянно высокое значение, агент переключается на этот навык и завершает эпизод по достижению цели (Рис. 4.2).

4.4 Эксперименты

4.4.1 Эксперименты в симуляторе

Табл. 7 показывает результаты предлагаемого метода по сравнению с текущими передовыми решениями. В качестве проверочных эпизодов были выбраны 20 сцен из части Val набора данных HM3D [45], которые агент никогда не видел во время своей фазы обучения. Результирующее число эпизодов составило 120 с равным распределением категорий целей (стул, кровать, растение, туалет, монитор, диван) со средним минимальным расстоянием до ближайшего объекта 8 м. В качестве базового уровня для сравнения были выбраны три метода. DDPPO [15] - это сквозной (end-to-end) метод RL, который был предоставлен авторами симулятора Habitat. SemExp [13] - это модульный подход, который включает в себя слияние обучаемого глобального планирования и классического локального планирования. Auxiliary RL [16] - это подход RL, который был дополнен вспомогательными функциями потерь. Для всех данных методов были выбраны лучшие доступные реализации с открытым исходным кодом и весами, но чтобы сделать эти методы применимыми к набору данных HM3D и используемому типу целевых объектов, во всех подходах использовался модуль семантической сегментации SegFormer.

Таблица 7 — Сравнение предлагаемого метода SkillFusion с другими алгоритмами на наборе данных HM3D.

Method	**Success**	**SPL**	**SoftSPL**
DDPPO [15]	0.18	0.10	0.35
SemExp [13]	0.24	0.14	0.26
Auxiliary RL [16]	0.51	0.29	0.34
SkillFusion	**0.64**	**0.36**	**0.38**

У обучаемых стратегий кодировщик RGB-кодировщик составляет значительную часть параметров нейронной сети, что замедляет фазу обучения. Кроме того, количество сцен в наборе данных ограничено, но от агента требуется, чтобы он мог навигировать за их пределами. Поэтому, обобщающая способ-

ность RGB-кодировщика в новых сценах является сложной проблемой. Решением этой проблемы может быть использование уже предварительно обученного RGB-кодировщика, который будет заморожен во время фазы обучения. В качестве такого кодировщика была выбрана модель CLIP [67]. Чтобы продемонстрировать, что сокращение обучаемых параметров все равно позволяет агенту выполнять задачи навигации, производительность задачи GoalReacher была сравнена с замороженным CLIP по сравнению с обучаемой моделью ResNet 4.6. Как показал данный эксперимент, производительность агента даже улучшилась. Для анализа влияния датчиков глубины на производительность и демонстрации того, что нейронная сеть опирается на встраивания CLIP не только для распознавания объектов, но и для решения задач навигации, навык GoalReacher был обучен исключительно на входных данных RGB. Предлагаемая архитектура с этим ограничением все равно показала высокий результат.

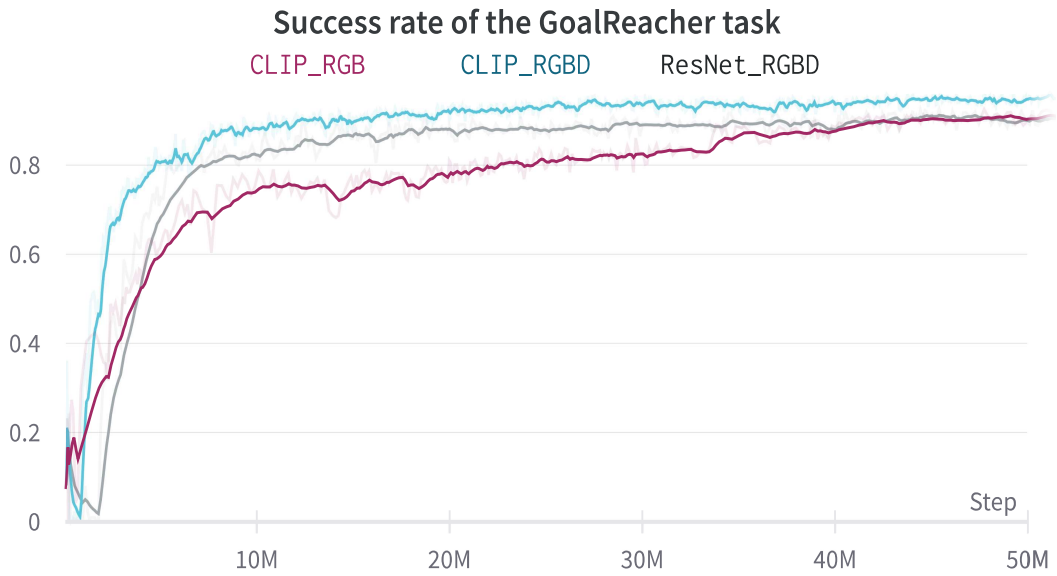

Рисунок 4.6 — Сравнение обучения навыка GoalReacher с использованием необучаемого кодировщика изображений CLIP и обучаемого кодировщика ResNet.

Чтобы продемонстрировать необходимость слияния классических и обучаемых подходов, были протестированы различные комбинации классических и обучаемых навыков. Результаты сравнения показывают, что только обучаемый подход лучше, чем только классический подход по метрике SPL в симуляторе, но имеет сопоставимую вероятность успеха (см. Таблицу 8). Это происходит потому, что классические методы полагаются только на карту, и являются “слепыми"за пределами ограничений диапазона глубины и тратят много времени

на исследование углов или тупиков, в то время как обучаемые методы могут видеть их, извлекая информацию из RGB камеры. Сопоставимая вероятность успеха обусловлена тем, что для классических методов проще решить, когда агенту нужно завершить эпизод, так как они могут рассчитать расстояние до цели с помощью метрической карты.

Таблица 8 — Абляция различных доступных наборов навыков агента во время выполнения эпизода.

Skill		**Metrics**		
Explore	GoalReacher	Success	SPL	SoftSPL
Classical	Classical	0.410	0.182	0.263
RL	RL	0.403	0.224	0.321
RL+Classical	Classical	0.511	0.299	0.309
RL+Classical	RL+Classical	0.547	0.316	0.365

Как показано в таблице 9, производительность модели SkillFusion чувствительна к качеству семантической сегментации. С истинным значением (GT) семантики результаты почти вдвое превосходят результаты SegFormer [6], но с реализацией фильтрации семантической карты для классических методов и добавлением действия “не уверен"в обучаемые методы, этот разрыв был по большей части компенсирован.

Таблица 9 — Абляции модуля семантической сегментации.

Method	**Success**	**SPL**	**SoftSPL**
SkillFusion (with no semantic filtering)	0.324	0.207	0.327
SkillFusion (with semantic filtering)	**0.547**	**0.316**	**0.365**
SkillFusion (ground truth semantic)	0.647	0.363	0.384

4.4.2 Эксперименты на роботе

Кроме экспериментов с симуляцией, был проведен ряд тестов на роботе Clearpath Husky (Рис. 4.7). От робота требовалось найти различные объекты (например, стул или диван) в неизвестной агенту среде (здание университета).

Рисунок 4.7 — Робот-платформа на базе шасси Clearpath Husky с камерой ZED (слева). Он был использован для оценки результатов на реальной сцене (справа).

Он выполнял задачи без какой-либо дополнительной настройки нейронных сетей.

Чтобы обнаружить сильные и слабые стороны классических и обучаемых методов в случае реального робота, они были сначала протестированы отдельно, а затем были проведены сравнения с методом SkillFusion. Все тестируемые методы были проверены на семи сценах с тремя разными целевыми объектами: синим стулом, красным стулом и синим диваном. Синий стул был расположен в углу коридора. В тесте с диваном два больших синих дивана были расположены в разных углах коридора. Красный стул был расположен за узким проходом в коридоре, и роботу пришлось исследовать весь коридор, прежде чем войти в этот узкий проход, чтобы достичь цели. Для синего стула и диванов алгоритмы были протестированы с трех разных начальных мест, в то время как для красного стула было использовано только одно начальное место.

Траектории для всех методов показаны на рис. 4.8, а значения метрик показаны в таблице 10. Были измерены пять метрик: коэффициент успешности, SPL, SoftSPL, длина пройденного роботом пути и время в пути. С только RL подходом робот дважды не смог достичь целевого объекта. В тесте с синим стулом робот достиг дивана вместо стула. Это произошло из-за ошибок семантического сегментатора и подхода к достижению цели без фильтрации семантических данных. В тесте с красным стулом робот проигнорировал узкий проход со стулом, потому что RL был обучен с функцией вознаграждения, про-

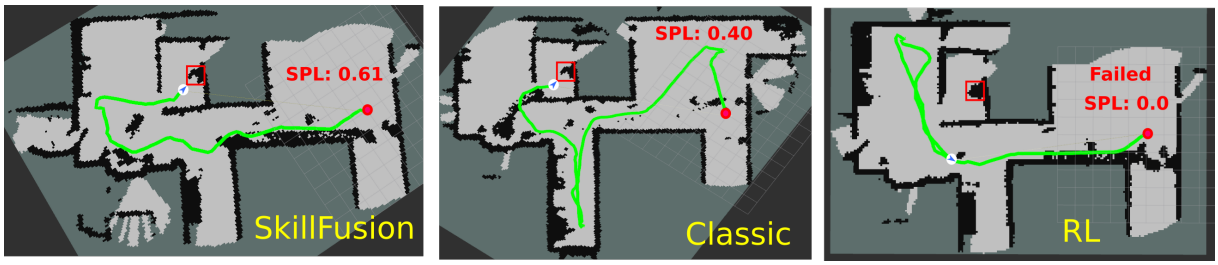

Рисунок 4.8 — Траектории робота при использовании разных методов: SkillFusion (слева), классический метод (посередине) и обучаемый метод (справа). Красный прямоугольник обозначает целевой объект, красный круг обозначает начальную точку, а белый круг с синей стрелкой обозначает точку остановки робота и направление.

порциональной исследованной области. В результате функция вознаграждения за вход в узкий проход была слишком низкой.

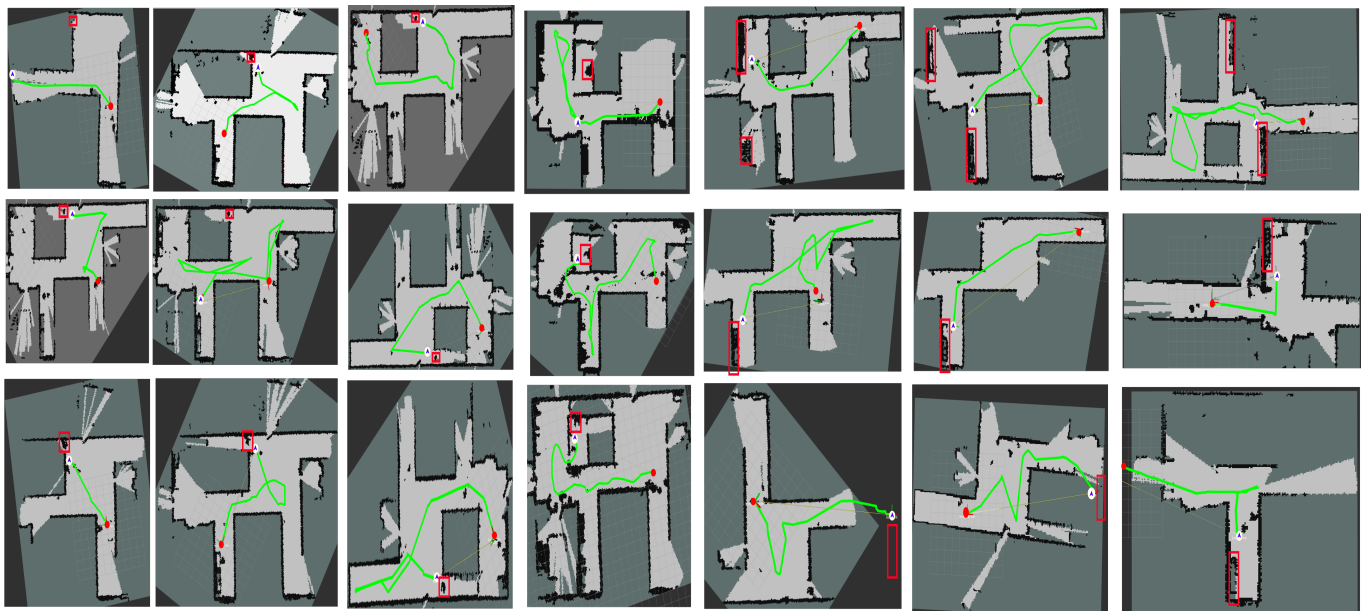

Рисунок 4.9 — Траектории робота: RL (верхний ряд), классическая схема (средний ряд) и SkillFusion (нижний ряд). Красный прямоугольник обозначает целевой объект, красный круг обозначает начальную точку, а белый круг с синей стрелкой обозначает точку завершения робота и направление.

В одном из трех тестов робот не смог найти синий стул, используя только классические навыки (рис. 4.9). Это было связано с ограниченным диапазоном 5 м для построения карты и шумами семантического сегментатора, которые привели к игнорированию целевого стула. В результате робот исследовал весь коридор, не отметив стул как цель.

Используя метод SkillFusion, робот успешно справился со всеми тестами, достигнув среднего значения SPL и SoftSPL равного 0,65. Средняя пройденная дистанция была сокращена до 23 м, по сравнению с 27 м при классическом подходе и 31 м при RL-подходе. Это демонстрирует успешную интеграцию сильных сторон обоих подходов - классического и основанного на обучении - реальному роботу.

Таблица 10 — Результаты отдельных тестов классических и обучаемых подходов на реальном роботе в сравнении с методом SkillFusion.

Method	Scene	Success	SPL	SoftSPL	Path Length, m	Time, s
RL	Chair 1	0.67	0.53	0.42	22	170
	Chair 2	0	0	0.42	36	300
	Sofa	1	0.39	0.39	38	190
	Average	0.71	0.39	0.41	31	200
Classic	Chair 1	0.67	0.35	0.41	28	180
	Chair 2	1	0.40	0.40	33	220
	Sofa	1	0.75	0.75	24	110
	Average	0.86	0.53	0.56	27	170
SkillFusion	Chair 1	1	0.61	0.61	25	150
	Chair 2	1	0.61	0.61	22	140
	Sofa	1	0.77	0.77	22	110
	Average	**1**	**0.65**	**0.65**	**23**	**130**

4.5 Выводы

В данной главе была рассмотрена сложная проблема визуальной навигации. Обычно она ставит перед воплощенным агентом (виртуальным или реальным роботом) задачу достичь объекта, принадлежащего к определенному классу (например, стол, стул и т.д.) в неизвестной среде, которую агент может локально наблюдать с помощью своего визуального датчика. В ходе данной главы были изучены как необучаемые (классические), так и обучаемые методы, необходимые для эффективного решения этой проблемы. Данная глава ясно показывает, что классические и обучаемые подходы к навигации роботов не противоречат, а дополняют друг друга. Для этого они были реализованы и в

симуляторе и в реальных условиях на высоком уровне и были определены их сильные и слабые стороны.

Для повышения метрики SPL и доли успешности траекторий классического подхода был настроен и улучшен метод исследования сцены на основе границ. Для применения классических навыков в реальных условиях, где недоступны датчики глубины и GPS, были использованы модули локализации и картирования с использованием лидара, что позволило достичь схожих метрик качества. Для повышения производительности обучаемых навыков была введена схема обучения со слиянием нескольких навыков и новое действие “не уверен”, которое не только увеличивает метрики, но и делает модель более устойчивой к внешним условиям. Эти улучшения также сделали обучаемые навыки переносимыми в реальный мир без каких-либо адаптации.

Каждая часть классических и обучаемых алгоритмов была реализована в качестве навыков агента и был предложен модуль принятия решений о выборе навыка, который на основе внутренней модели вознаграждения каждого навыка переключает агента между классическими и обучаемыми навыками в течение эпизода. Данный предложенный механизм слияния подходов основывается на фундаментальных различиях между классическими и обучаемыми методами. В проведенных экспериментах, обучаемые методы на основе обучения с подкреплением хорошо справляются с быстрым исследованием больших областей и способны извлекать больше информации из датчиков для более интеллектуального исследования, но они не так точны, и их способности к запоминанию значительно уступают классическим методам.

Заключение

Основные результаты работы заключаются в следующем.

– Для задачи навигации до точки был предложен иерархический метод с автоматическим выбором подцелей и интеграцией алгоритмов картирования и семантической сегментации. Работоспособность данного метода проверена как в более простой 2D постановке, так и в фотореалистичном симуляторе в условиях шума данных в датчиках и актуаторах. Проведено множество численных экспериментов, показывающих производительность данного метода в сравнении с методами без выделения иерархий по скорости обучения.
– Был предложен новый подход к навигационной задаче поиска целевых объектов. Как показали проведенные численные эксперименты, при стандартной постановке задачи, существующие методы ограничены тем, что в семантически сложных сценах, исследование без априорной информации о сцене занимает в два раза больше времени по сравнению с использованием полной карты местности. Однако построение точной карты местности для каждой сцены с помощью визуальной информации на мобильном роботе не представляется возможным. Для решения этой проблемы были предложены ориентиры в виде списка координат комнат и их типа. С обновленной формулировкой задачи была создана новая иерархическая архитектура, в которой используются навыки, которые можно комбинировать и повторно использовать в различных навигационных задачах без изменений. Был разработан процесс переноса обучаемых навыков на реального робота через реконструкцию сцены в симуляционную среду с фотореалистичным качеством, и дообучением в ней агента перед переносом стратегии на реального робота.
– Классические и обучаемые подходы к навигации роботов были реализованы в виде навыков агента и был предложен модуль выбора навыков, который на основе внутренней модели оценки полезности каждого навыка, делает выбор между ними во время выполнения задачи. Предлагаемый метод SkillFusion опирается на фундаментальные различия между классическими и обучаемыми методами. Проведенные численные экс-

перименты показали прирост доли успешных траекторий на 30% при использовании предлагаемого подхода по сравнению с классическим и обучаемым подходами. Добавление обучаемой стратегии к классической при выполнении навыка исследования сцены показало прирост на 20% за счет избегания застревания агента в узких местах, где агент не может построить точную карту препятствий. При выполнении навыка следования до цели, обучаемая стратегия показала прирост 10% итоговой метрики качества за счет способности дальнего обнаружения объектов, которые не могут быть нанесены на карту с достаточной точностью.

Список сокращений и условных обозначений

RL - Reinforcement Learning, обучение с подкреплением

MDP - Markov Decision Process, марковский процесс принятия решений (МППР)

POMDP - Partially Obervable MDP, частично обозреваемый МППР

Q-learning - Q-обучение, off-policy алгоритм табличного RL

A2C - Advantage Actor-Critic, on-policy алгоритм глубоко RL

PPO - Proximal Policy Optimization, on-policy алгоритм глубокого RL. Улучшенная версия A2C

LSTM - Long Short-Term Memory, долгая краткосрочная память

FC - Fully Connected, полносвязный слой нейронной сети

MLP - Multilayer perceptron, многослойная сеть из полносвязных слоёв

CNN - Convolutional Neural Network, сверточная нейросеть

Словарь терминов

обучение с подкреплением (reinforcement learning) - Область машинного обучения, которая занимается алгоритмами последовательного принятия решений в среде, динамика которой не известна.

доход (return, reward-to-go) - сумма будущих дисконтированных наград начиная с определенного момента времени $G_t = \sum_{T \geq i=t} \gamma^(i-t) r_i$

предобучение (Pretraining) - Предварительное обучение модели, применяемое до обучения модели на целевой задаче.

дообучение (Fine-tuning) - Обучение предобученной модели на новой целевой задаче.

многозадачное обучение (multitask learning) - Обучение модели решать сразу несколько задач.

подход с явной стратегии (on-policy) - Семейство алгоритмов обучения с подкреплением, которым необходимо собирать данные при помощи той же стратегии, которая на них обучается.

подход без явной стратегии (off-policy) - Семейство алгоритмов обучения с подкреплением, которые могут обучать стратегию на данных собранных другой стратегией.

рекуррентные сети (recurrent neural networks, RNN) - Архитектура нейросетевой памяти. На каждом шаге меняет вектор состояния $h_t = RNN(x_t, h_{t-1})$, который выполняет роль хранилища прошлой информации.

обратное распространение ошибки через время (Backpropogation Through Time) - Основной метод обучения рекуррентных сетей. Эквивалентен методу обратного распространения ошибки, с той разницей, что градиенты проводятся между двумя соседними запусками RNN на последовательных временных шагах

Список литературы

1. Habitat: A Platform for Embodied AI Research / Manolis Savva* [и др.] // Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). — 2019.
2. A Generalist Agent / S. Reed [и др.]. — 2022. — DOI: 10.48550/ARXIV.2205.06175. — URL: https://arxiv.org/abs/2205.06175.
3. TEACh: Task-driven Embodied Agents that Chat / A. Padmakumar [и др.]. — 2021. — DOI: 10.48550/ARXIV.2110.00534. — URL: https://arxiv.org/abs/2110.00534.
4. AI2-THOR: An Interactive 3D Environment for Visual AI / E. Kolve [и др.]. — 2019. — arXiv: 1712.05474 [cs.CV].
5. Application of pretrained large language models in embodied artificial intelligence // Doklady Mathematics. т. 106. — Springer. 2022. — S85—S90.
6. SegFormer: Simple and efficient design for semantic segmentation with transformers / E. Xie [и др.] // Advances in Neural Information Processing Systems. — 2021. — т. 34. — с. 12077—12090.
7. BlendMask: Top-down meets bottom-up for instance segmentation / H. Chen [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2020. — с. 8573—8581.
8. SOLOv2: Dynamic, Faster and Stronger / X. Wang [и др.] // arXiv preprint arXiv:2003.10152. — 2020.
9. *Sumikura S.*, *Shibuya M.*, *Sakurada K.* OpenVSLAM: A versatile visual SLAM framework // Proceedings of the 27th ACM International Conference on Multimedia. — 2019. — с. 2292—2295.
10. *Mur-Artal R.*, *Tardos J. D.* ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras // IEEE Transactions on Robotics. — 2017. — окт. — т. 33, № 5. — с. 1255—1262. — DOI: 10.1109/tro.2017.2705103. — URL: http://dx.doi.org/10.1109/TRO.2017.2705103.

11. Using Deep Reinforcement Learning with Automatic Curriculum Learning for Mapless Navigation in Intralogistics / H. Xue [и др.] // Applied Sciences. — 2022. — т. 12, № 6. — DOI: 10.3390/app12063153. — URL: https://www.mdpi.com/2076-3417/12/6/3153.

12. Theta^*: Any-angle path planning on grids / A. Nash [и др.] // AAAI. т. 7. — 2007. — с. 1177—1183.

13. Object Goal Navigation using Goal-Oriented Semantic Exploration / D. S. Chaplot [и др.]. — 2020. — DOI: 10.48550/ARXIV.2007.00643. — URL: https://arxiv.org/abs/2007.00643.

14. Learning to Explore using Active Neural SLAM / D. S. Chaplot [и др.]. — 2020. — DOI: 10.48550/ARXIV.2004.05155. — URL: https://arxiv.org/abs/2004.05155.

15. DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames / E. Wijmans [и др.]. — 2019. — DOI: 10.48550/ARXIV.1911.00357. — URL: https://arxiv.org/abs/1911.00357.

16. Auxiliary tasks and exploration enable objectgoal navigation / J. Ye [и др.] // Proceedings of the IEEE/CVF International Conference on Computer Vision. — 2021. — с. 16117—16126.

17. *Sutton R. S.*, *Barto A. G.* Reinforcement learning: An introduction. — MIT press, 2018.

18. Auxiliary Tasks and Exploration Enable ObjectNav / J. Ye [и др.]. — 2021. — arXiv: 2104.04112 [cs.CV].

19. Mastering Atari with Discrete World Models / D. Hafner [и др.]. — 2020. — DOI: 10.48550/ARXIV.2010.02193. — URL: https://arxiv.org/abs/2010.02193.

20. *Rasmussen D.*, *Voelker A.*, *Eliasmith C.* A neural model of hierarchical reinforcement learning // PloS one. — 2017. — т. 12, № 7. — e0180234.

21. *Hengst B.* Hierarchical approaches // Reinforcement Learning: State-of-the-Art. — 2012. — с. 293—323.

22. Large Batch Simulation for Deep Reinforcement Learning / B. Shacklett [и др.] // International Conference On Learning Representations (ICLR). — 2021.

23. *Aleksey S.*, *Panov A. I.* Hierarchical actor-critic with hindsight for mobile robot with continuous state space // Advances in Neural Computation, Machine Learning, and Cognitive Research III: Selected Papers from the XXI International Conference on Neuroinformatics, October 7-11, 2019, Dolgoprudny, Moscow Region, Russia. — Springer. 2020. — с. 62—70.

24. Learning embodied agents with policy gradients to navigate in realistic environments / A. Staroverov [и др.] // Advances in Neural Computation, Machine Learning, and Cognitive Research IV: Selected Papers from the XXII International Conference on Neuroinformatics, October 12-16, 2020, Moscow, Russia. — Springer. 2021. — с. 212—221.

25. HPointLoc: Point-Based Indoor Place Recognition Using Synthetic RGB-D Images / D. Yudin [и др.] // Neural Information Processing: 29th International Conference, ICONIP 2022, Virtual Event, November 22–26, 2022, Proceedings, Part III. — Springer. 2023. — с. 471—484.

26. Successor Feature Landmarks for Long-Horizon Goal-Conditioned Reinforcement Learning / C. Hoang [и др.]. — 2021. — arXiv: 2111.09858 [cs.LG].

27. Real-time object navigation with deep neural networks and hierarchical reinforcement learning / A. Staroverov [и др.] // IEEE Access. — 2020. — т. 8. — с. 195608—195621.

28. *Staroverov A.*, *Panov A. I.* Hierarchical landmark policy optimization for visual indoor navigation // IEEE Access. — 2022. — т. 10. — с. 70447—70455.

29. Skill Fusion in Hybrid Robotic Framework for Visual Object Goal Navigation / A. Staroverov [и др.] // Robotics. — 2023. — т. 12, № 4. — с. 104.

30. *Hart P. E.*, *Nilsson N. J.*, *Raphael B.* A formal basis for the heuristic determination of minimum cost paths // IEEE transactions on Systems Science and Cybernetics. — 1968. — т. 4, № 2. — с. 100—107.

31. Implementation of nonlinear model predictive path-following control for an industrial robot / T. Faulwasser [и др.] // IEEE Transactions on Control Systems Technology. — 2016. — т. 25, № 4. — с. 1505—1511.

32. *Soetanto D.*, *Lapierre L.*, *Pascoal A.* Adaptive, non-singular path-following control of dynamic wheeled robots // 42nd IEEE international conference on decision and control (IEEE Cat. No. 03CH37475). т. 2. — IEEE. 2003. — с. 1765—1770.

33. Model predictive path following control for autonomous cars considering a measurable disturbance: Implementation, testing, and verification / H. Guo [и др.] // Mechanical Systems and Signal Processing. — 2019. — т. 118. — с. 41—60.

34. Vision-and-Language Navigation: Interpreting visually-grounded navigation instructions in real environments / P. Anderson [и др.]. — 2018. — arXiv: 1711.07280 [cs.CV].

35. Embodied Question Answering / A. Das [и др.]. — 2017. — arXiv: 1711.11543 [cs.CV].

36. On Evaluation of Embodied Navigation Agents / P. Anderson [и др.]. — 2018. — arXiv: 1807.06757 [cs.AI].

37. OpenAI Gym / G. Brockman [и др.]. — 2016. — arXiv: 1606.01540 [cs.LG].

38. The Arcade Learning Environment: An Evaluation Platform for General Agents / M. G. Bellemare [и др.] // Journal of Artificial Intelligence Research. — 2013. — июнь. — т. 47. — с. 253—279. — DOI: 10.1613/jair.3912. — URL: https://doi.org/10.1613%2Fjair.3912.

39. *Todorov E.*, *Erez T.*, *Tassa Y.* MuJoCo: A physics engine for model-based control // 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. — 2012. — с. 5026—5033. — DOI: 10.1109/IROS.2012.6386109.

40. OpenSpiel: A Framework for Reinforcement Learning in Games / M. Lanctot [и др.]. — 2020. — arXiv: 1908.09453 [cs.LG].

41. ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning / M. Kempka [и др.]. — 2016. — arXiv: 1605.02097 [cs.LG].

42. DeepMind Lab / C. Beattie [и др.]. — 2016. — arXiv: 1612.03801 [cs.AI].

43. Matterport3D: Learning from RGB-D Data in Indoor Environments / A. Chang [и др.] // International Conference on 3D Vision (3DV). — 2017.

44. Interactive Gibson Benchmark: A Benchmark for Interactive Navigation in Cluttered Environments / F. Xia [и др.] // IEEE Robotics and Automation Letters. — 2020. — апр. — т. 5, № 2. — с. 713—720. — DOI: 10.1109/lra.2020.2965078. — URL: https://doi.org/10.1109%2Flra.2020.2965078.

45. Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI / S. K. Ramakrishnan [и др.]. — 2021. — DOI: 10.48550/ARXIV.2109.08238. — URL: https://arxiv.org/abs/2109.08238.

46. The Replica Dataset: A Digital Replica of Indoor Spaces / J. Straub [и др.]. — 2019. — arXiv: 1906.05797 [cs.CV].

47. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes / A. Dai [и др.]. — 2017. — arXiv: 1702.04405 [cs.CV].

48. Proximal Policy Optimization Algorithms / J. Schulman [и др.]. — 2017. — arXiv: 1707.06347 [cs.LG].

49. *Achiam J.* OpenAI Spinning Up. — 2018. — URL: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html (дата обр. 18.06.2023).

50. *Zhang H.*, *Yu T.* AlphaZero // Deep Reinforcement Learning: Fundamentals, Research and Applications. — 2020. — с. 391—415.

51. *Fu M. C.* AlphaGo and Monte Carlo tree search: the simulation optimization perspective // 2016 Winter Simulation Conference (WSC). — IEEE. 2016. — с. 659—670.

52. Playing Atari with Deep Reinforcement Learning / V. Mnih [и др.]. — 2013. — arXiv: 1312.5602 [cs.LG].

53. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor / T. Haarnoja [и др.]. — 2018. — arXiv: 1801.01290 [cs.LG].

54. Asynchronous Methods for Deep Reinforcement Learning / V. Mnih [и др.]. — 2016. — arXiv: 1602.01783 [cs.LG].

55. Trust Region Policy Optimization / J. Schulman [и др.]. — 2017. — arXiv: 1502.05477 [cs.LG].

56. *Dayan P.*, *Hinton G. E.* Feudal reinforcement learning // Advances in neural information processing systems. — 1992. — т. 5.

57. *Rasmussen D.*, *Voelker A.*, *Eliasmith C.* A neural model of hierarchical reinforcement learning // PloS one. — 2017. — т. 12, № 7. — e0180234.

58. *Hengst B.* Hierarchical Approaches // Reinforcement Learning: State-of-the-Art / под ред. M. Wiering, M. van Otterlo. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2012. — с. 293—323. — ISBN 978-3-642-27645-3. — DOI: 10.1007/978-3-642-27645-3_9. — URL: https://doi.org/10.1007/978-3-642-27645-3_9.

59. Hindsight experience replay / M. Andrychowicz [и др.] // Advances in neural information processing systems. — 2017. — т. 30.

60. *Wijmans E.*, *Essa I.*, *Batra D.* How to Train PointGoal Navigation Agents on a (Sample and Compute) Budget. — 2020. — arXiv: 2012.06117 [cs.CV].

61. Deep Residual Learning for Image Recognition / K. He [и др.]. — 2015. — arXiv: 1512.03385 [cs.CV].

62. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling / J. Chung [и др.]. — 2014. — arXiv: 1412.3555 [cs.NE].

63. SplitNet: Sim2Sim and Task2Task Transfer for Embodied Visual Navigation / D. Gordon [и др.]. — 2019. — DOI: 10.48550/ARXIV.1905.07512. — URL: https://arxiv.org/abs/1905.07512.

64. *Gadzicki K.*, *Khamsehashari R.*, *Zetzsche C.* Early vs late fusion in multimodal convolutional neural networks // 2020 IEEE 23rd International Conference on Information Fusion (FUSION). — IEEE. 2020. — с. 1—6.

65. Offline Visual Representation Learning for Embodied Navigation / K. Yadav [и др.]. — 2022. — DOI: 10.48550/ARXIV.2204.13226. — URL: https://arxiv.org/abs/2204.13226.

66. Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos / B. Baker [и др.]. — 2022. — DOI: 10.48550/ARXIV.2206.11795. — URL: https://arxiv.org/abs/2206.11795.

67. Learning Transferable Visual Models From Natural Language Supervision / A. Radford [и др.]. — 2021. — DOI: 10.48550/ARXIV.2103.00020. — URL: https://arxiv.org/abs/2103.00020.

68. Simple but Effective: CLIP Embeddings for Embodied AI / A. Khandelwal [и др.]. — 2021. — DOI: 10.48550/ARXIV.2111.09888. — URL: https://arxiv.org/abs/2111.09888.

69. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation / M. Deitke [и др.]. — 2022. — DOI: 10.48550/ARXIV.2206.06994. — URL: https://arxiv.org/abs/2206.06994.

70. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam / C. Campos [и др.] // IEEE Transactions on Robotics. — 2021. — т. 37, № 6. — с. 1874—1890.

71. Real-time loop closure in 2D LIDAR SLAM / W. Hess [и др.] // 2016 IEEE international conference on robotics and automation (ICRA). — IEEE. 2016. — с. 1271—1278.

72. *Labbé M.*, *Michaud F.* RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation // Journal of Field Robotics. — 2019. — т. 36, № 2. — с. 416—446.

73. *Santosh D.*, *Achar S.*, *Jawahar C.* Autonomous image-based exploration for mobile robot navigation // 2008 IEEE International Conference on Robotics and Automation. — IEEE. 2008. — с. 2717—2722.

74. An improved frontier-based approach for autonomous exploration / W. Gao [и др.] // 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV). — IEEE. 2018. — с. 292—297.

75. *Muravyev K.*, *Bokovoy A.*, *Yakovlev K.* Enhancing exploration algorithms for navigation with visual SLAM // Russian Conference on Artificial Intelligence. — Springer. 2021. — с. 197—212.

76. *Kojima N.*, *Deng J.* To Learn or Not to Learn: Analyzing the Role of Learning for Navigation in Virtual Environments. — 2019. — DOI: 10.48550/ARXIV.1907.11770. — URL: https://arxiv.org/abs/1907.11770.

77. *Mishkin D.*, *Dosovitskiy A.*, *Koltun V.* Benchmarking Classic and Learned Navigation in Complex 3D Environments. — 2019. — DOI: 10.48550/ARXIV.1901.10915. — URL: https://arxiv.org/abs/1901.10915.

78. Cognitive Mapping and Planning for Visual Navigation / S. Gupta [и др.]. — 2017. — DOI: 10.48550/ARXIV.1702.03920. — URL: https://arxiv.org/abs/1702.03920.

79. PONI: Potential Functions for ObjectGoal Navigation with Interaction-free Learning / S. K. Ramakrishnan [и др.]. — 2022. — arXiv: 2201.10029 [cs.CV].

80. Bayesian Controller Fusion: Leveraging Control Priors in Deep Reinforcement Learning for Robotics / K. Rana [и др.]. — 2023. — arXiv: 2107.09822 [cs.RO].

81. *Kim J.*, *Seo Y.*, *Shin J.* Landmark-Guided Subgoal Generation in Hierarchical Reinforcement Learning. — 2021. — arXiv: 2110.13625 [cs.LG].

82. *Alatise M. B.*, *Hancke G. P.* A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods // IEEE Access. — 2020. — т. 8. — с. 39830—39846.

83. *Jadidi M. G.*, *Miro J. V.*, *Dissanayake G.* Gaussian processes autonomous mapping and exploration for range-sensing mobile robots // Autonomous Robots. — 2018. — т. 42, № 2. — с. 273—290.

84. *Fang B.*, *Ding J.*, *Wang Z.* Autonomous robotic exploration based on frontier point optimization and multistep path planning // IEEE Access. — 2019. — т. 7. — с. 46104—46113.

85. *Al Khatib E. I.*, *Jaradat M. A. K.*, *Abdel-Hafez M. F.* Low-Cost Reduced Navigation System for Mobile Robot in Indoor/Outdoor Environments // IEEE Access. — 2020. — т. 8. — с. 25014—25026.

86. *Russell S.*, *Norvig P.* Artificial Intelligence: A Modern Approach. — 4th. — USA : Pearson, 2020. — ISBN 0134610997.

87. ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects / D. Batra [и др.] // arXiv:2006.13171. — 2020.

88. A Benchmark for the Evaluation of RGB-D SLAM Systems / J. Sturm [и др.] // Proc. of the International Conference on Intelligent Robot Systems (IROS). — 10.2012.

89. Mask r-cnn / K. He [и др.] // Proceedings of the IEEE international conference on computer vision. — 2017. — с. 2961—2969.

90. Yolact++: Better real-time instance segmentation / D. Bolya [и др.] // arXiv preprint arXiv:1912.06218. — 2019.

91. Deep Snake for Real-Time Instance Segmentation / S. Peng [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2020. — с. 8533—8542.

92. Polarmask: Single shot instance segmentation with polar representation / E. Xie [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2020. — с. 12193—12202.

93. Polytransform: Deep polygon transformer for instance segmentation / J. Liang [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2020. — с. 9131—9140.

94. The Pascal Visual Object Classes (VOC) Challenge / M. Everingham [и др.] // International Journal of Computer Vision. — 2010. — июнь. — т. 88, № 2. — с. 303—338.

95. DXSLAM: A Robust and Efficient Visual SLAM System with Deep Features / D. Li [и др.]. — 2020. — arXiv: 2008.05416 [cs.CV].

96. From Coarse to Fine: Robust Hierarchical Localization at Large Scale / P.-E. Sarlin [и др.]. — 2018. — arXiv: 1812.03506 [cs.CV].

97. *Geiger A.*, *Lenz P.*, *Urtasun R.* Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite // Conference on Computer Vision and Pattern Recognition (CVPR). — 2012.

98. Visual Odometry Revisited: What Should Be Learnt? / H. Zhan [и др.] // arXiv preprint arXiv:1909.09803. — 2019.

99. *Grupp M.* evo: Python package for the evaluation of odometry and SLAM. — 2017. — https://github.com/MichaelGrupp/evo.

100. Dueling Network Architectures for Deep Reinforcement Learning / Z. Wang [и др.]. — 2016. — arXiv: 1511.06581 [cs.LG].

101. *Fujimoto S.*, *Hoof H. van*, *Meger D.* Addressing Function Approximation Error in Actor-Critic Methods. — 2018. — arXiv: 1802.09477 [cs.AI].

102. Detectron2 / Y. Wu [и др.]. — 2019. — https://github.com/facebookresearch/detectron2.

103. MMDetection: Open MMLab Detection Toolbox and Benchmark / K. Chen [и др.] // arXiv preprint arXiv:1906.07155. — 2019.

104. Are We Ready for Service Robots? The OpenLORIS-Scene Datasets for Lifelong SLAM / X. Shi [и др.] // 2020 International Conference on Robotics and Automation (ICRA). — 2020. — с. 3139—3145.

105. Exploration by Random Network Distillation / Y. Burda [и др.]. — 2018. — arXiv: 1810.12894 [cs.LG].

106. Object Goal Navigation using Goal-Oriented Semantic Exploration / D. S. Chaplot [и др.]. — 2020. — arXiv: 2007.00643 [cs.CV].

107. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? / A. Kadian [и др.] // IEEE Robotics and Automation Letters. — 2020. — окт. — т. 5, № 4. — с. 6670—6677. — DOI: 10.1109/lra.2020.3013848. — URL: https://doi.org/10.1109%2Flra.2020.3013848.

108. MINOS: Multimodal Indoor Simulator for Navigation in Complex Environments / M. Savva [и др.]. — 2017. — arXiv: 1712.03931 [cs.LG].

109. Joint 2D-3D-Semantic Data for Indoor Scene Understanding / I. Armeni [и др.]. — 2017. — arXiv: 1702.01105 [cs.CV].

110. *Mishkin D.*, *Dosovitskiy A.*, *Koltun V.* Benchmarking Classic and Learned Navigation in Complex 3D Environments. — 2019. — arXiv: 1901.10915 [cs.CV].

111. ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects / D. Batra [и др.]. — 2020. — arXiv: 2006.13171 [cs.CV].

112. Are we making real progress in simulated environments? measuring the sim2real gap in embodied visual navigation / A. Kadian [и др.]. — 2019.

113. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age / C. Cadena [и др.] // IEEE Transactions on robotics. — 2016. — т. 32, № 6. — с. 1309—1332.

114. *Fugal J.*, *Bae J.*, *Poonawala H. A.* On the Impact of Gravity Compensation on Reinforcement Learning in Goal-Reaching Tasks for Robotic Manipulators // Robotics. — 2021. — т. 10, № 1. — DOI: 10.3390/robotics10010046. — URL: https://www.mdpi.com/2218-6581/10/1/46.

115. Mastering Atari, Go, chess and shogi by planning with a learned model / J. Schrittwieser [и др.] // Nature. — 2020. — дек. — т. 588, № 7839. — с. 604—609. — DOI: 10.1038/s41586-020-03051-4. — URL: https://doi.org/10.1038%2Fs41586-020-03051-4.

116. *Rozenberszki D.*, *Majdik A. L.* LOL: Lidar-only odometry and localization in 3D point cloud maps // 2020 IEEE International Conference on Robotics and Automation (ICRA). — IEEE. 2020. — с. 4379—4385.

Список рисунков

Список таблиц