ВВЕДЕНИЕ

В нашу эпоху, когда практически каждый является предпринимателем, имеет своё дело, будь то малый, средний или крупный бизнес, основной целью таких людей становится, естественно, сокращение расходов и увеличение прибыли. В рамках данной концепции остро стоит вопрос: каковы способы реализации подобных амбиций?

Одной из основных статей расходов является сотрудник компании. В поддержание его работоспособности вкладывается достаточное количество денежных ресурсов:

1) Заработная плата;

2) Больничные;

3) Отпускные;

4) Декретный отпуск (когда речь идёт о женщинах);

5) Потребляемая электроэнергия:

6) Обеспечение офиса водой:

7) Рабочие станции;

8) Офисные принадлжености;

9) Тренинги и другие обучающие мероприятия;

10) Корпоративы.

При уменьшении заработной платы, вероятнее всего, сотрудник предпочтёт сменить место работы на более высокооплачиваемое (уйдёт к конкурентам). При сокращении штата возможен упадок производительности. Выходом из положения работодатели видят замену сотрудников-людей на гуманоидных роботов. В виду того, что робот:

1) Не подвержен болезням;

2) Не планирует создавать семью;

3) Не нуждается в отпуске;

4) Не нуждается в ежемесячной оплате труда;

5) Не нуждается в воде и освещении;

6) Не нуждается в многократном обучении, такие сотрудники рассматриваются работодателями как более выгодные. Безусловно, это не отменяет того, что робот будет всё так же расходовать электричество и будет нуждаться в технической поддержке. Несмотря на это, такая замена персонала рассматривается как выгодная сделка. В условиях актуальности данной темы команде компании ООО НПФ «Микро Вейв» была поставлена цель собрать и разработать такого робота, который бы мог решать следующие задачи:

1) Вербальное общение на русском языке

2) Движение по относительно гладкой и ровной поверхности

3) Движение по заданному маршруту

4) Движение по оптимально выбранному маршруту

5) Объезд неподвижных препятствий

6) Остановка перед внезапно возникшим препятствием

7) Жестикуляция

8) Демонстрация необходимой информации на встроенном экране Данные задачи сформулированы и обозначены на сайте компании, посвящённом роботу, разрабатываемому командой программистов. В виду того, что данные задачи были модульно распределены между разработчиками, в данной научно-исследовательской работе будут рассматриваться решения только для задач, включающих в себя ориентирование на местности и жестикуляцию.

ИССЛЕДОВАНИЕ ОБЛАСТИ РОБОТОТЕХНИКИ И ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Говоря о робототехнике, нельзя не вспомнить о том, что данная отрасль тесно связана с отраслью, занимающейся исследованием и разработкой искусственного интеллекта. Искусственный интеллект – это интеллект, демонстрируемый техникой. В информатике область исследования искусственного интеллекта определяется как наука об «интеллектуальных агентах», т.е., наука о таких устройствах, которые воспринимают окружающую их среду и совершают действия, которые увеличивают шанс этих устройств в достижении поставленной цели. Говоря коротко, термин «искусственный интеллект» применяется, когда техника имитирует когнитивные функции, которые у людей ассоциируются с человеческим разумом, такие, как «обучение» и «самостоятельное решение задач». По мере того, как техника становится всё более и более способной, умственные способности, которые когда-то считались требующими от машин «интеллекта», исчезают из этого определения. Например, оптическое распознавание символов больше не воспринимается как пример искусственного интеллекта – теперь это обычная технология. Возможности, классифицируемые в настоящее время как искусственный интеллект, включают в себя:

1) Успешное распознавание человеческой речи;

2) Системы, которые участвуют в состязаниях с игроками высокого уровня, например, шахматные состязания;

3) Самоуправляемые автомобили;

4) Интеллектуальные сети доставки контента;

5) Военные симуляторы;

6) Интерпретация сложных данных.

Исследования в сфере искусственного интеллекта подразделяются на подотрасли, которые фокусируются на специфических задачах, подходах, использовании определённого инструментария или удовлетворении конкретных приложений. Основными задачами и целями исследования ИИ являются:

1) Рассуждение;

2) Знание;

3) Планирование;

4) Обучение;

5) Обработка естественного языка (общение);

6) Восприятие окружающей среды;

7) Способность перемещать и взаимодействовать с объектами. Так называемый «общий интеллект» относится к долгосрочным целям рассматриваемой области. В используемые исследованием подходы входят:

1) Статистический;

2) Вычислительный интеллект;

3) Традиционный «символьный» ИИ.

В сфере ИИ используется большое количество инструментариев, в том числе:

1) Информированный поиск;

2) Математические оптимизации;

3) Логика;

4) Вероятностные методы;

5) Экономика. Знания, методы и решения в отрасль ИИ опираются на информатику, математику, психологию, лингвистику, философию, неврлогию, искусственную психологию и многие другие. 14 Отрасль была основана на утверждении, что человеческий разум может быть «описан настолько детально и точно, что можно даже создать машину, которая сможет имитировать его». Что поднимает философские размышления и доводы о природе ума и этике создания искусственных существ, наделённых человекоподобным интеллектом, проблемах, которые были изучены в мифологии, домыслах и философии ещё в античности. Некоторые люди также считают, что если искусственный интеллект будет прогрессировать с огромной скоростью, то он может стать опасностью для человечества. В двадцать первом веке методы искусственного интеллекта, как аппаратные (с использованием символического подхода), так и программные (субсимволические), пережили возрождение после одновременных достижений в области вычислительной техники, величины обучающих наборов и теоретического осмысления. Вместе с тем данные методы ИИ становятся неотъемлемой частью технологической индустрии, помогая решать многие сложные задачи в информатике. Относительно недавние успехи в сфере искусственного интеллекта и, особенно, в машинной обработке, способствовали росту автономных процессов, таких как беспилотные самолёты и самоуправляемые автомобили, став главным двигателем инноваций в автомобильной промышленности. Общая исследовательская цель искусственного интеллекта – создать технологию, которая позволит компьютерам и машинам функционировать разумным образом, т.е., самостоятельно и рационально. Основная задача имитирования (или создания) интеллекта была разбита на подзадачи. Они состоят из определённых черт или возможностей, которые, по мнению исследователей, должны быть присущи интеллектуальным системам. Вот лишь некоторые из них, которые получили наибольшее внимание.

1.1 Задачи, стоящие перед разработчиками гуманоидных роботов

1.1.1 Рассуждение, решение проблем Исследователи-первопроходцы разработали алгоритмы, имитирующие пошаговое мышление, которые люди используют при решении головоломок, например, собирание мозаики, или делают логические выводы. К концу 80-ых и 90-ых исследователи искусственного интеллекта разработали методы для решения задач с неточной или неполной информацией, используя концепции из теории вероятностей и экономики. Для сложных задач алгоритмы могут потребовать огромные вычислительные ресурсы – большинство из них испытывают комбинаторный взрыв: объём требуемой памяти или времени вычсиления становится несоизмеримо большим для задач определённого размера. Таким образом, поиск более эффективных алгоритмов решения проблем становится высокоприоритетным. Обычно человек использует быстрые интуитивные суждения вместо пошаговых рассуждений, которые были изначально смоделированы исследователями искусственного интеллекта. ИИ добился прогресса, используя для решения задачи субсимвольный метод: подходы «воплощённых агентов» подчёркивают важность сенсомоторных навыков к более высоким рассуждениям; исследовательские нейронные сети пытаются моделировать внутри мозга структуры, которые дают начало новому навыку; статистические подходы искусственного интеллекта имитируют человеческие способности.

1.1.2 Представление знаний Представление знаний и проектирование знаний являются основными для исследования искусственного интеллекта. Многие из задач, которые должна решить техника, потребуют обширных знаний о мире. Среди сущностей, которые ИИ должен будет распознавать, находятся:

1) Объекты;

2) Свойства;

3) Категории;

4) Связи между объектами;

5) Ситуации;

6) События;

7) Положение в пространстве и времени;

8) Причины и следствия;

9) Знания о знаниях (иначе говоря, что мы знаем о том, что знают другие);

10) Другие, менее изученные области.

Представление «того, что существует» является онтологией – множеством объектов, отношений, концепций и проч., о которых знает устройство. Самые общие называются верхними онтологиями, которые пытаются обеспечить основу для всех других знаний. Среди самых трудоёмких задач в воспроизведении знаний выделяют: 1) Рассуждения по умолчанию и проблемы квалификации; 2) Обширность базы знаний; 3) Субсимволическая форма некоторых концепций из базы знаний.

1.1.3 Рассуждения по умолчанию и проблемы квалификации Многое из того, что известно человеку, принимает форму «рабочих предположений». Например, если в разговоре фигурирует птица, люди обычно представляют: «животное», «размером с кулак», «которое поёт» и «которое умеет летать». Ни одно из этих свойств не относится ко всем без исключения птицам. Джон Маккарти обозначил эту проблему в 1969 году как «проблему квалификации»: для любого правила здравого смысла, которое стараются воспроизвести исследователи ИИ, существует огромное количество исключений. Практически ничто не является просто истинным или ложным в том смысле, в котором требуется абстрактная логика. Исследование ИИ изучили ряд решений этой задачи.

1.1.4 Обширность базы знаний Число атомарных фактов, которые знает среднестатистический человек, достаточно велико. Исследовательские проекты, которые пытаются построить полную базу знаний, основанных на здравом смысле, требуют огромных объёмов кропотливой работы онтологической инженерии: она, база знаний, должна быть построена вручную (одна полная концепция за раз). Основная цель состоит в том, чтобы компьютер понимал достаточно концепций, чтобы иметь возможность обучиться, читая такие источники, как Интернет, и, таким образом, быть способным добавлять свою собственную онтологию.

1.1.5 Субсимволическая форма некоторых концепций из базы знаний Значительная часть того, что известно человеку, не представлена «фактами» или «утверждениями», которые можно выразить в устной форме. Например, игрок в шахматы избежит конкретной шахматной позиции, потому что он будет чувствовать себя в ней «слишком уязвимым». Или искусствовед может взглянуть на статую и понять, что это подделка. Эти чувства известны как «интуиция» и «тенденция», которые представлены в мозге неосознанно и субсимвольно. Знания, подобные этому, сообщают, поддерживают и обеспечивают контекст для символьного, осознанного утверждения. На данном этапе есть надежда, что вычислительный или статистический искусственные интеллекты предоставят способы представления такого рода знаний.

1.1.6 Планирование Интеллектуальные агенты должны быть способны устанавливать цели и добиваться их. Им необходим способ визуализации будущего, иными словами, они должны иметь представление о состоянии мира и меть возможность делать выводы о том, как их действия будут его изменять. Также такие агенты должны иметь возможность делать выбор, который максимизирует полезность (или «ценность») доступных вариантов воздействия на окружающую среду. В классических задачах планирования агент может предположить, что он является единственным действующим актёром, а также может быть уверен в том, каковы будут его последствия его действий. Однако, если агент не является единственным актёром, он должен периодически проверять, соответствует ли мир его прогнозам, и должен изменить свой порядок действий, как только это становится необходимо ввиду требования того, чтобы агент рассуждал в условиях неопределённости. «Многоагентное планирование» использует сотрудничество и конкуренцию многих агентов для достижения заданной цели. Эмерджентное поведение, подобное этому, используется эволюционными алгоритмами и интеллектом «пчелиного роя».

1.1.7 Обучение Машинное обучение – это исследование компьютерных алгоритмов, которые автоматически обновляются с помощью опыта и занимают центральное место в исследованиях ИИ с момента создания отрасли. Неконтролируемое обучение – это способность находить шаблоны во входящем потоке. Контролируемое обучение включает в себя как классификацию, так и численную регрессию. Классификация используется для определения того, к какой категории принадлежит что-либо, после просмотра ряда примеров сущностей из нескольких категорий. Регрессия – это попытка создать функцию, которая описывает взаимосвязь между входящим потоком и потоком на выходе и прогнозирует, как должны изменяться выходные данные при изменении входных параметров. В обучении с подкреплением агент «вознаграждается» за правильные ответы и «наказывается» за неудовлетворительные. Агент использует эту последовательность наград и наказаний для формирования стратегии при работе в области поставленной задачи. Эти три типа обучения могут быть проанализированы с точки зрения теории принятия решений, используя такие понятия, как «полезность». Математический анализ алгоритмов машинного обучения и их производительность – это раздел теоретической информатики, известный как теория вычислительного обучения. В рамках развивающейся робототехники были разработаны подходы к обучению в области развития для непрерывного кумулятивного приобретения наборов новых навыков роботом путём автономного самостоятельного исследования и социального взаимодействия с учителями-людьми и с использованием таких руководящих механизмов, как активное обучение, развитие, двигательное взаимодействие и имитация.

1.1.8 Обработка естественного языка (общение) Обработка естественного языка дает технике возможность читать и понимать языки, на которых говорят люди. Достаточно мощная система обработки естественного языка обеспечит возможность использования пользовательских интерфейсов на естественном языке и получения знаний непосредственно из источников, написанных человеком, таких как тексты новостей. Некоторые простые применения обработки на естественном языке включают извлечение информации, интеллектуальный анализ текста, ответы на вопросы и машинный перевод. Общим методом обработки и извлечения смысла из естественного языка является семантическая индексация. Увеличение скорости обработки и снижение затрат на хранение данных значительно повышает эффективность индексирования больших объёмов абстракций, присутствующих во введённых пользователем данных. Интерпретация непрерывного потока звуков, исходящих от человека, в реальном времени является сложной задачей для компьютера, главным образом из-за большой вариабельности речи. Одно и то же слово, произнесенное одним и тем же человеком, может звучать по-разному в зависимости от местной акустики, громкости, предыдущего слова и т.д. Всё становится еще сложнее, когда у говорящего есть акцент. Тем не менее, большие успехи были достигнуты в этой области с тех пор, как Дэвис, Биддульф и Балашек разработали первую «систему голосового ввода», которая в 1952 году распознала «десять символов, произнесённых одним пользователем, со стопроцентной точностью». В настоящее время наилучшие системы могут распознавать непрерывную естественную речь со скоростью до 160 слов в минуту с точностью до 95%.

1.1.9 Восприятие Машинное восприятие – это возможность использовать входящие сигналы от датчиков (таких как камеры, микрофоны, тактильные датчики, гидролокаторы и др.) для вывода аспектов окружающей среды. Компьютерное зрение – это способность анализировать визуальный входные данные. Данная подзадача в сфере исследования и разработки искусственного интеллекта разбивается ещё на несколько подзадач: распознавание речи, распознавание лиц и распознавание объектов.

1.1.10 Социальный интеллект Аффективные вычисления – это изучение и разработка систем и устройств, которые могут распознавать, интерпретировать, обрабатывать и моделировать человеческие эмоции. Это междисциплинарная область, охватывающая информатику, психологию и когнитивную науку. И хотя истоки этой области можно проследить еще до ранних философских исследований эмоций, более современная отрасль информатики впервые упоминается в 1995 году в статье Розалинды Пикард об аффективных вычислениях. Толчком для исследования стала способность имитировать эмпатию. Иными словами, машина должна интерпретировать эмоциональное состояние людей и адаптировать свое поведение к ним, давая соответствующий ответ на эти эмоции.

Роботизированные лица были созданы Hanson Robotics с использованием их эластичного полимера под названием Frubber, что позволяет использовать большое количество выражений лица благодаря эластичности резинового лицевого покрытия и встроенных подповерхностных двигателей (сервоприводов). Покрытие и сервоприводы закрепляются на металлическом черепе. Робот должен знать, как подойти к человеку, судя по их выражению лица и языку тела. Является ли человек счастливым, напуганным или сумасшедшим, влияет на тип взаимодействия, ожидаемого от робота. Подобным же образом роботы, такие как Kismet и более недавнее дополнение, Nexi могут производить ряд выражений лица, позволяя ему иметь значимые социальные контакты с людьми. Эмоции и социальные навыки играют две роли для интеллектуального агента. Во-первых, он должен уметь предсказывать действия других, понимая их мотивы и эмоциональные состояния, что включает в себя элементы теории игр, теории принятия решений, а также способность моделировать человеческие эмоции и навыки восприятия для обнаружения эмоций. Кроме того, в целях содействия взаимодействию между человеком и компьютером интеллектуальная машина может захотеть проявить эмоции, даже если она на самом деле не испытывает их, для того, чтобы проявлять чувствительность к эмоциональной динамике поведения человека. Лаборатория социально-интеллектуальных машин Технологического института Джорджии изучает новые концепции обучающего взаимодействия с роботами. Целью проектов является социальный робот, который изучает целевые задачи при помощи демонстрации ему их человеком без предварительного знания концепций высокого уровня. Эти новые концепции основаны на низкоуровневых данных непрерывного датчика посредством неконтролируемого обучения, и цели задач впоследствии изучаются с использованием байесовского подхода. Эти концепции могут быть использованы для передачи знаний при решении последующих задач, что приведёт к более быстрому изучению этих задач.

1.1.11 Творческие способности Область исследования искусственного интеллекта также посвящена творчеству как теоретически (с философской и психологической точки зрения), так и практически (через конкретные реализации систем, которые генерируют результаты, которые можно считать творческими, или системы, которые идентифицируют и оценивают творчество). Также данная область вычислительных исследований тесно связана с областью искусственной интуиции и искусственного мышления.

1.1.12 Движение и взаимодействия с объектами Несмотря на то, что значительная часть роботов в настоящее время находится под контролем человека или работает в статической среде, растет интерес к роботам, которые могут работать автономно в динамической среде. Эти роботы требуют некоторой комбинации навигационного оборудования и программного обеспечения, чтобы пересечь их среду. В частности, непредвиденные события (например, люди и другие препятствия, которые не являются стационарными) могут вызвать проблемы или столкновения. Некоторые высокоразвитые роботы, такие как ASIMO и робот Meinü, имеют особенно хорошее навигационное оборудование и программное обеспечение для роботов. Кроме того, саморегулирующиеся автомобили, атвомобиль Эрнста Дикманнса и записи в DARPA Grand Challenge способны хорошо воспринимать окружающую среду и впоследствии принимать навигационные решения на основе этой информации. Большинство из этих роботов используют навигационное устройство GPS с путевыми точками вместе с радаром, иногда в сочетании с другими сенсорными данными, такими как лидар, видеокамеры и системы инерциального наведения для лучшей навигации между путевыми точками.

1.1.13 Общий искусственный интеллект Многие исследователи полагают, что их работа в конечном итоге будет включена в механизм с искусственным общим интеллектом, сочетающим все навыки, описанный ранее и превосходящие все (или почти все) человеческие способности. Некоторые из исследователей считают, что для такого проекта могут потребоваться антропоморфические функции, такие как искусственное сознание или искусственный мозг. Есть предположение, что для решения многих вышеперечисленных задач требуется именно завершённый общий искусственный интеллект. Например, даже простая, специфичная задача, такая как машинный перевод, требует, чтобы машина читала и записывала на обоих языках (нейролингвистическое программирование), следовала замыслу автора текста (понимала подоплёку текста), знала, о чём говорят (знание темы), и правдиво представляла замысел автора (социальный интеллект). Задача, такая как машинный перевод, считается «ИИ-полноценной». Чтобы механизмы могли достичь производительности на уровне человека, нужно решить все поставленные задачи.

1.2 Подходы Не существует единой объединяющей теории или парадигмы, которая является направляющей для исследований искусственного интеллекта. Исследователи расходятся во мнениях по многим вопросам. Некоторые из наиболее давних вопросов, оставшихся без ответа, таковы: должен ли искусственный интеллект имитировать естественный интеллект, изучая психологию или неврологию? Или человеческая биология не имеет отношения к исследованиям искусственного интеллекта так же, как и орнитология не имеет ничего общего с авиационной техникой? Можно ли описать интеллектуальное поведение с использованием простых, лаконичных принципов, таких как логика или оптимизация? Или это обязательно требует решения большого числа совершенно не связанных между собой проблем?

Можно ли воспроизвести разум, используя символы высокого уровня, похожие на слова и идеи? Или он требует «субсимвольной» обработки? Джон Хаугеланд, который придумал термин GOFAI (Good Old-Fashioned Artificial Intelligence – хороший консервативный искусственный интеллект), также предположил, что ИИ правильнее будет называть синтетическим интеллектом – термином, который с тех пор был принят некоторыми исследователями, не являющимися последователями идеи GOFAI. Стюарт Шапиро делит исследования ИИ на три подхода, которые он называет вычислительной психологией, вычислительной философией и информатикой. Вычислительная психология используется для создания компьютерных программ, которые имитируют человеческое поведение. Вычислительная философия используется для разработки адаптивного, свободно мыслящего компьютерного ума. Внедрение информатики служит цели создания компьютеров, которые могут выполнять задачи, которые раньше могли выполнить только люди. Вместе человеческое поведение, ум и действия составляют искусственный интеллект.

1.2.1 Кибернетика и симуляция мозга В 1940-х и 1950-х годах ряд разработчиков исследовал связь между неврологией, теорией информации и кибернетикой. Некоторые из них строили машины, которые использовали электронные сети для воспроизведения зачаточного интеллекта, такие как «черепахи» У. Грейта Уолтера и «Зверь» Джона Хопкинса. Вычислительная нейронаука изучает функцию мозга с точки зрения обработки информации свойствами структур, которые составляют нервную систему. Это междисциплинарная вычислительная наука, которая связывает различные области нейронауки, когнитивной науки и психологии с электротехникой, информатикой, математикой и физикой. Вычислительная нейронаука отличается от психологического коннекционизма и от изучения теорий дисциплин, таких как машинное обучение, нейронные сети и теория вычислительного обучения, в том смысле, 25 что она подчеркивает описания функциональных и биологически реалистичных нейронов (и нейронных систем) и их физиологии и динамики. Эти модели охватывают существенные особенности биологической системы в многочисленных пространственно-временных масштабах, от мембранных токов, белков и химической связи с колебаниями сети, столбчатой и топографической архитектурой, а также обучением и памятью. К 1960 году этот подход в значительной степени был отменен, хотя его элементы были возрождены в 1980-х годах.

1.2.2 Символьный подход Символический искусственный интеллект - это коллективное имя для всех методов исследования искусственного интеллекта, которые основаны на высокоуровневых «символических» (удобочитаемых человеком) представлениях о проблемах, логике и поиске. Символический ИИ был доминирующей парадигмой исследований ИИ с середины 1950-х годов до конца 80-х годов. Когда доступ к цифровым компьютерам стал возможен в середине 1950-х годов, исследователи ИИ стали изучать возможность того, что человеческий интеллект может быть сведен к символьной манипуляции. Исследование было сосредоточено в трех институтах: Университете Карнеги- Меллона, Стэнфорде и Массачусетском технологическом институте, и у каждого из них была своя стратегия по ведению исследований. Джон Хоглэнд назвал эти подходы к ИИ «старым добрым AI» или «GOFAI». В 60-е годы символьные подходы добились больших успехов в моделировании высокого уровня мышления в небольших демонстрационных программах. Подходы, основанные на кибернетике или нейронных сетях, были отброшены или отодвинуты на задний план. Исследователи 1960-х и 1970-х годов были убеждены, что при помощи символических подходов в конечном итоге удастся создать машину с искусственным общим интеллектом, и считали это своей целью. Наиболее успешной формой символического ИИ являются экспертные системы, которые используют сеть производственных правил. Производственные правила связывают символы в отношении, аналогичном оператору If-Then (если – то).

Экспертная система обрабатывает правила, чтобы делать выводы и определять, какая ей нужна дополнительная информация, то есть какие нужно задать вопросы, используя символы, которые сможет прочесть человек.

a) Когнитивное моделирование Экономист Герберт Симон и Аллен Ньюэлл изучали человечческие навыки решения задач и пытались их формализовать, а их работа заложила основы области искусственного интеллекта, а также когнитивную науку, исследования операций и науку управления. Их исследовательская группа использовала результаты психологических экспериментов для разработки программ, имитирующих методы, которые люди использовали для решения проблем. Эта традиция, сконцентрированная в Университете Карнеги- Меллона, в конечном итоге увенчалась развитием архитектуры Соар в середине 1980-х годов.

b) Подход, основанный на логике В отличие от Ньюэлла и Саймона, Джон Маккарти считал, что машинам не нужно моделировать человеческую мысль, но вместо этого он должен попытаться найти сущность абстрактного мышления и решения проблем, независимо от того, используют ли люди одни и те же алгоритмы. Его лаборатория в Стэнфорде сосредоточилась на использовании формальной логики для решения широкого круга проблем, включая представление знаний, планирование и обучение. Логика также была в центре внимания работы в Эдинбургском университете и в других странах Европы, что привело к разработке языка программирования Prolog и науки логического программирования.

c) Антилогический или «неряшливый» Исследователи из Массачусетского технологического института (такие как Марвин Мински и Сеймур Паперт) обнаружили, что решение сложных проблем в области зрения и обработки естественного языка требует специальных решений: они утверждали, что не существует простого и общего принципа (например, логики), который бы охватывал все Аспекты интеллектуального поведения. Роджер Шенк описал их «антилогичные» подходы как «неряшливые» (в отличие от «аккуратных» парадигм в Стэнфорде). «Разумные» базы знаний являются примером «неряшливого» ИИ, поскольку они должны быть построены вручную, по одному сложному понятию за раз.

d) Подход, основанный на знаниях Когда в 1970 году стали доступны компьютеры с большим объёмом памяти, исследователи из всех трех ранее описанных направлений исследований начали собирать знания в области применения ИИ. Эта «революция знаний» привела к разработке и внедрению экспертных систем (введенных Эдвардом Фейгенбаумом), первой действительно успешной формой программного обеспечения для ИИ. Революция знаний также была обусловлена осознанием того, что огромное количество знаний потребуется во многих простых приложениях ИИ.

1.2.3 Субсимвольный подход К 1980-м годам прогресс в символическом ИИ, казалось, заглох, и многие считали, что символические системы никогда не смогут имитировать все процессы человеческого познания, особенно восприятие, робототехнику, обучение и распознавание образов. Ряд исследователей начали изучать «субсимволические» подходы к конкретным проблемам искусственного интеллекта. Субсимвольные методы позволяют подойти к интеллекту без конкретных представлений о знании

a) Воплощённый интеллект Данный подход включает в себя воплощенный, «расположенный», поведенческий и совершенно новый ИИ. Исследователи из смежной области робототехники, такие как Родни Брукс, отвергли символический ИИ и сосредоточились на основных технических проблемах, которые позволят двигаться дальше и всё больше утверждаться развитию роботов. Их работа возродила несимволическую точку зрения ранних исследователей кибернетики 1950-х годов и вновь внесла использование теории управления в ИИ. Это совпало с развитием тезиса о воплощенном мышлении в связанной с ним области когнитивной науки: идея о том, что идея навыков человеческого тела (такие как движение, восприятие и визуализация) необходимы для более высокого интеллекта.

b) Вычислительный интеллект и программные вычисления Интерес к нейронным сетям и «коннекционизма» был возрожден Дэвидом Румельхартом и другими в середине 80-х годов. Нейронные сети являются примером программных вычислений: они являются решениями задач, которые не могут быть решены с полной логической определенностью, и задач, где часто бывает достаточно приближенного решения. Другие подходы к компьютерному искусственному интеллекту включают нечеткие системы, эволюционные вычисления и многие статистические инструменты. Применение программных вычислений к ИИ коллективно изучается появляющейся дисциплиной вычислительного интеллекта.

c) Машинное обучение и статистика В 1990-е годы исследователи ИИ разработали сложные математические инструменты для решения конкретных подзадач. Эти инструменты действительно научны в том смысле, что их результаты поддаются измерению и поддаются проверке, и именно благодаря этим инструментам стали возможны многие из последних успехов ИИ. Общий математический язык также позволил достичь высокого уровня взаимодействия с более 29 устоявшимися областями, такими как математика, экономика или исследование операций. Стюарт Рассел и Питер Норвиг описывают это движение как не что иное, как «революция» и «победа сетей». Критики утверждают, что эти методы, за немногими исключениями, слишком сфокусированы на конкретных проблемах и не смогли решить долгосрочную задачу общего интеллекта. Существует постоянная дискуссия об актуальности и валидности статистических подходов в ИИ, примером чему являются споры между Питером Норвигом и Ноамом Хомски.

1.2.4 Внедрение подходов

a) Парадигма интеллектуального агента Интеллектуальный агент – это система, которая воспринимает окружающую среду и предпринимает действия, которые максимизируют его шансы на успех в достижении поставленной цели. Простейшие интеллектуальные агенты – это программы, которые решают конкретные задачи. Более сложные агенты включают в себя также людей и организации людей (например, фирмы). Эта парадигма дает исследователям право изучать отдельные задачи и находить решения, которые поддаются проверке и полезны, не соглашаясь на один единственный подход. Агент, который решает конкретную задачу, может использовать любой подход, который работает: некоторые агенты символичны и логичны, некоторые – используют субсимвольные нейронные сети, а другие могут использовать новые подходы. Парадигма также даёт исследователям общий язык для общения с другими областями, такими как теория принятия решений и экономика, которые также используют концепции абстрактных агентов. Парадигма интеллектуального агента получила широкое признание в 1990-х годах.

b) Архитектура агентов и когнитивные архитектуры Исследователи разработали системы для построения интеллектуальных систем из взаимодействующих интеллектуальных агентов 30 в многоагентной системе. Система с символическими и субсимволическими компонентами представляет собой гибридную интеллектуальную систему, а изучение таких систем – это интеграция систем искусственного интеллекта. Иерархическая система управления обеспечивает мост между субсимволическим ИИ на самом низком, реактивном, уровне и традиционном символическом ИИ на его высших уровнях, где ослабленные временные ограничения позволяют планировать и моделировать мир. Предшествующая архитектура Родни Брукса была ранним предложением для такой иерархической системы.

1.3 Инструментарий За 50 лет исследований в сфере ИИ разработано большое количество инструментов для решения самых сложных проблем в информатике. Некоторые из наиболее общих из этих методов будут рассмотрены далее.

1.3.1 Поиск и оптимизация Многие проблемы в искусственном интеллекте могут быть решены теоретически путем интеллектуального поиска во многих возможных решениях: рассуждение может быть сведено к выполнению поиска. Например, логическое доказательство можно рассматривать как поиск пути, ведущего от предпосылок к выводам, где каждый шаг является применением правила вывода. Алгоритмы планирования выполняют поиск деревьев целей и подцелей, пытаясь найти путь к основной цели, процесс, называемый анализом средств-концов. Алгоритмы робототехники для перемещения конечностей и захвата объектов используют локальные поиски в конфигурационном пространстве. Многие алгоритмы обучения используют алгоритмы поиска, основанные на оптимизации. Простые исчерпывающие поиски редко бывают достаточными для решения большинства задач реального мира: пространство поиска (количество мест для поиска) быстро растет до астрономических чисел. В результате поиск выполняется слишком медленно или никогда не завершается. Решение для многих задач состоит в использовании «эвристики» или «эмпирических правил», которые исключают варианты, которые вряд ли приведут к цели (так называемая «обрезка дерева поиска»). Эвристика предоставляет программе «лучшее предположение» для пути, которое ведёт к решению. Эвристика ограничивает поиск решений меньшим размером выборки. В 90-е годы на основе математической теории оптимизации был сделан совершенно другой вид поиска. Для многих задач можно начать поиск с некоторой формы предположения, а затем постепенно уточнять детали до тех пор, пока не будут сделаны дополнительные уточнения. Эти алгоритмы могут быть визуализированы как слепое восхождение на холм: мы начинаем поиск в случайной точке на ландшафте, а затем, с помощью прыжков или шагов, продолжаем двигаться вверх, пока не достигнем вершины. Другими алгоритмами оптимизации являются «имитационный отжиг», «поиск луча» и случайная оптимизация. Эволюционное вычисление использует форму поиска оптимизации. Например, они могут начинаться с популяции организмов (догадок), а затем позволять им мутировать и рекомбинироваться, выбирая только наиболее приспособленных особей, чтобы выжить в каждом поколении (уточняя детали). Формы эволюционных вычислений включают в себя алгоритмы интеллекта роя, такие как колония муравьев или оптимизация роя частиц, и эволюционные алгоритмы, такие как генетические алгоритмы, программирование генной экспрессии и генетическое программирование.

1.3.2 Логика Логика используется для представления знаний и решения задач, но может быть применена и для достижения других целей. Например, в алгоритме SATPLAN (Planning as Satisfiability – метод автоматизированного планирования) используется логика для планирования, а индуктивное логическое программирование является методом обучения. 32 В исследованиях ИИ используются несколько различных форм логики. Логика высказываний – это логика утверждений, которые могут быть истинными или ложными. Логика первого порядка также допускает использование кванторов и предикатов и может выражать факты об объектах, их свойствах и их отношениях друг с другом. Нечёткая логика – это версия логики первого порядка, которая позволяет истинности утверждения быть представленным как значение между 0 и 1, а не просто «правдой» (1) или «ложью» (0). Нечёткие системы могут использоваться для неопределенных рассуждений и широко использовались в современных системах контроля промышленных и потребительских товаров. Субъективная логика моделирует неопределенность другим и более явным образом, чем нечёткая логика: данное биномиальное мнение удовлетворяет убеждению + неверие + неопределенность = 1 в бета-распределении. По этому методу незнание можно отличить от вероятностных утверждений, которые агент делает с высокой достоверностью. Логика по умолчанию, немонотонная логика и описание – это формы логики, разработанные для помощи в задачах, относящихся к рассуждениям по умолчанию и квалификации. Были разработаны несколько расширений логики для обработки конкретных областей знаний, таких как: логика описания; исчисление ситуации, исчисление событий и бысрое исчисление (для представления событий и времени); исчисление предпосылок; исчисление доверия и модальная логика.

1.3.3 Вероятностные методы для неточных рассуждений Многие задачи в сфере искусственного интеллекта (в рассуждении, планировании, обучении, восприятии и робототехнике) требуют, чтобы агент работал с неполной или неопределенной информацией. Исследователи ИИ разработали ряд мощных инструментов для решения этих проблем с использованием методов теории вероятностей и экономики. Байесовские сети – очень общий инструмент, который может быть использован для решения большого числа задач: рассуждение (с 33 использованием байесовского алгоритма вывода), обучение (с использованием алгоритма максимизации ожидания), планирование (с использованием решения сетей) и восприятие (с использованием динамических байесовских сетей). Вероятностные алгоритмы могут также использоваться для фильтрации, прогнозирования, сглаживания и нахождения пояснений для потоков данных, помогая системам восприятия анализировать процессы, которые происходят с течением времени (например, скрытые модели Маркова или фильтры Калмана). Ключевой концепцией экономики является «полезность»: мера того, насколько ценным является что-то для интеллектуального агента. Были разработаны точные математические инструменты, которые анализируют, как агент может выбирать и планировать, используя теорию решений, анализ решений, и теорию информационной ценности. Эти инструменты включают такие модели, как марковские процессы принятия решений, сети динамических решений, теорию игр и механизм проектирования.

1.3.4 Классификаторы и методы статистического обучения Простейшие приложения ИИ можно разделить на два типа: классификаторы («если блестит, значит, это бриллиант») и контроллеры («если блестит – подбери»). Контроллеры, однако, также классифицируют условия перед выводом действий, и поэтому классификация является центральной частью многих систем ИИ. Классификаторы – это функции, использующие сопоставление с образцом для определения ближайшего соответствия. Они могут быть настроены в соответствии с примерами, что делает их очень выгодными для использования в AI. Эти примеры известны как наблюдения или шаблоны («паттерны»). В контролируемом обучении каждый шаблон принадлежит конкретному предопределённому классу. Класс можно рассматривать как решение, которое должно быть принято. Все наблюдения, объединенные с метками их класса, известны как набор данных. Когда получено новое наблюдение, это наблюдение классифицируется на основе предыдущего опыта. 34 Классификатор может обучаться различными способами: существует много статистических и машинных подходов к обучению. Наиболее широко используемыми классификаторами являются нейронная сеть, ядерные методы, такие как вектор опорных машин, алгоритм k ближайшего соседа, модель гауссовой смеси, наивный байесовский классификатор и дерево решений. Производительность этих классификаторов была сравнена по широкому спектру задач. Производительность классификатора во многом зависит от характеристик подлежащих классификации данных. Нет единого классификатора, который лучше всего работает на всех обозначенных задачах. Определение подходящего классификатора для заданной задачи – всё же скорее искусство, чем наука.

1.3.5 Нейронные сети Изучение неисследованных искусственных нейронных сетей началось за десять лет до того, как была создана область исследований ИИ в работах Уолтера Питтса и Уоррена МакКалуша. Франк Розенблатт изобрел персептрон, обучающую сеть с одним слоем, подобную старой концепции линейной регрессии. К числу первых пионеров относятся также Алексей Григорьевич Ивахненко, Теуво Кохонен, Стивен Гроссберг, Кунихико Фукусима, Кристоф фон дер Мальсбург, Дэвид Уилшоу, Шунь-Ичи Амари, Бернард Уидроу, Джон Хопфилд, Эдуардо Р. Каяниелло и другие. Основными категориями сетей являются ациклические или направленные нейронные сети (где сигнал проходит только в одном направлении) и рекуррентные нейронные сети (которые позволяют получать обратную связь и кратковременные воспоминания о предыдущих входных событиях). К числу наиболее популярных сетей прямой связи относятся персептроны, многослойные персептроны и радиальные базисные сети. Нейронные сети могут быть применены к задаче интеллектуального управления (для робототехники) или обучения, используя такие методы, как обучение Hebbian, GMDH или конкурентное обучение. Сегодня нейронные сети часто обучаются по алгоритму обратного распространения, который примерно с 1970 года являлся обратным режимом автоматической дифференциации, опубликованный Seppo Linnainmaa, и был введён в нейронные сети Пол Вербосом. Иерархическая временная память – это подход, который моделирует некоторые структурные и алгоритмические свойства неокортекса.

1.3.6 Глубокие нейронные сети прямого распространения Глубокое изучение многоуровневых искусственных нейронных сетей трансформировало многие важные отрасли сферы искусственного интеллекта, включая компьютерное зрение, распознавание речи, обработку естественного языка и другие. Выражение «глубокое обучение» было представлено Риной Дектер в 1986 году в сообществе «Машинное обучение» и получило широкую огласку после того, как Игорь Айзенберг и его коллеги внедрили его в искусственные нейронные сети в 2000 году. Первые функциональные сети глубокого обучения были опубликованы Алексеем Григорьевичем Ивахненко и В. Г. Лапой в 1965 году. Эти сети проходят обучение по одному уровню за раз. В статье Ивахненко от 1971 года описывается изучение глубокого многослойного персептрона с упреждением с восемью слоями, уже намного более глубокого, чем многие последующие сети. В 2006 году публикация Джеффри Хинтона и Руслана Салахутдинова представила еще один способ предварительного обучения многослойных упреждающих нейронных сетей (FNN) одним слоем за раз, обрабатывая каждый слой поочередно как неконтролируемый ограниченный аппарат Больцмана, а затем используя контролируемое обратное распространение для тонкой настройки. Подобно мелким искусственным нейронным сетям, глубокие нейронные сети могут моделировать сложные нелинейные отношения. За последние несколько лет прогресс в алгоритмах машинного обучения и компьютерного оборудования привёл к более эффективным методам подготовки глубоких нейронных сетей, 36 которые содержат много уровней нелинейных скрытых модулей и очень большой выходной слой. Глубокое изучение часто использует свёрточные нейронные сети (CNN), происхождение которых можно заметить ещё аж в Неокогнитроне, введённом Кунихико Фукусимой в 1980 году. В 1989 году Янн ЛеКун и его коллеги применили обратное распространение к такой архитектуре. В начале 2000-х годов в промышленных приложениях CNN уже обрабатывали примерно 10-20% всех чеков, написанных в США. С 2011 года быстрые реализации CNN на графических процессорах выиграли много тендеров, посвящённых визуальному распознаванию образов. Глубокие упреждающие нейронные сети использовались в сочетании с обучением с подкреплением с помощью программы AlphaGo, Google Deepmind, которая была первой, кто победил профессионального игрока в игре «Go». 1.3.7 Глубокие рекуррентные нейронные На раннем этапе глубокое обучение также применялось для обучения последовательностям с помощью рекуррентных нейронных сетей (RNN), которые являются общими компьютерами и могут запускать произвольные программы для обработки произвольных последовательностей входных данных. Глубина RNN не ограничена и зависит от длины входной последовательности. RNN могут обучаться с помощью градиентного спуска, но страдают от нерешённой задачи исчезающего градиента. В 1992 году было показано, что неконтролируемая предварительная подготовка стека возвратных нейронных сетей может ускорить последующее контролируемое изучение глубоких последовательных проблем. Многочисленные исследователи теперь используют варианты глубоко обучающегося рекуррентного NN, названного сетью длинной кратковременной памяти (LSTM), опубликованной Hochreiter & Schmidhuber в 1997 году. LSTM часто обучается Временной Классификацией Коннекционизма – Connectionist Temporal Classification (CTC). В Google, 37 Microsoft и Baidu этот подход произвёл революцию в распознавании речи. Например, в 2015 году распознавание речи Google испытало резкий скачок в производительности на 49% благодаря LSTM, подготовленному CTC, который теперь доступен через Google Voice для миллиардов пользователей смартфонов. Google также использовал LSTM для улучшения машинного перевода, моделирования языков и многоязыковой обработки языков. LSTM в сочетании с CNN также улучшили автоматическое скрытие изображений и множество других приложений. 1.3.8 Теория контроля Теория управления, «внук» кибернетики, имеет много важных приложений, особенно в робототехнике. a) Контроллеры нейросетей Интеллектуальное управление – это класс методов управления, которые используют различные подходы к искусственному интеллекту, такие как нейронные сети, байесовская вероятность, нечеткая логика, машинное обучение, эволюционные вычисления и генетические алгоритмы. Нейронные сети были использованы для решения проблем практически во всех областях науки и техники. Контроль нейронной сети в основном включает два шага: идентификацию системы и непосредственный контроль. Показано, что сеть с прямой связью с нелинейными, непрерывными и дифференцируемыми функциями активации обладает универсальной аппроксимирующей способностью. Для идентификации системы также использовались повторяющиеся сети. Учитывая набор пар данных ввода- вывода, идентификация системы направлена на формирование отображения между этими парами данных. Предполагается, что такая сеть фиксирует динамику системы. b) Байесовские контроллеры Байесовская вероятность породила ряд алгоритмов, которые широко используются во многих современных системах управления, служащих 38 оценщиками пространства состояний некоторых переменных, которые используются в контроллере. Фильтр Калмана и фильтр частиц являются двумя примерами популярных байесовских компонентов управления. Байесовский подход к проектированию контроллера часто требует значительных усилий при выводе так называемой модели системы и модели измерения, которые представляют собой математические соотношения, связывающие переменные состояния с измерениями датчиков, имеющимися в управляемой системе. В этом отношении он очень тесно связан с теоретико-системным подходом к проектированию управления. 1.3.9 Языки Исследователями искусственного интеллекта было разработано несколько специализированных языков для исследований ИИ, в том числе Lisp и Prolog. 1.3.10 Оценка прогресса В 1950 году Алан Тьюринг предложил общую процедуру для проверки интеллекта агента, известного теперь как тест Тьюринга. Эта процедура позволяет протестировать почти все основные проблемы искусственного интеллекта. Тем не менее, это очень сложная задача, и в настоящее время все агенты терпят неудачу. Искусственный интеллект может также оцениваться по конкретным задачам, таким как небольшие задачи по химии, распознавание рукописного ввода и игра в игры. Такие тесты были названы экспертными тестами Тьюринга. Меньшие задачи обеспечивают более достижимые цели, и число положительных результатов постоянно растёт. Например, производительность в игре в шашки оптимальна, производительность в шахматах высока – близка к человеческаой и приближается к тому, чтобы превозмочь человеческие способности (суперчеловеческая производительность), а вот производительность в решении многих повседневных задач, таких как распознавание лица или 39 пересечение комнаты, не натыкаясь на препятствия, является «недочеловеческой». Совершенно другой подход измеряет машинный интеллект посредством тестов, которые разрабатываются на основе математических определений интеллекта. Примеры таких тестов начинаются в конце 90-х годов, разрабатывая аналитические тесты с использованием понятий из сложности Колмогорова и сжатия данных. Двумя важными преимуществами математических определений являются их применимость к нечеловеческому разуму и отсутствие потребности в тестерах, определяющих «человечность». Производная от теста Тьюринга – это полностью автоматизированный «публичный» тест Тьюринга, разделяющий людей и компьютеры – Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA). Как следует из названия, это помогает определить, что пользователь является реальным человеком, а не компьютером, представляющим человека. В отличие от стандартного теста Тьюринга, CAPTCHA управляется машиной и нацелена на человека, а управляема человеком и направлена на машину. Компьютер просит пользователя выполнить простой тест, затем генерирует оценку для этого теста. Компьютеры неспособны решить задачу, поэтому правильные решения считаются результатом того, кто проходит тест. Общим типом CAPTCHA является тест, который требует ввода искаженных букв, цифр или символов, которые появляются на изображении, не поддающемся расшифровке компьютером. 40 2 ВЫБОР ИНСТРУМЕНТАРИЯ Прежде чем приступить к разработке необходимо сделать выбор инструментария. Он должен быть максимально задокументированным, обладать большим количеством библиотек и расширений, способствующих решению не только задач, поставленных перед разработчиками на данный момент, но и решению задач, которые будут возникать в ходе разработки других модулей робота. Проанализировав, насколько это возможно в условиях коммерческой тайны, инструментарий, используемый основными конкурентами – Promobot – было выяснено, что разработчики данного продукта предпочли использование платформ, основанных на ядре Linux. Также исследование отзывов о продукте Promobot и участие в 2015 году на выставке, посвящённой инновациям, позволили выяснить, что продукт конкурентов не соответствовал заявленным характеристикам: через программу удалённого доступа к роботу был подключен оператор, которому передавался видео- и аудиопоток, полученный из окружающей робота среды и который в соответствии с полученными данными управлял поведением робота. Такое открытие дало наглядный пример того, что потребителя обмануть не удастся и потеря доверия на этапе презентации может впоследствии только помешать продвижению продукта. Данные платформы обладают обширной документацией, бесплатным доступом, поддерживаются большим количеством устройств и имеют достаточный набор инструментов, ориентированных на робототехнику и разработку искусственного интеллекта. 2.1 Выбор фреймворка для программирования робота Ввиду того что данный модуль разрабатывался в условиях работы в стартапе, основным критерием к выбору используемого инструментария являлась его низкая стоимость. 41 2.1.1 ROS – Robot Operating System ROS, или Robot Operating System, представляет собой набор программных инфраструктур для разработки программного обеспечения роботов, обеспечивающий функциональность, похожую на операционную систему, на гетерогенном компьютерном кластере. ROS предоставляет стандартные сервисы операционной системы, такие как абстракция аппаратных средств, низкоуровневое управление устройствами, реализация часто используемых функций, передача сообщений между процессами и управление пакетами. Запуск наборов процессов на основе ROS представлен в графической архитектуре, где обработка выполняется в узлах, которые могут получать, отправлять и мультиплексировать датчик, управление, состояние, планирование, исполнительный механизм и другие сообщения. Несмотря на важность реакционной способности и малой задержки в управлении роботом, сама ROS не является ОС реального времени (RTOS), хотя есть возможность интегрировать ROS с кодом реального времени. Отсутствие поддержки систем реального времени рассматривается в создании ROS 2.0. Программное обеспечение в экосистеме ROS можно разделить на три группы: 1) Языковые и платформо-независимые инструменты, используемые для создания и распространения программного обеспечения на основе ROS; 2) Реализации клиентской библиотеки ROS, такие как roscpp, rospy, и roslisp; 3) Пакеты, содержащие прикладной код, который использует одну или несколько клиентских библиотек ROS. Как независимые от языка инструменты, так и основные клиентские библиотеки (C ++, Python, LISP) выпускаются в соответствии с лицензией BSD, и поэтому являются программным обеспечением с открытым исходным кодом и свободны для коммерческого и исследовательского использования. Большинство других пакетов лицензируются под различные лицензии с 42 открытым исходным кодом. Эти пакеты реализуют обычно используемые функции и приложения, такие как драйверы оборудования, модели роботов, типы данных, планирование, восприятие, одновременную локализацию и отображение, инструменты моделирования и другие алгоритмы. Основные клиентские библиотеки ROS (C ++, Python, LISP) ориентированы на Unix-подобную систему, главным образом из-за их зависимости от больших коллекций зависимостей программного обеспечения с открытым исходным кодом. Для этих клиентских библиотек Ubuntu Linux указан как «Supported» («поддерживаемая ОС»), а другие варианты, такие как Fedora Linux, macOS и Microsoft Windows, обозначены как «Experimental» («Экспериментальная ОС») и поддерживаются сообществом. Родная клиентская библиотека Java ROS, rosjava, однако, не разделяет этих ограничений и позволила написать программное обеспечение на основе ROS для ОС Android. Rosjava также позволяет интегрировать ROS в официально поддерживаемый набор инструментов MATLAB, который можно использовать в Linux, macOS и Microsoft Windows. Также была разработана клиентская библиотека JavaScript roslibjs, которая позволяет интегрировать программное обеспечение в систему ROS через любой совместимый со стандартами веб-браузер. ROS содержит множество реализаций с открытым исходным кодом общих функциональных возможностей и алгоритмов робототехники. Эти реализации с открытым исходным кодом организованы в «пакеты». Многие пакеты включены в состав дистрибутивов ROS, в то время как другие могут быть разработаны частными лицами и распространены через сайты совместного использования кода, такие как github. Также рассматривались и другие аналоги операционных систем для роботов. Например, Microsoft Robotics Developer Studio. 43 2.1.2 MRDS – Microsoft Robotics Developer Studio Microsoft Robotics Developer Studio (Microsoft RDS, MRDS) – это среда Windows для управления и моделирования роботов. Он предназначен для академических, любительских и коммерческих разработчиков и обрабатывает широкий спектр аппаратных средств роботов. Для этого требуется операционная система Microsoft Windows 7. RDS основана на CCR (Concurrency and Coordination Runtime – параллелизм и координация времени выполнения): реализация параллельной библиотеки на основе .NET для управления асинхронными параллельными задачами. Этот метод включает в себя использование пересылки сообщений и легкую сервис-ориентированную среду исполнения DSS (Decentralized Software Services – децентрализованные программные сервисы), которая позволяет организовать несколько сервисов для достижения сложного поведения. К ним относятся: визуальный программный инструмент, Microsoft Visual Programming Language для создания и отладки робототехнических приложений, веб-интерфейсов и окон, 3D-моделирование (включая аппаратное ускорение), легкий доступ к датчикам и исполнительным устройствам робота. Первичный язык программирования – C #. Microsoft Robotics Developer Studio включает поддержку пакетов для добавления других сервисов в пакет. В настоящее время доступны Microsoft Soccer Simulation и Sumo Competition от Microsoft, а также разработанный сообществом симулятор лабиринта, программа для создания миров со стенами, которые могут быть исследованы виртуальным роботом, и набор сервисов для OpenCV. Большинство дополнительных пакетов размещено на CodePlex (поиск для Robotics Studio). Также доступны учебные материалы. К недостаткам Microsoft Robotics Developer Studio можно отнести следующие аспекты: 44 1) У данной операционной системы отсутствует учёт, а также поддержка условий, в которых на данный момент находится и решает задачи робот, таких как поверхность, по которой движется агент, погодные условия и прочее; 2) При эксплуатации интеллектуального агента в расчёт принимается только его имитация, параметры которой могут не до конца соответствовать реальным параметрам робота; 3) Чем более точная имитация используется операционной системой, тем больше для неё требуется настроек; 4) «Физика» в данной операционной системе является упрощённой (несмотря на интегрирование PhysX). Портирование ROS на Microsoft Windows возможно, но ещё не полностью реализовано. 2.1.3 The Player Project The Player Project (ранее просто Player / Stage Project) представляет собой проект для создания свободного программного обеспечения для исследования робототехники и сенсорных систем. Его компоненты включают сетевой сервер Player и симуляторы платформы роботов Stage. Хотя точную статистику и трудно получить, Player является одним из самых популярных интерфейсов роботов с открытым исходным кодом в исследованиях и высшем образовании. Большинство крупных интеллектуальных журналов и конференций по робототехнике регулярно публикуют статьи с реальными и имитируемыми экспериментами роботов с использованием Player и Stage. The Player Project является проектом, в рамках которого в настоящее время разрабатываются два проекта в области программного обеспечения, связанных с робототехникой. К ним относятся сетевой сервер Player с роботами и среда моделирования роботов Stage 2D. Проект был основан в 2000 году Брайаном Герки, Ричардом Воганом и Эндрю Говардом в Университете Южной Калифорнии в Лос-Анджелесе и широко используется в 45 исследованиях и обучении робототехники. Он выпускает свое программное обеспечение в соответствии с GNU General Public License с документацией в соответствии с GNU Free Documentation License. The Player представляет собой набор API-интерфейсов (например, position2d, bumper, ir, speech, power), которые могут быть реализованы с помощью шасси робота (Roomba, Khephera и т. Д.), возможно, через последовательную линию или сеть, Stage (2D-симулятор) или Gazebo (3Dсимулятор). Программное обеспечение Player работает на совместимых с POSIX операционных системах, включая Linux, Mac OS X, Solaris, варианты BSD и Microsoft Windows. The Player может быть описан как «слой абстракции робота», в котором все устройства абстрагируются в набор предопределенных интерфейсов. The Player поддерживает широкий спектр аппаратного обеспечения (сенсорные устройства и роботизированные платформы). Он также содержит поддержку клиентской библиотеки для ряда языков программирования, включая C, C ++, Python и Ruby. Клиентские библиотеки сторонних разработчиков доступны на таких языках, как Java и Tcl. Дополнительные функции включают в себя минимальный и гибкий дизайн, поддержку взаимодействия с несколькими устройствами одновременно, а также конфигурацию сервера «на лету». Симулятор Stage – это двухмерная среда моделирования с несколькими роботами, построенная поверх FLTK. Stage предоставляет базовую среду моделирования, которую можно масштабировать для моделирования от одного до сотен роботов одновременно. Stage может использоваться самостоятельно, чтобы имитировать поведение робота с помощью пользовательских управляющих программ. Stage также может взаимодействовать с Player, позволяя игрокам получать доступ к имитированным датчикам и устройствам через интерфейсы Player. 46 Симулятор робота Gazebo 3D был компонентом в Player Project с 2004 по 2011 год. Gazebo интегрировала физический движок ODE, рендеринг OpenGL и код поддержки для моделирования датчиков и управления приводом. В 2011 году Gazebo стала независимой проектной поддержкой Willow Garage. Разница между ROS и Player включает в себя многие аспекты. В частности, это зависит от цели, которую преследуют разработчики. Player незаменим для простых мобильных платформ. В основу его разработки легло обеспечение простого доступа к датчикам и двигателям роботов Pioneer, оборудованных лазерным дальномером. В то же время ROS разрабатывался для полного спектра мобильных платформ: с различными датчиками и манипудяторами. И если сравнивать ROS с Player, то однозначно можно сказать, что ROS легче использовать в распределенной вычислительной среде. Также можно добавить, что высокоуровневая сторона более развита в ROS, чем в Player. Также следует учесть, что Player предлагает больше драйверов оборудования, а ROS – больше реализаций алгоритмов. Вместе с этим было бы справедливо сказать, что ROS является более мощной и гибкой системой, чем Player, но, как обычно, большая мощность и гибкость были достигнуты засчёт и большей сложности самой системы. Концепции Player и ROS очень схожи. ROS использует много кода из проекта Player. Есть узлы ROS, в которых используется код из драйверов Player, и оба, Stage и Gazebo, хорошо поддерживаются и широко используются в ROS-сообществе. В качестве операционной системы для написания алгоритмов для робота была выбрана ROS. Далее были рассмотрены и изучены модули, которые впоследствии были использованы для разработки алгоритмов, которые служили поставленной цели. 47 2.2 Выбор модулей, необходимых для построения заданных алгоритмов для робота Для проектирования алгоритмов необходимо использовать библиотеки, распространяемые с ROS. Для этого были изучены модули, предлагаемые на официальном ресурсе фреймворка. 2.2.1 Модуль amcl Amcl – вероятностная система локализации для робота, движущегося в 2D. Он реализует адаптивный (или KLD-дискретизированный) метод локализации Монте-Карло (как описано Дитером Фокс), который использует фильтр частиц для отслеживания позы робота на известной роботу карте. Данный модуль получен от драйвера amcl Эндрю Говарда, разработанного для Player. Многие из алгоритмов и их параметры описаны в книге «Вероятностная робототехника». В частности, используются следующие алгоритмы из этой книги: sample\_motion\_model\_odometry, beam\_range\_finder\_model, likelihood\_field\_range\_finder\_model, Augmented\_MCL и KLD\_Sampling\_MCL. В настоящее время данный модуль работает только с лазерным сканированием и лазерными картами. Его возможности могут быть расширены для работы с другими данными датчиков. amcl берёт карту на основе карты, построенной лазером, и сканов лазера и преобразует сообщения, а выходные данные создают оценку полученной информации. При запуске amcl инициализирует свой фильтр частиц в соответствии с предоставленными параметрами. Из-за значений по умолчанию, если не заданы никакие параметры, начальное состояние фильтра будет облаком частиц умеренного размера с центром в точке пространства (0,0,0). 48 Amcl преобразует поступающие лазерные сканы в рамку одометрии. Таким образом, должен существовать путь через дерево трансформаций (tf) из кадра, в котором лазерное сканирование публикуется в рамке одометрии. Детали реализации: после получения первого лазерного сканирования amcl просматривает преобразование между рамой лазера и базовой рамой (~ base\_frame\_id) и фиксирует её навсегда. Поэтому amcl не может справиться с лазером, который движется относительно базы (робота). На рисунке 2.2.1 показана разница между локализацией с использованием одометрии и amcl. Во время работы amcl оценивает преобразование базового фрейма (~base\_frame\_id) по отношению к глобальному фрейму (~ global\_frame\_id), но публикует только преобразование между глобальным фреймом и фреймом одометрии (~ odom\_frame\_id). По существу, это преобразование учитывает дрейф, который происходит с использованием «Dead Reckoning». Рисунок 2.2.1 – Локализация с использованием одометрии и amcl 49 2.2.2 Модуль teb\_local\_planner Пакет teb\_local\_planner реализует плагин к base\_local\_planner из 2Dнавигационного стека. Основной метод под названием Timed Elastic Band локально оптимизирует траекторию робота относительно времени выполнения траектории, отделения от препятствий и соблюдения кинодинамических ограничений во время выполнения. Этот модуль реализует оптимальный локальный планировщик траектории онлайн для навигации и управления мобильными роботами в качестве плагина для навигационного пакета ROS. Начальная траектория, сгенерированная глобальным планировщиком, оптимизируется во время выполнения, минимизируя время выполнения траектории (реализация задачи оптимальности по времени), преодолевая препятствия и соблюдая кинодинамические ограничений, такие как удовлетворение максимальных скоростей и ускорений. Текущая реализация соответствует кинематике неголономных роботов (дифференциальных приводов и автомобильных роботов). Поддержка голономных роботов включена с начала использования Kinetic. Оптимальная траектория эффективно получается при решении разреженной скарифицированной задачи многокритериальной оптимизации. Пользователь может указать весовые коэффициенты для оптимизации, чтобы определить поведение в случае возникновения противоречивых целей. Так как локальные планировщики, такие как Timed-Elastic-Band, часто застревают на локально оптимальной траектории, поскольку они не могут проходить через препятствия, реализуется расширение. Параллельно оптимизируется подмножество допустимых траекторий особых топологий. Локальный планировщик может переключиться на текущую глобально оптимальную траекторию среди набора представленных. Отличительные топологии получаются при использовании понятия гомологий и 50 гомотопических классов. Дальнейшая информация по этому расширению будет предоставлена в ближайшее время. Модуль teb\_local\_planner позволяет пользователю устанавливать параметры, чтобы настроить поведение робота. Эти параметры сгруппированы в несколько категорий: конфигурация робота, допуск цели, конфигурация траектории, препятствия, оптимизация, планирование в отличительных топологиях и разные параметры. Некоторые из них выбираются так, чтобы соответствовать base\_local\_planner. Для модуля планируются некоторые функции и улучшения, которые будут представлены в ближайшем будущем. Как то: - Добавить и улучшить функции безопасности в случае неизбежных препятствий (например, для препятствий, которые расположены очень близко к цели); - Реализация подходящего поведения для «эвакуации»; - Улучшения и решения для случаев, когда планировщик колеблется между несколькими локально оптимальными решениями (не на топологической основе, а из-за возникающего шума и т. д.). 2.2.3 Модуль move\_base Пакет move\_base обеспечивает реализацию действия, которое, учитывая цель в окружающей среде, попытается достичь её с помощью мобильной базы. Узел move\_base связывает глобальный и местный планировщик для выполнения своей задачи глобальной навигации. Он поддерживает любой глобальный планировщик, поддерживающий интерфейс nav\_core :: BaseGlobalPlanner, указанный в ьщдуле nav\_core, и любой локальный планировщик, придерживающийся интерфейсом nav\_core :: BaseLocalPlanner, указанного в пакете nav\_core. Узел move\_base также поддерживает две карты затрат: одну для глобального планировщика и одну 51 для локального планировщика, которые используются для выполнения задач навигации. На рисунке 2.2.3 представлена схема взаимодействия компонентов модуля. Рисунок 2.2.3.1 – Схема взаимодействия компонентов модуля move\_base Узел move\_base предоставляет интерфейс ROS для настройки, запуска и взаимодействия с навигационным стеком на роботе. Выше показан вид верхнего уровня узла move\_base и его взаимодействие с другими компонентами. Синий цвет обозначает платформу робота, серый цвет обозначает опциональные компоненты, которые предоставляются для всех систем, а белые узлы обозначают необходимые компоненты, которые также предоставляются для всех систем. Запуск узла move\_base на роботе, который правильно настроен (более подробно см. Документация в навигационном стеке), приводит к тому, что робот попытается достичь поставленной цели с базой в пределах заданного пользователем допусков. В отсутствие динамических препятствий узел move\_base в конечном итоге попадает в этот допуск для своей цели или сбоя сигнала пользователю. Узел move\_base может дополнительно выполнять поведение восстановления, когда робот воспринимает себя как зависший. На 52 рисунке 2.2.3.2 представлены действия по умолчанию, предпринимаемые узлом move\_base, чтобы попытаться очистить пространство. Рисунок 2.2.3.2 – Действия по умолчанию, предпринимаемые узлом move\_base и направленные на очищение пространства Во-первых, препятствия вне указанного пользователем региона будут удалены с карты робота. Затем, по возможности, робот выполнит вращение на месте, чтобы освободить место. Если это также не удастся, робот будет более агрессивно очищать свою карту, удаляя все препятствия за пределами прямоугольной области, в которой он может вращаться на месте. За этим последует еще одна ротация на месте. Если все эти действия окажутся тщетным, робот будет считать свою цель невыполнимой и уведомит пользователя о том, что он прервал операцию. Эти способы восстановления можно настроить с помощью параметра recovery\_behaviors и отключить с помощью параметра recovery\_behavior\_enabled. 2.2.4 Модуль gmapping Этот модуль содержит оболочку ROS для Gmapping OpenSlam. Модуль gmapping предоставляет SLAM на основе лазера (одновременную локализацию и сопоставление) в качестве узла ROS, называемого slam\_gmapping. Используя slam\_gmapping, вы можете создать 2D-карту сетки занятости (например, план здания) из данных лазера и позы, собранных мобильным роботом. 53 Недавно в качестве эффективных средств решения проблемы одновременной локализации и картографирования (SLAM) были введены фильтры с частицами Рао-Блэквеллиза. Этот подход использует фильтр частиц, в котором каждая частица несет индивидуальную карту среды. Соответственно, ключевым вопросом является то, как уменьшить количество частиц. В модуле приведены адаптивные методы для уменьшения числа частиц в фильтре Рао-Блэквеллизированной частицы для изучения сетчатых отображений. Предлагается подход к расчёту точного распределения предложения с учетом не только движения робота, но и самого последнего наблюдения. Это резко уменьшает неопределённость о позе робота на этапе прогнозирования в фильтре. Кроме того, применяется подход, который выборочно осуществляет повторную выборку операций, которые серьёзно уменьшают проблемы разрушения частиц. Этот подход использует необработанные данные лазерного диапазона и одометрию. Эта версия оптимизирована для лазерных сканеров большой дальности, таких как SICK LMS или PLS-сканер. Лазеры с коротким радиусом, такие как сканер Hokuyo, не будут хорошо работать со стандартными параметрами. Чтобы использовать slam\_gmapping, необходим мобильный робот, который предоставляет данные о одометрии и оснащен горизонтально установленным лазерным дальномером. Узел slam\_gmapping попытается преобразовать каждое входящее сканирование в фрейм одометра (odometry). Подход SLAM доступен в виде библиотеки и может быть легко использован как чёрный ящик. Однако внесение изменений в алгоритм требует некоторого опыта работы с C ++. 2.2.4 Модуль openni\_tracker Трекер OpenNI передает кадры OpenNI скелета, используя трансформации (библиотека tf). 54 Rosrun openni\_tracker openni\_tracker – независимый узел, который не требует запуска других узлов. Однако требуется убедиться, что Kinect включён. Например, на Turtlebot необходимо запустить kinect.launch. После того, как модуль будет запущен, необходимо встать перед Kinect и поднять руки вверх (т.е. встать в позу сдающегося). После будут заметны некоторые изменения в следующих сообщениях. 2.2.4 Модуль costmap\_2d Этот пакет предоставляет реализацию 2D-карты затрат, которая принимает с датчиков данные об окружающей среде, создает 2D-или 3D-сетку занятости данных – какой пиксель на текущей карте занят, и строят радиус инфляции у каждого объекта – радиус вокруг самого объекта, который считается занятым. Так же они берут статичную карту и проделывают тот же алгоритм. А потом соединяют получившиеся «куски» в одну, глобальную карту занятости. Пакет costmap\_2d предоставляет конфигурируемую структуру, которая поддерживает информацию о том, где робот должен перемещаться в форме сетки занятости. Costmap использует сенсорные данные и информацию со статической карты для хранения и обновления информации о препятствиях в окружающей среде через объект costmap\_2d::Costmap2DROS. Объект costmap\_2d:: Costmap2DROS обеспечивает только двумерный интерфейс для своих пользователей, что означает, что запросы о препятствиях могут быть сделаны только в столбцах. Например, стол и ботинок в том же положении в плоскости XY, но с разными позициями Z, что приведёт к тому, что соответствующие ячейки costmap в costmap\_2d::Costmap2DROS будут иметь идентичную стоимость. Это сделано для того, чтобы помочь планировать маршрут в «плоских» пространствах. К моменту выпуска Hydro основные методы, используемые для записи данных в карту затрат, стали полностью настраиваемыми. В слое существует 55 каждый бит функциональности. Например, статическая карта – это один слой, а препятствия – ещё один слой. По умолчанию слой препятствий хранит информацию в трёх измерениях. Поддержка данных о 3D-препятствиях позволяет слою справляться с разметкой и очисткой более рационально. Схема Costmap автоматически подписывается на топики датчиков через ROS и обновляется соответствующим образом. Каждый датчик используется либо для маркировки (вставки информации о препятствиях в карту затрат), либо для очистки (удаления информации о препятствиях из карты затрат), либо в обоих случаях. Операция маркировки – это всего лишь индекс в массиве, изменяющий стоимость ячейки. Однако операция очистки состоит из трассировки лучей по сетке от начала датчика до конца для каждого зарегистрированного наблюдения. Если для хранения информации о препятствиях используется трёхмерная структура, информация о препятствиях из каждого столбца при вводе в карту проекта проецируется на два измерения. Хотя каждая ячейка в таблице затрат может иметь один из 255 различных значений стоимости, базовая структура, которую она использует, может представлять только три. В частности, каждая ячейка в этой структуре может быть свободной, занятой или неизвестной. Каждому статусу присваивается специальное значение стоимости при проецировании в карту затрат. Столбцам, которые имеют определенное количество занятых ячеек, присваивается стоимость costmap\_2d :: LETHAL\_OBSTACLE, столбцам с определенным числом неизвестных ячеек назначается стоимость costmap\_2d :: NO\_INFORMATION, а другим столбцам назначается стоимость costmap\_2d :: FREE\_SPACE. Карта затрат выполняет циклы обновления карты со скоростью, указанной параметром update\_frequency. Каждый цикл, когда поступают данные датчиков, операции по маркировке и очистке выполняются в основной структуре занятости карты затрат, и эта структура проецируется в таблицу 56 затрат, где соответствующие значения стоимости присваиваются в соответствии с правилами, описанными ране. После этого каждая инфляция препятствия выполняется для каждой ячейки со стоимостью costmap\_2d :: LETHAL\_OBSTACLE. Данная операция состоит в распространении значений стоимости за пределы от каждой занятой ячейки до указанного пользователем радиуса инфляции. Ниже приводится подробная информация об этом процессе инфляции. Для того чтобы вставить данные из источников датчиков в карту затрат, объектом costmap\_2d :: Costmap2DROS широко пользуются. В частности, предполагается, что все преобразования между кадрами координат, заданными параметром global\_frame, параметром robot\_base\_frame и источниками датчиков, подключены и обновлены. Параметр transform\_tolerance устанавливает максимальную задержку между этими преобразованиями. Если дерево tf (дерево трансформаций) не обновляется с такой ожидаемой скоростью, стек навигации останавливает робота. Инфляция – процесс распространения значений стоимости из занятых ячеек, которые уменьшаются с расстоянием. Для этой цели определяются 5 специальных символов для значений стоимостной карты, поскольку они относятся к роботу. 1) «Lethal» стоимость означает, что в ячейке существует фактическое препятствие (рабочее пространство). Поэтому если центр робота находился в этой камере, робот, очевидно, был бы в столкновении. 2) «Inscribed» стоимость означает, что ячейка меньше, чем вписанный радиус робота от фактического препятствия. Таким образом, робот, безусловно, сталкивается с каким-то препятствием, если центр робота находится в ячейке, которая находится выше или равна вписанной стоимости. 3) «Possibly circumscribed» стоимость аналогична «вписанной», но с использованием описанного радиуса робота как пограничного расстояния. Таким образом, если центр робота находится в ячейке выше или выше этого 57 значения, то всё зависит от ориентации робота, независимо от того, сталкивается он с препятствием или нет. В термине используется слово «possibly»(«возможно»), потому что данная ячейка может оказаться не препятствием, а некоторыми пользовательскими предпочтениями, из-за чего данная стоимость была внесена на карту затрат. Например, если пользователь хочет, чтобы робот пытался избежать конкретной области здания, он может вписать свои собственные веса (стоимости) в карту затрат для этого региона независимо от существования каких-либо препятствий. 4) Стоимость «Freespace» считается равной нулю и это означает, что нет ничего, что должно препятствовать прохождению робота через эту ячейку. 5) «Unknown» стоимость означает отсутствие информации о данной ячейке. Пользователь карты может интерпретировать это так, как считает нужным. 6) Всем остальным расходам присваивается значение между «Freespace» и «Possibly circumscribed» в зависимости от их расстояния от «Lethal» ячейки и функции распада, предоставляемой пользователем. Существует два основных способа инициализации объекта costmap\_2d :: Costmap2DROS. Первый заключается в том, чтобы заполнить ею сгенерированную пользователем статическую карту. В этом случае карта затрат инициализируется в соответствии с информацией о ширине, высоте и препятствиях, предоставляемой статической картой. Эта конфигурация обычно используется в сочетании с системой локализации, такой как amcl, которая позволяет роботу регистрировать препятствия в каркасе карты и обновлять свою карту затрат из данных датчиков при прохождении через её среду. Второй способ инициализации объекта costmap\_2d :: Costmap2DROS – присвоить ему ширину и высоту и установить для параметра rolling\_window значение true. Параметр rolling\_window удерживает робота в центре карты, когда он перемещается по всей локации, удаляя информацию о препятствиях с карты, когда робот перемещается слишком далеко от данной области. Этот 58 тип конфигурации чаще всего используется в одометрической системе координат, где робот заботится только о препятствиях в предел