федеральное государственное автономное образовательное учреждение
высшего образования «Московский физико-технический институт
(национальный исследовательский университет)»

На правах рукописи

Жамал Маис

Разработка и исследование методов и алгоритмов адаптивного планирования маневров беспилотного автомобиля

Специальность 1.2.1. Искусственный интеллект и машинное обучение

ДИССЕРТАЦИЯ
на соискание учёной степени
кандидата технических наук

Научный руководитель:
кандидат физико-математических наук
Панов Александр Игоревич

Долгопрудный — 2024

Оглавление

Стр.

Стр.

Стр.

Введение

Системы автономного вождения занимают лидирующие позиции в технологическом прогрессе, потенциально закладывая основы для революционно нового транспорта в будущем, позволяя автомобилям управлять и функционировать самостоятельно. В основе этих систем лежит несколько ключевых компонентов, каждый из которых необходим для обеспечения безопасной и эффективной автономной работы.

Ключевыми компонентами системы автономного вождения [5] являются локализация (localization), восприятие (perception), планирование (planning) и управление (control). Их совместная работа позволяет транспортным средствам автономно ориентироваться и действовать в сложных условиях. Благодаря использованию и внедрению передовых датчиков, алгоритмов и картографических технологий автономные системы вождения обещают стать более безопасным, эффективным и доступным транспортом для всего общества[6].

Быстрое развитие автономных транспортных средств (АТС) стимулировало значительные исследовательские усилия в области разработки методов автоматического планирования. Эти методы играют ключевую роль в обеспечении возможности АТС передвигаться в сложных и динамичных средах, гарантируя при этом безопасность, эффективность и устойчивость. Автоматическое планирование в АТС представляет собой множество вызовов, связанных с необходимостью учитывать широкий спектр неопределенностей, начиная с непредсказуемых условий окружающей среды и заканчивая поведением других участников дорожного движения.

Автоматическое планирование включает в себя процесс принятия решений и создания безопасной и эффективной траектории, по которой будет двигаться автономное транспортное средство [7; 8]. Небольшая ошибка в принятии решений, планировании траектории или предсказании траектории может привести к потенциально опасным ситуациям, включая столкновения и небезопасные маневры. Имея доступ к HD-карте и используя данные от модулей восприятия и локализации, а также модели предсказания, алгоритмы планирования анализируют окружающую среду, предвидят будущие сценарии, основываясь на предсказанных будущих движениях других

транспортных средств, пешеходов и объектов в окружении, и определяют оптимальный курс действий [6]. Действие включает в себя принятие решений по различным маневрам, таким как уступка (yielding) или следование за ведущим автомобилем, удержание полосы (lane-keeping), смена полосы (lane-change), слияние (merging), обгон (overtaking) и проезд перекрестков.

Задачи принятия решений для автономного вождения обычно решаются с помощью классических и основанных на обучении методов [9]. К классическим методам принятия решений относятся методы, основанные на правилах, методы оптимизации и вероятностные подходы. Методы, основанные на правилах [10; 11], являются интерпретируемыми и простыми в реализации, но испытывают трудности в сложных и динамичных условиях. Методы оптимизации [12; 13] хорошо моделируют взаимодействия, однако часто терпят неудачу в реальных условиях из-за предположения об «оптимальной стратегии». Вероятностные методы [14; 15] хорошо сочетаются с другими подходами, но имеют низкую эффективность в сложных средах.

Методы, основанные на обучении, такие как статистическое обучение, глубокое обучение и обучение с подкреплением (RL, Reinforcement Learning), обладают различными преимуществами и недостатками. Статистические методы универсальны, но требуют большого объема данных и имеют низкую точность. Глубокое обучение обеспечивает высокую точность и полностью использует данные окружающей среды, однако требует больших наборов данных и вычислительных мощностей, а также имеет проблемы с интерпретацией [16]. Обучение с подкреплением хорошо справляется с неопределенностью и динамикой, но сталкивается с рядом значительных вызовов [17; 18]. К ним относятся проверка работоспособности систем на основе RL, преодоление разрыва между симуляцией и реальностью, достижение эффективности выборки, разработка эффективных функций вознаграждения и интеграция безопасности в процессы принятия решений для автономных агентов, что также может привести к проблемам стабильности и переобучения (overfitting).

В данной диссертации решаются критические задачи в области автоматического планирования маневров для автономных транспортных средств, разрабатывая новые методы, направленные на улучшение принятия решений в динамичных и неопределенных условиях. В исследовании предложены четыре метода, задействующие два основных подхода:

два метода, основанные на обучении, с использованием обучения с подкреплением и генетического программирования, и два метода, основанные на детерминированном планировании с использованием выборок (сэмплеров). Эти методики разработаны для повышения надежности и эффективности автономного принятия решений, улучшая возможности технологий автономных транспортных средств.

Работа, представленная в настоящей диссертации, вносит свой вклад в развитие современных технологий планирования для автономных транспортных средств, предлагая решения, повышающие безопасность, надежность и адаптивность в реальных условиях вождения. Разработка этих методов необходима и важна для дальнейшего прогресса технологии автономных транспортных средств.

Целью данной работы является разработка и исследование методов и алгоритмов адаптивного планирования маневров беспилотного автомобиля.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. Исследовать и разработать адаптивный метод принятия решений для планирования маневра обгона в динамичных условиях.
2. Исследовать и разработать адаптивные методы, основанные на обучении с подкреплением, для маневров парковки в системах автономного вождения.
3. Разработать эффективные адаптивные методы планирования, объединяющие эвристическое принятие решений и планирование движения для ряда маневров, включая обгон, смену полосы движения и проезд перекрестков, в автономных системах вождения.

Научная новизна: В данном исследовании рассматривается применение методов, основанных на обучении, с использованием обучения с подкреплением и эволюционных алгоритмов, а также детерминированного планирования с использованием выборок (сэмплеров) для адаптивного планирования маневров для автономных транспортных средств.

1. Реализован обучаемый метод принятия решений для планирования маневра обгона в динамической среде на основе эволюционного алгоритма.
2. Реализован метод обучения с подкреплением для адаптивного планирования парковки в стохастической динамической среде.

3. Разработан метод FFStreams адаптивных маневров обгона и смены полосы движения в динамических условиях, основанный на детерминированном планировании с использованием выборок траекторий.
4. Разработан оригинальный метод FFStreams++ адаптивного планирования маневров с предсказанием поведения объектов в динамичных условиях вождения.

Практическая значимость

Разработанные методы могут быть использованы для решения многих практических задач. Они могут применяться в системах помощи водителю (ADAS, Advanced Driver Assistance Systems), улучшая такие функции, как удержание полосы и предотвращение столкновений. Данное исследование может повысить эффективность автономных систем логистики и доставки, оптимизируя планирование, снижая операционные расходы и улучшая надежность автономных грузовиков и роботов-курьеров. Кроме того, в сервисах совместного использования транспорта, таких как автономные такси, данные методы могут способствовать эффективному управлению парком транспортных средств, позволяя автомобилям реагировать на изменяющийся спрос пассажиров. Разработанные решения по автоматизации парковки имеют потенциал для использования в инициативах умных городов, способствуя снижению заторов и оптимизации использования пространства в парковочных структурах.

Методология и методы исследования.

Разработанные в данном исследовании алгоритмы используют как детерминированные, так и вероятностные модели планирования для исследования принятия решений в условиях неопределенности, применяя сочетание экспериментов на основе симуляций и алгоритмического анализа. Симуляция имеет ключевое значение для имитации реальных сценариев, позволяя проводить контролируемое тестирование поведения автономных транспортных средств в структурированной и неструктурированной среде. Для оценки эффективности моделей планирования применяются количественные показатели эффективности, включая вычислительную эффективность, безопасность и оптимизацию пути. Эта методология также комбинирует детерминированное планирование с выборочным и вероятностным подходами

к планированию, предлагая понимание того, как каждый метод справляется с различными неопределенностями, присущими автономному вождению.

С точки зрения методов, в исследовании используются фреймворки на основе деревьев поведения (BT, Behavior Trees), генетическое программирование и обучение с подкреплением для улучшения процессов принятия решений в автономных системах. Эти техники реализованы и протестированы на платформе Apollo [19], где симуляции разработаны для имитации сложных сценариев вождения, таких как навигация по шоссе и парковка. Оценка адаптивного планирования маневров использует деревья поведения как с предварительными знаниями, так и без них, чтобы оценить, как система адаптируется к непредвиденным событиям. Дополнительно применяются эвристические методы поиска, включая подходы на основе потоков и модели предсказания траектории, такие как QCNet [20], для дальнейшей оптимизации процессов принятия решений. Экспериментальные оценки проводятся с использованием установленных бенчмарков, таких как CommonRoad [21], для обеспечения надежности и обобщаемости.

Основные положения, выносимые на защиту:

1. Адаптивное планирование маневра обгона с использованием подхода на основе дерева поведения, включающего генетическое программирование для динамической адаптации к новым условиям вождения.
2. Метод обучения с подкреплением для адаптивного планирования маневров в задаче парковке, где использование обучение по расписанию (curriculum learning) позволило системе постепенно обучаться сложным задачам вождения.
3. Метод FFStreams, новый подход к быстрому эвристическому поиску с использованием выборок потоков (streams) для эффективной генерации траекторий. Благодаря интеграции потока конфигурации (configuration stream) и потока траекторий (trajectory stream), система планирования успешно сократила пространство поиска, обеспечив при этом создание выполнимых траекторий.
4. Оригинальный метод FFStreams++, который сочетает быстрый эвристический поиск с предсказанием траекторий через модель QCNet для адаптивного планирования маневров. Интеграция предсказания траекторий улучшила процесс принятия решений, позволяя точно

и в реальном времени прогнозировать движение окружающих препятствий.

Достоверность полученных результатов обеспечивается методикой численного эксперимента. Результаты находятся в соответствии с результатами, полученными другими авторами. Для каждого из алгоритмов предлагается его детальное описание, а также полный список гиперпараметров.

Апробация работы. Основные результаты работы докладывались на:

- (63-й Конференции МФТИ) , 63-й Всероссийской научной конференции МФТИ, 2020, Долгопрудный.
- (БТС-ИИ-2021) 6-м Всероссийском научно-практическом семинаре «Беспилотные транспортные средства с элементами искусственного интеллекта», 2020, Москва.
- (AI-2021) Artificial Intelligence XXXVIII: 41st SGAI International Conference on Artificial Intelligence, 2021, Кембридж.
- семинаре «Высокоуровное планирование и оптимизация траектории в Apollo», Центр Когнитивного Моделирования МФТИ, 2020, Долгопрудный.
- семинаре «Адапативное планирования в Apollo, RL в адаптации трактории Apollo», Центр Когнитивного Моделирования МФТИ, 2021, Долгопрудный.
- семинаре «PDDLStream и планировщик Fast-Forward для планирования манёвров беспилотного автомобиля», Центр Когнитивного Моделирования МФТИ, 2022, Долгопрудный.
- семинаре «Планирование манёвров беспилотного автомобиля с использованием PDDLStream», Центр Когнитивного Моделирования МФТИ, 2022, Долгопрудный.

Содержание диссертации соответствует паспорту специальности 1.2.1. Искусственный интеллект и машинное обучение, в частности, пунктам:

- 5. Методы и технологии поиска, приобретения и использования знаний и закономерностей, в том числе – эмпирических, в системах искусственного интеллекта. Исследования в области совместного применения методов машинного обучения и классического математического моделирования. Методы и средства использования экспертных знаний.

- 6. Формализация и постановка задач управления и (поддержки) принятия решений на основе систем искусственного интеллекта и машинного обучения. Разработка систем управления с использованием систем искусственного интеллекта и методов машинного обучения в том числе – управления роботами, автомобилями, БПЛА и т.п.
- 12. Исследования в области «доверенных» систем класса ИИ, включая проблемы формирования тестовых выборок прецедентов, надежности, устойчивости, переобучения и т.д.

Публикации. Основные результаты по теме диссертации изложены в трех печатных изданиях [1—3], в том числе два из которых опубликованы в изданиях собственного перечня журналов МФТИ категории К1.

В статье [1] автор лично разработал и внедрил адаптивную систему планирования маневров для автономных транспортных средств с использованием деревьев поведения на платформе Apollo. Это исследование было посвящено оптимизации принятия решений в динамических условиях вождения. В [2] автор внес вклад в разработку подходов, основанных на обучении, для адаптивных маневров при парковке в беспилотных автомобилях, участвуя в разработке методик, улучшающих способность агента автономно выполнять сложные маневры в ограниченных условиях. В работе [3] автор представил и реализовал FFStreams, алгоритм быстрого поиска с использованием потоков для эффективного планирования автономных маневров. Эта работа значительно повышает скорость и точность принятия решений в автономных системах, представляя основу для интеграции предсказаний траектории с эвристическими методами поиска. В работе [4] автор представил и реализовал FFStreams++, алгоритм быстрого эвристического поиска с редсказанием траекторий через модель QCNet для адаптивного планирования маневров.

Объем и структура работы. Диссертация состоит из введения, шести глав, заключения и одного приложения. Полный объём диссертации составляет 139 страниц, включая 34 рисунка и 9 таблиц. Список литературы содержит 73 наименования.

Глава 1. Автоматическое планирование

В настоящей главе рассматривается критически важная область автоматического планирования в контексте автономных транспортных средств (АТС), которая играет ключевую роль в обеспечении безопасной и эффективной навигации в сложных условиях. Автоматическое планирование включает процесс принятия решений и генерации траекторий, позволяя АТС выполнять различные маневры, адаптируясь к динамичным и неопределенным условиям.

Глава организована следующим образом. Раздел 1.1 вводит основные концепции и технологические достижения, лежащие в основе беспилотных автомобилей, подчеркивая их технологическую эволюцию и основные системы, обеспечивающие их работу. В разделе 1.2 рассматриваются конкретные требования к планированию маневров беспилотного автомобиля, акцентируя внимание на сложностях и вызовах принятия решений в реальном времени в динамичных условиях, а также подчеркивая необходимость сложных алгоритмов планирования для эффективного решения этих задач. Раздел 1.3 описывает математические основы и формулировки задач для методов детерминированного планирования, где предполагается, что среда полностью известна и предсказуема.

Далее, в разделе 1.4 рассматриваются конкретные формулировки задач, связанные с детерминированным планированием, которое использует методы сэмплирования. В этом разделе приводятся математические обозначения и описательный язык, помогающие понять, как сэмплеры могут повысить точность планирования. В разделе 1.5 исследуются формулировки, используемые в вероятностном планировании, которое включает модели для обработки неопределенностей и динамических изменений в окружающей среде. Наконец, в разделе 1.6 формулировка задачи расширяется, чтобы рассмотреть планирование в условиях значительной неопределенности и обучение стратегии с использованием обучения с подкреплением.

Цель этой главы — заложить основу для дальнейшего изучения и развития современных методов планирования маневров беспилотного автомобиля за счет понимания роли автоматического планирования в автономных транспортных средствах, его сложности и различных стратегий планирования, включая их формулировки задач и математические основы.

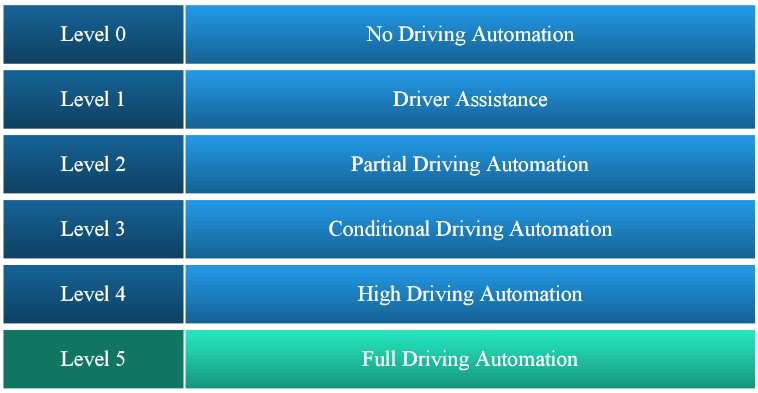


Рисунок 1.1 — Уровни автоматизации беспилотных автомобилей согласно SAE [22].

1.1 Беспилотные автомобили

Автономные транспортные средства, также известные как самоуправляемые автомобили, представляют собой революционное достижение в области робототехники. Они могут работать самостоятельно, без вмешательства человека, и используют сложные датчики, включая радар, GPS, камеры и лидар, для постоянного мониторинга и интерпретации окружающей обстановки, что позволяет им выполнять необходимые маневры. Общество автомобильных инженеров (SAE, Society of Automotive Engineers) классифицирует возможности автоматизации этих автомобилей по шести различным уровням [22], начиная с уровня 0 и заканчивая уровнем 5 (рисунок 1.1).

На уровне 0 отсутствует автоматизация, и человек-водитель несет ответственность за выполнение всех задач, при этом система помощи водителю (ADAS) автомобиля предоставляет только пассивную поддержку, например, предупреждения. Уровень 1 вводит минимальную автоматизацию, где ADAS может помочь с рулевым управлением или ускорением/торможением, но водитель остается полностью ответственным за управление. На уровне 2 ADAS может одновременно контролировать и рулевое управление, и ускорение/торможение в определенных условиях, хотя водитель должен оставаться бдительным и готовым взять управление на себя в любой момент. Уровень 3 позволяет ADAS выполнять все задачи вождения при определенных обстоятельствах, однако водитель должен оставаться готовым взять на себя управление при необходимости. Уровень 4 продвигается дальше, позволяя

ADAS управлять всеми аспектами вождения без вмешательства человека, но только лишь в определенных средах или условиях. Наконец, уровень 5 представляет полную автоматизацию, при которой автомобиль может работать полностью автономно в любых условиях без необходимости участия человека.

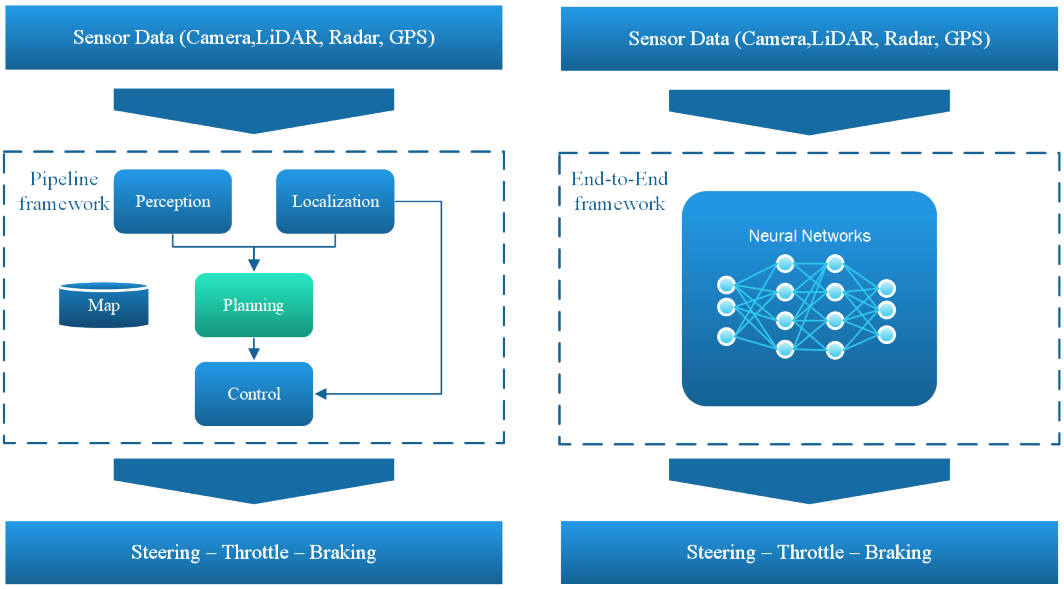


Рисунок 1.2 — Фреймворки Pipeline и End-to-End для беспилотных автомобилей.

Ключевыми компонентами системы автоматического управления беспилотным автомобилем являются локализация, восприятие, планирование и управление [5]. Эти компоненты работают вместе в рамках Pipeline (рисунок 1.2 (a)), обеспечивая беспилотным автомобилям возможность автономной навигации и функционирования в сложных и динамичных условиях. Локализация обеспечивает точное определение местоположения автомобиля в пространстве, используя данные GPS, сенсоров и карт. Восприятие включает в себя интерпретацию данных от сенсоров, таких как камеры, лидар и радар, для наблюдения и понимания окружающей дорожной среды, в том числе обнаружения и отслеживания препятствий, распознавания дорожных знаков и идентификации других транспортных средств и пешеходов. Планирование — это процесс, в ходе которого автомобиль определяет наиболее подходящее действие, например, выбор маршрутов, смену полосы, выполнение маневров, а также решение о том, когда остановиться или продолжить движение. Управление — это заключительная стадия, на которой автомобиль выполняет запланированные действия посредством рулевого управления, ускорения и торможения.

В отличие от вышеописанного, End-to-End фреймворк (рисунок 1.2 (b)) рассматривает всю задачу вождения как единую проблему машинного обучения. Данные с сенсоров используются в качестве входных сигналов, а управляющие команды для рулевого управления, ускорения и торможения — в качестве выходных. End-to-End фреймворк основан на нейронных сетях; модель вождения приобретает знания через имитационное обучение, изучает стратегии вождения с использованием обучения с подкреплением и постоянно совершенствуется благодаря параллельному обучению [5]. Pipeline фреймворк предоставляет интерпретируемость, что помогает анализировать различные управляющие команды и поведение автомобиля, а также выявлять причины ошибок в случае неожиданного поведения. Однако он требует значительных вычислительных ресурсов и множества вручную определенных эвристик, что создает проблемы с обобщением и производительностью в реальном времени. Основанный на обучении End-to-End фреймворк, напротив, улучшает обобщение и устойчивость, но делает это за счет интерпретируемости, что затрудняет отслеживание ошибок или объяснение процессов принятия решений.

Интегрируя передовые сенсоры, сложные алгоритмы и новейшие технологии картографирования, системы автоматического управления АТС обещают обеспечить более безопасный, эффективный и доступный транспорт для общества в целом [6]. Данная диссертация сосредоточена на разработке передовых алгоритмов для адаптивного планирования в рамках Pipeline фреймворка в полностью автономных транспортных средствах на уровне 5 автономии. Наша цель — расширить границы текущих технологических возможностей, прокладывая путь для транспортных средств, которые могут динамически адаптироваться к сложным условиям и сценариям вождения, достигая большего уровня автономности по сравнению с современными стандартами.

1.2 Планирование в беспилотных автомобилях и его вызовы

В системе автоматического управления беспилотным автомобилем система должна перемещаться от начальной точки к месту назначения, следовать глобально запланированному маршруту, планировать локальную

траекторию в соответствии с правилами дорожного движения и избегать статичных и динамических объектов вокруг автомобиля, предвидя их поведение и следуя поведению, подобному тому, что демонстрирует водитель-человек. Слой планирования получает данные от слоя восприятия окружающей среды и локализации, планирует оптимальную траекторию и передает ее слою управления для выполнения.

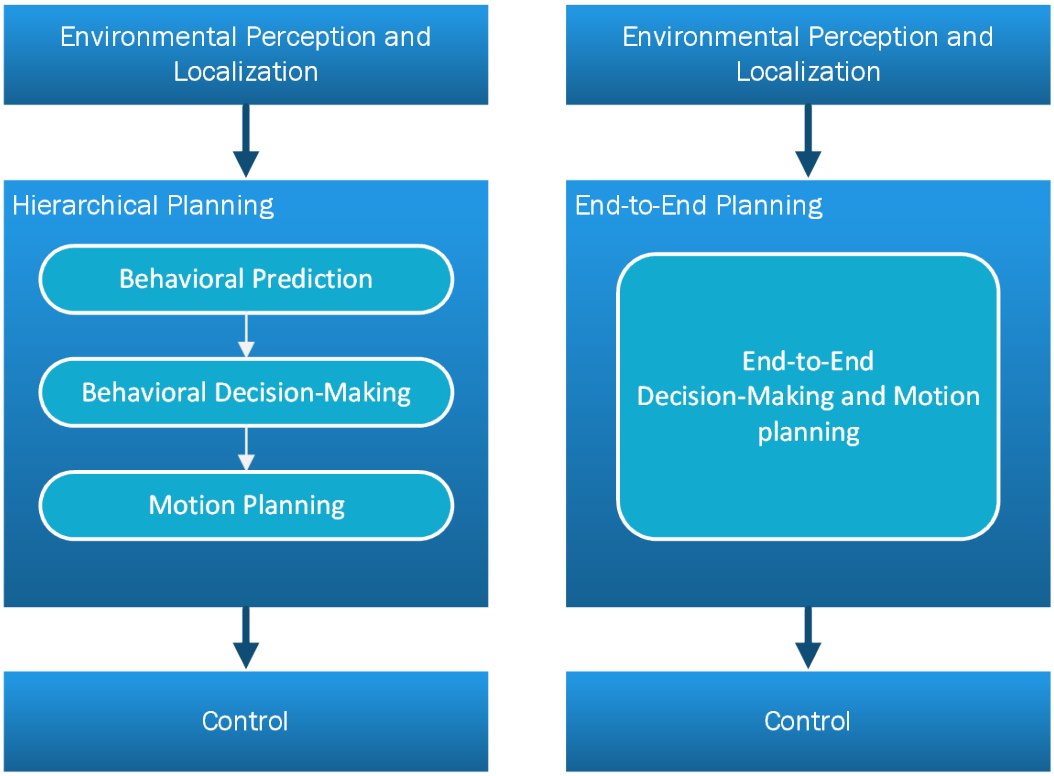


Рисунок 1.3 — Иерархическое и сквозное (End-to-End) планирование в системах автоматического управления беспилотным автомобилем.

Слой планирования включает в себя две основные структуры: иерархическую и сквозную структуру (End-to-End) [23] (рисунок 1.3). В иерархическом планировании сначала предсказывается поведение окружающих препятствий, таких как транспортные средства и пешеходы, после чего осуществляется выбор поведения и планирование движения. В сквозной структуре планирование основывается на нейронных сетях, которые обучаются принятию решений и планированию движения на основе данных восприятия, неявно предсказывая поведение окружающих препятствий.

При иерархическом планировании во время предсказания поведения автономная система вождения использует алгоритмы на основе обучения для предсказания будущего поведения и траекторий других транспортных средств и пешеходов на определённый горизонт предсказания (prediction

horizon), основываясь на их предыдущих состояниях, типах и данных с карты высокого разрешения (HD-карты). При принятии решений о поведении система принимает высокоуровневые решения о маневрах, исходя из глобального маршрута, будущих траекторий препятствий и HD-карты, соблюдая правила дорожного движения и обеспечивая безопасность. Эти решения включают в себя маневры для различных сценариев на многополосных шоссе и в городских условиях, такие как удержание полосы, перестроение, слияние потоков, обгон и проезд перекрестков в транспортном потоке. Планирование движения генерирует траекторию, по которой должен следовать беспилотный автомобиль для выполнения решения о маневре. В иерархической структуре принятие решений о поведении и планирование движения могут быть разделены (декуплированы), когда сначала принимается высокоуровневое решение о поведении, а затем на низком уровне планировщик движения его реализует, либо они могут быть интегрированы.

Разделение принятия решений и планирования траектории может привести к появлению невыполнимых траекторий, что потребует повторного планирования «с нуля» с учетом низкоуровневых ограничений. Интегрированное планирование задач и движений (TAMP, Integrated Task and Motion Planning) [24—26] решает эту проблему, объединяя высокоуровневое планирование задач с низкоуровневым планированием движений для получения выполнимых решений для задач с большим горизонтом, включающих геометрические ограничения и логические рассуждения.

Методы предсказания траектории можно разделить [27] на традиционные методы, методы, основанные на глубоком обучении, и методы, основанные на обучении с подкреплением (RL). Традиционные методы включают в себя различные подходы. Вероятностные модели оценивают распределения вероятностей будущих траекторий с помощью байесовских фильтров, таких как фильтры Калмана и фильтры частиц, чтобы справиться с неопределенностью предсказания [28]; модели, основанные на физике, полагаются на законы физики и принципы кинематики [29]. Подходы, основанные на обучении [30—33], используют машинное обучение и глубокое обучение на больших наборах данных о вождении для предсказания траекторий, при этом перспективными являются рекуррентные нейронные сети (RNNs, Recurrent Neural Networks), сверточные нейронные сети (CNNs, Convolutional Neural

Networks) и гибридные модели. Методы RL обучаются оптимальным стратегиям для предсказания будущих траекторий [34].

Традиционные методы предсказания траекторий эффективны и просты в реализации, но испытывают трудности при работе со сложными взаимодействиями и неопределенностями. Продвинутые подходы машинного обучения, такие как глубокое обучение и обучение с подкреплением (RL), улучшают точность и устойчивость. Глубокое обучение улавливает сложные закономерности и разнообразные сценарии, однако требует больших размеченных наборов данных и вычислительных мощностей, а также сталкивается с проблемами интерпретируемости моделей. Методы RL обучаются на данных и взаимодействиях с окружающей средой для точных предсказаний, но должны сбалансировать сложность алгоритма, доступность обучающих данных и способность к обобщению на различные реальные сценарии.

Принятие решений в автономном вождении обычно осуществляется с использованием как классических методов, так и методов на основе обучения [9]. Классические методы включают в себя подходы, основанные на правилах, оптимизации и вероятностных моделях. Методы, основанные на правилах [10; 11], являются простыми и интерпретируемыми, но испытывают трудности при работе в сложных, динамических условиях. Методы оптимизации [12; 13] превосходно моделируют взаимодействия, однако часто терпят неудачу в реальных сценариях из-за своей зависимости от предположений об «оптимальной стратегии». Вероятностные методы [14; 15], хотя и эффективны в сочетании с другими подходами, как правило, менее продуктивны в сложных условиях.

Методы на основе обучения, включая статистические подходы, глубокое обучение и обучение с подкреплением (RL), предполагают уникальные преимущества и сложности. Статистические методы являются адаптивными, но требуют обширных данных и, как правило, обладают более низкой точностью. Глубокое обучение достигает высокой точности и эффективно использует данные об окружающей среде, однако требует больших наборов данных, значительных вычислительных ресурсов и сталкивается с проблемами интерпретируемости [16]. Обучение с подкреплением (RL) хорошо подходит для работы с неопределенностями и динамическими средами, но сталкивается с серьезными вызовами [17; 18], такими как проверка эффективности

системы, преодоление разрыва между симуляцией и реальностью, повышение эффективности выборок, разработка эффективных функций вознаграждения и обеспечение безопасности в процессах принятия решений. Эти вызовы могут также привести к таким проблемам, как нестабильность и переобучение (overfitting) в автономных системах.

Согласно [35], подходы к принятию решений можно классифицировать на три категории: планирование на основе правил, реактивное и интерактивное планирование. Планирование на основе правил включает в себя подходы, которые принимают решения, используя текущие состояния окружающих транспортных средств без учета их динамики и, следовательно, предсказания их траекторий. Система принятия решений на основе правил включает эвристики, настроенные вручную для различных сценариев вождения. Реактивное планирование относится к алгоритмам, которые принимают решения в ответ на предсказанные траектории окружающих транспортных средств на конечном горизонте. Часто используемый подход основан на симуляции окружающей среды вперед.

Интерактивное планирование включает реакцию окружающих автомобилей в планы беспилотного автомобиля. Подходы, основанные на частично наблюдаемых марковских процессах принятия решений (POMDPs, Partially Observable Markov Decision Processes), классифицируются как планирование с учётом взаимодействия, так как они определяют оптимальные действия беспилотного автомобиля в ответ на ненаблюдаемые намерения окружающих транспортных средств. Другой подход с учетом взаимодействия, основанный на теории игр, выбирает действия беспилотного автомобиля и других окружающих автомобилей, которые максимизируют вознаграждение, оценивая все возможные комбинации действий или предполагая, что агенты имеют различные уровни взаимодействия.

Принятие решений и автоматическое планирование в искусственном интеллекте можно разделить на два основных класса [36]: детерминированное планирование и вероятностное планирование. Рассмотренные ранее методы охватывают весь этот спектр, от детерминированных до вероятностных подходов.

Детерминированное планирование играет важнейшую роль в обеспечении систематического достижения целей агентами в хорошо определенных условиях. Детерминированное планирование включает создание

последовательности действий, которые агент должен выполнить для перехода из начального состояния в желаемое целевое состояние, при этом каждое действие приводит к предсказуемому результату без какой-либо неопределенности. Эта форма планирования основана на предположении, что окружающая среда полностью наблюдаема, эффекты действий известны с уверенностью, а система ведет себя детерминированным образом.

Вероятностное планирование, с другой стороны, занимается принятием решений в условиях, характеризуемых неопределенностью и случайностью. В отличие от детерминированного планирования, где результат каждого действия известен и предсказуем, вероятностное планирование признает, что реальные условия часто включают в себя непредсказуемость, когда действия могут иметь несколько возможных исходов, каждый из которых связан с определенной вероятностью.

1.3 Детерминированное планирование

Классическое планировочное пространство определяется как тройка $\Sigma = (S, A, \mathcal{T})$ или четверка $\Sigma = (S, A, \mathcal{T}, cost)$, где:

- S — пространство состояний — это конечное множество состояний, в которых может находиться система.
- A — пространство действий — это конечное множество действий, которые может выполнить актор.
- $\mathcal{T} : S \times A \to S$ — функция перехода состояний. Если (s, a) находится в области определения $\mathcal{T}$, то действие a применимо в состоянии s, при этом $\mathcal{T}(s, a)$ является предсказанным результатом. В противном случае действие a не применимо в состоянии s.
- $cost : S \times A \to [0,\infty)$ — это частичная функция, имеющая ту же область определения, что и $\mathcal{T}$. Эта стоимость может представлять собой денежные затраты, время или другой параметр, который требуется минимизировать.

В детерминированном планировании предполагается, что мы можем с уверенностью предсказать, какое состояние будет достигнуто, если действие a выполнено в состоянии s.

Каждое состояние $s \in S$ представляет собой описание свойств различных объектов в окружающей среде планировщика. Свойство называется *жестким (rigid)*, если оно остается неизменным в каждом состоянии из S, и *изменяющимся (varying)*, если оно может различаться от одного состояния к другому. Для представления объектов и их свойств используются три множества B, R и X, которые должны быть конечными:

- B — множество имен всех объектов, а также любых математических констант, которые могут потребоваться для представления свойств этих объектов. Множество B обычно делится на различные подмножества (роботы, препятствия, математические константы и т.д.).
- R — множество жестких отношений, которое представляет жесткие свойства Σ. Каждое $r \in R$ будет n-арным (для некоторого n) отношением над множеством B.
- X — множество синтаксических терминов, называемых переменными состояния, представляет изменяющиеся свойства Σ, при этом значение каждого $x \in X$ зависит исключительно от состояния s.

Набор объектов и свойств, входящих в множества B, R и X, зависит от того, какие части окружающей среды планировщику нужно учитывать.

Переменная состояния над множеством B является синтаксическим термином

$$x = sv\,(b_1, \ldots, b_k) \tag{1.1}$$

где sv — символ, называемый именем переменной состояния, и каждый b_i является элементом множества B. Каждая переменная состояния x имеет диапазон значений, $\mathrm{Range}(x) \subseteq B$, который представляет собой множество всех возможных значений для x.

Функция присвоения переменной над X — это функция s, которая отображает каждую переменную $x_i \in X$ в значение $z_i \in \mathrm{Range}\,(x_i)$. Если $X = \{x_1, \ldots, x_n\}$, то, поскольку функция является множеством упорядоченных пар, ее можно записать как набор утверждений:

$$s = \{x_1 = z_1, x_2 = z_2, \ldots, x_n = z_n\} \tag{1.2}$$

Так как X и B конечны, то и количество функций присвоения переменных также конечно. Пространство состояний переменных состояния — это множество S функций присвоения переменных для некоторого множества

переменных состояния X. Каждая функция присвоения переменных в S называется состоянием в S.

Положительный литерал (positive literal), или атом (атомарная формула), — это выражение, имеющее одну из следующих форм:

$$\mathrm{rel}(z_1, \ldots, z_n) \quad or \quad \mathrm{sv}(z_1, \ldots, z_n) = z_0; \tag{1.3}$$

где rel — это имя жёсткого отношения, sv — имя переменной состояния, а каждый z_i является либо переменной (обычной математической переменной, а не переменной состояния), либо именем объекта. Отрицательный литерал — это выражение, имеющее одну из следующих форм:

$$\neg\mathrm{rel}(z_1, \ldots, z_n) \quad or \quad \mathrm{sv}(z_1, \ldots, z_n) \neq z_0 \tag{1.4}$$

Литерал называется замкнутым (ground), если он не содержит переменных, и незамкнутым (unground) в противном случае.

Предположим, что R и X — это множества жёстких отношений и переменных состояния для множества объектов B, а S — это пространство состояний переменных состояния над X. Шаблон действия для S — это tuple $\alpha = (\mathrm{head}(\alpha), \mathrm{pre}(\alpha), \mathrm{eff}(\alpha), \mathrm{cost}(\alpha))$ or $\alpha = (\mathrm{head}(\alpha), \mathrm{pre}(\alpha), \mathrm{eff}(\alpha))$, элементы которого определяются следующим образом:

– head (α) — это синтаксическое выражение следующей формы:

$$\mathrm{act}\,(z_1, z_2, \ldots, z_k)\,,$$

где act — это символ, называемый именем действия, а $z_1, z_2, \ldots, z_k$ — переменные, называемые параметрами. Параметры должны включать в себя все переменные (обычные переменные, не переменные состояния), которые появляются где-либо в $\mathrm{pre}(\alpha)$ и $\mathrm{eff}(\alpha)$. Каждый параметр z_i имеет диапазон возможных значений, Range $(z_i) \subseteq B$.

– $\mathrm{pre}(\alpha) = \{p_1, \ldots, p_m\}$ — это набор предусловий, каждое из которых является литералом.

– $\mathrm{eff}(\alpha) = \{e_1, \ldots, e_n\}$ — это набор эффектов, каждый из которых представляет собой выражение следующей формы:

$$sv\,(t_1, \ldots, t_j) \leftarrow t_0$$

где $sv\,(t_1, \ldots, t_j)$ является целью эффекта, а t_0 — значением, которое будет присвоено. Ни одна цель не может появляться в $\mathrm{eff}(\alpha)$ более одного раза.

– $\text{cost}(\alpha)$ — это число $c > 0$, обозначающее стоимость применения действия. Если оно опущено, то по умолчанию принимается, что $\text{cost}(\alpha) = 1$.

Действие переменной состояния (state-variable action) — это замкнутый (ground) экземпляр a шаблона действия α, который удовлетворяет следующим требованиям: все литералы жестких отношений в $\text{pre}(a)$ должны быть истинными в R и ни одна цель не может появляться более одного раза в $\text{eff}(a)$. Если a является действием, а состояние s удовлетворяет $\text{pre}(a)$, то a применимо в s и предсказанный результат его применения — это состояние:

$$\begin{aligned}\mathcal{T}(s,a) =& \{(x,w) \mid \text{eff}(a) \text{ содержит эффект} x \leftarrow w\} \\ & \cup \{(x,w) \in s \mid x \text{ не является целью какого-либо эффекта в eff } (a)\}\end{aligned} \tag{1.5}$$

Если a не применимо в s, то $\mathcal{T}(s,a)$ не определено. Таким образом, если a применимо в s, то

$$(\mathcal{T}(s,a))(x) = \begin{cases} w, & \text{if eff}(a) \text{ contains an effect } x \leftarrow w \\ s(x), & \text{otherwise.} \end{cases} \tag{1.6}$$

Это приводит к следующему выводу: если B, R, X и S определены, как указано ранее, и $\mathcal{A}$ — это множество шаблонов действий, таких что для каждого $\alpha \in \mathcal{A}$ диапазон каждого параметра является подмножеством B, а $A = \{$ все действия переменных состояния (state-variable actions), которые являются экземплярами элементов из $\mathcal{A}\}$. Если $\mathcal{T}$ — это функция перехода, определенная как ранее, то $\Sigma = (S, A, \mathcal{T}, cost)$ представляет собой планировочную область переменных состояния (state-variable planning domain).

План π в области планирования переменных состояния — это конечная последовательность действий:

$$\pi = \langle a_1, a_2, \ldots, a_n \rangle \tag{1.7}$$

Длина плана — это $|\pi| = n$, а его стоимость — это сумма стоимостей действий: $\text{cost}(\pi) = \sum_{i=1}^{n} \text{cost}\,(a_i)$.

План $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ применим в состоянии s_0, если существуют состояния $s_1, \ldots, s_n$, такие что $\mathcal{T}\,(s_{i-1}, a_i) = s_i$ для $i = 1, \ldots, n$, и это определяется следующим образом:

$$\begin{aligned}\mathcal{T}\,(s_0, \pi) &= s_n \\ \widehat{\mathcal{T}}\,(s_0, \pi) &= \langle s_0, \ldots, s_n \rangle\end{aligned} \tag{1.8}$$

Задача планирования с переменными состояния — это тройка $\mathcal{P} = (\Sigma, s_0, G)$, где Σ — область планирования с переменными состояния, s_0 — состояние, называемое начальным состоянием, а G — множество замкнутых литералов, называемых целью. Решением для $\mathcal{P}$ является любой план $\pi = \langle a_1, \ldots, a_n \rangle$, такой что состояние $\mathcal{T}(s_0, \pi)$ удовлетворяет G.

1.4 Детерминированное планирование с использованием сэмплеров

Язык определения предметной области планирования (PDDL, Planning Domain Definition Language) — это формальный язык, разработанный в конце 1990-х годов для описания детерминированных задач планирования в искусственном интеллекте. Для эффективного моделирования задачи планирования на PDDL необходимы два основных компонента: домен (domain) и задача (problem). Домен Σ определяет общие действия и объекты, вовлеченные в процесс планирования, в то время как задача $\mathcal{P}$ описывает начальное состояние и желаемые цели для конкретного экземпляра домена. PDDL2.1 [37] ввел числовые флуенты (numeric fluents) и метрики плана, улучшив язык за счет возможности интеграции метрик оптимизации в задачи планирования.

Домен планирования определяется как $\Sigma = \langle T, P, F, A \rangle$, где:

- *Типы* T — множество типов, используемых для классификации объектов в домене,
- *Предикаты* P - множество предикатов, $P = \{p_1, p_2, \ldots, p_p\}$, где каждый предикат p может быть применен к определённой последовательности объектов $\{b_1, b_2, \ldots, b_o\}$ для формирования литералов,
- *Функции* F - множество функций (переменных), $F = \{f_1, f_2, \ldots, f_f\}$, где каждая функция f может быть применена к определённой последовательности объектов, присваивая им значение,
- *Действия* A - множество действий, $A = \{a_1, a_2, \ldots, a_j\}$, где каждое действие a определяется набором аргументов объекта $\bar{b} = \langle b_1, b_2, ..., b_r \rangle$ и набором предусловий $pre(a)$ для $\bar{b}$, включающих в себя положительные литералы $pre^+(a)$ и отрицательные литералы $pre^-(a)$, которые должны выполняться для применения действия, а также набором

эффектов $eff(a)$ для $\bar{b}$, включающих в себя положительные $eff^+(a)$ и отрицательные литералы $eff^-(a)$, являющиеся результатом применения действия.

Действие a применимо в состоянии s, если

$$\left(pre^{+}(a(\bar{b})) \subseteq s \right) \wedge \left(pre^{-}(a(\bar{b})) \cap s = \varnothing \right)$$

и результирующее состояние S' после применения действия a в состоянии S:

$$s' = \left(s \backslash eff^{-}(a(\bar{b})) \right) \cup eff^{+}(a(\bar{b}))$$

Задача планирования определяется как $\mathcal{P} = \langle B, s_0, g \rangle$, где:

– *Объекты* B — множество объектов, являющихся экземплярами (instances) определенных типов,
– *Начальное состояние* s_0 — множество положительных литералов, описывающих начальное состояние,
– *Целевое состояние* G — множество как положительных, так и отрицательных литералов, выражающих целевое состояние.

План $\pi = \left[a_1\left(\bar{b}_1\right), \ldots, a_k\left(\bar{b}_k\right) \right]$ — это конечная последовательность из k экземпляров действий, такая что каждое действие $a_i\left(\bar{b}_i\right)$ применимо в состоянии s_{i-1}, приводя к состоянию s_i. Целевое состояние G достигается после выполнения всей последовательности π.

Поток (stream) $st(\bar{b})$ — это условная функция (sampler) набора объектных аргументов *входа* $\bar{b} = \langle b_1, b_2, ..., b_m \rangle$. Эта функция может изменять задачу планирования $\mathcal{P}$, генерируя *выходной* набор новых объектов $\bar{r} = \langle r_1, r_2, ..., r_l \rangle$, а также набор сертифицированных фактов, связанных с ними, где $st.cert = p \mid \forall \bar{b} \in \bar{b}, \forall \bar{r} \in st(\bar{b}).p(\bar{b} + \bar{r})$. Поток может возвращать None, если генерация новых объектов невозможна. Поток может быть применен к входным параметрам $\bar{b}$ только в том случае, если существует набор положительных литералов p, связанных с ними в домене, где $st.dom = \{p \mid \forall \bar{b} \in \bar{b} \cdot p(\bar{b})\}$. Поток может изменять начальное состояние s_0 в задаче $\mathcal{P}$.

Когда потоки интегрируются в автоматическое планирование на PDDL, процесс выполняется итеративно на различных уровнях планирования. Изначально применимые потоки используются для модификации начального состояния. Затем планировщик PDDL выполняет эвристический поиск для нахождения оптимального плана. Если план не найден, уровень планирования повышается, и процесс повторяется. Эта итеративная процедура продолжается

до тех пор, пока не будет либо найден оптимальный план, либо достигнут максимальный уровень планирования.

1.5 Вероятностное планирование

Обычно формальной моделью вероятностного планирования являются марковские процессы принятия решений (MDPs). MDP представляет собой недетерминированную систему переходов состояний вместе с распределением вероятностей и распределением затрат. Распределение вероятностей определяет, какова вероятность перехода в состояние s' при выполнении действия a в состоянии s.

Вероятностный домен планирования определяется как $\Sigma = (S, A, \mathcal{T}, \Pr, \text{cost})$, где:

- S и A — это конечные множества состояний и действий соответственно;
- $\mathcal{T} : S \times A \to 2^S$ — это функция перехода состояний, которая соответствует следующему стационарному марковскому распределению вероятностей;
- $\Pr(s' \mid s, a)$ — это вероятность достижения состояния $s' \in \mathcal{T}(s, a)$, когда действие a выполняется в состоянии s; она такова, что $\Pr(s' \mid s, a) \neq 0$ тогда и только тогда, когда $s' \in \mathcal{T}(s, a)$; и
- cost: $S \times A \to \mathbb{R}^+$ — это функция затрат: $\text{cost}(s, a)$ — это стоимость действия a в состоянии s.

1.6 Вероятностное планирование с неопределенностью

Модели MDP предполагают, что после каждого перехода состояния актор знает, в какое состояние s он перешел; затем он выполняет действие $\pi(s)$, соответствующее состоянию s. В модели частично наблюдаемых марковских процессов принятия решений (POMDP) считается, что актор не знает своего текущего состояния, но знает значение некоторой переменной наблюдения o. У него также есть распределение вероятностей $\Pr(o \mid s, a)$ для наблюдения

o после выполнения действия a в состоянии s. Это дает ему вероятностное распределение возможных состояний, в которых он может находиться, называемое степенью уверенности текущего агента (belief): $b(s \mid a, o)$. В работе [38] было показано, что последнее наблюдение o не обобщает прошлое выполнение, а последнюю степень уверенности (belief) обобщает. Следовательно, задача планирования POMDP может быть рассмотрена как MDP-задача в пространстве степеней уверенности. Начинают начальной степени уверенности b_0 (распределение для начальных состояний) и вычисляет стратегию, которая дает для каждой точки уверенности b действие $\pi(b)$, ведущее к цели, также выраженной в пространстве уверенности.

Чтобы узнать оптимальную стратегию, используется обучение с подкреплением. Основные элементы обучения с подкреплением включают агента и окружающую среду. Агент воздействует на среду через свои действия, а среда реагирует на эти действия с помощью подкрепляющего сигнала.

Среда описывается марковским процессом принятия решений (S, A, p, r, γ), где

- S — это пространство состояний,
- A — пространство действий,
- $p : p(* \mid s, a)$ — модель переходов, которая представляет собой распределение по S,
- $R : S \times A \times S \longrightarrow R$ — функция вознаграждения,
- $\gamma \in [0,1)$ — коэффициент дисконтирования.

Действия, выполняемые агентом, описываются стратегией $\pi(* \mid s)$. В этой работе стратегия агента $\pi(* \mid s)$ представляет собой условное распределение на пространстве A: если действия дискретные, то распределение также дискретное, а для непрерывных действий оно выражается в виде многомерного нормального распределения.

Цель агента заключается в оптимизации функции полезности:

$$J(\pi) = \mathbb{E}_{\tau \sim \pi}[R(\tau)] \tag{1.9}$$

где $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$ - это траектория агента, T — максимальная продолжительность эпизода, $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r\left(s_t, a_t, s_{t+1}\right)$ — отдача эпизода.

Для оценки оптимальности стратегии вводится функция полезности для состояния $s \in S$:

$$V^{\pi}(s) := \mathbb{E}_{\tau \sim \pi}\left[R(\tau) \mid s_0 = s\right] \tag{1.10}$$

Также вводится функция полезности для состояния-действия $a \in A$:

$$Q^{\pi}(s,a) := \mathbb{E}_{\tau\sim\pi}\left[R(\tau) \mid s_0 = s, a_0 = a\right] \tag{1.11}$$

Оптимальные функции полезности, таким образом, принимают следующий вид:

$$\begin{aligned} V^{*}(s) &:= \max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[R(\tau) \mid s_0 = s\right] \\ Q^{*}(s,a) &:= \max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[R(\tau) \mid s_0 = s, a_0 = a\right] \end{aligned} \tag{1.12}$$

Во многих методах обучения с подкреплением (RL) стратегия π и функции полезности параметризуются нейронными сетями, таким образом: $\pi = \pi_{\theta}, V^{\pi} = V_{\varphi}^{\pi_{\theta}}, Q^{\pi} = Q_{\varphi}^{\pi_{\theta}}$.

Для поиска оптимальной стратегии в соответствии с функцией полезности $J\left(\pi_{\theta}\right)$ используется градиент стратегии:

$$\nabla_{\theta} J(\pi) = \nabla_{\theta} \mathbb{E}_{\tau\sim\pi_{\theta}}[R(\tau)] \tag{1.13}$$

После раскрытия градиент стратегии преобразуется в удобную для численных методов оптимизации формулу:

$$\nabla_{\theta} J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum \nabla_{\theta} \log \pi_{\theta}\left(a_t \mid s_t\right) R(\tau)\right] \tag{1.14}$$

В задачах управления автономным транспортом предполагается, что агент не обладает полной информацией о среде и использует данные с камер и лидаров, а также информацию о цели. Поэтому в данной работе рассматривается целеобусловленный частично наблюдаемый марковский процесс принятия решений, определяемый как $(S, G, A, O, T, r, \gamma)$, где:

– O — пространство наблюдений

– G — пространство целей, где $G = S$.

Глава 2. Обзор литературы

В динамично развивающейся области беспилотного вождения ведется значительное количество исследований, направленных на повышение безопасности, эффективности и надежности автономных транспортных средств. В данной главе представлен всеобъемлющий обзор литературы, обобщающий ключевые вклад и методологии в критически важных областях, включая предсказание движения, принятие решений и планирование траекторий. Рассматривая текущее состояние исследований, настоящий обзор подчеркивает значительные достижения и выявляет текущие вызовы, которые определяют будущее развитие беспилотных автомобилей.

Глава организована следующим образом. В разделе 2.1 рассматриваются методы предсказания движения, которые играют важную роль в предвидении перемещений других участников дорожного движения, тем самым усиливая процессы принятия решений автономными системами. Затем в разделе 2.2 исследуются стратегии принятия решений, сосредоточенные на том, как эти системы интерпретируют сложные данные об окружающей среде, чтобы выбрать наиболее подходящие действия. После этого в разделе 2.3 рассматриваются методы планирования движения, описывающие способы, используемые для создания безопасных и осуществимых траекторий движения беспилотных автомобилей в различных сценариях.

Наконец, в разделе 2.4 анализируются подходы, объединяющие принятие решений с планированием движения, с оценкой как разделенных, так и интегрированных структур. В этом разделе подробно рассматривается применение этих подходов к конкретным задачам вождения, таким как обгон, смена полосы движения и парковка, а также обсуждаются компромиссы и инновации, связанные с гармонизацией высокоуровневого принятия решений и низкоуровневого управления движением.

Посредством данной структурированной аналитики глава стремится предоставить ясное понимание современных методов и тех вызовов, которые необходимо преодолеть для дальнейшего повышения возможностей беспилотных автомобилей.

2.1 Предсказание движения

Предсказание движения, или предсказание траектории, играет важную роль в системах автономного вождения. Точное прогнозирование движения других участников дорожного движения повышает возможности автономной системы в принятии решений, позволяя ей более эффективно реагировать на динамичную и непредсказуемую среду и адаптироваться к различным дорожным сценариям.

Методы предсказания траекторий движения можно классифицировать на традиционные методы, методы глубокого обучения и подходы на основе обучения с подкреплением (RL) [27]. К традиционным методам относятся вероятностные модели, которые используют байесовские фильтры, такие как фильтры Калмана и фильтры частиц, для оценки вероятностных распределений будущих траекторий и управления неопределенностями в предсказаниях [28], а также физические модели, основанные на законах физики и кинематике [29]. Методы, использующие машинное и глубокое обучение [30—33], применяют эти техники на больших наборах данных о движении, при этом рекуррентные нейронные сети (RNN), сверточные нейронные сети (CNN) и гибридные модели демонстрируют высокий потенциал. В RL-подходах акцент делается на изучении оптимальных стратегий для предсказания будущих траекторий [34].

Несмотря на то, что традиционные методы предсказания траекторий эффективны и просты в реализации, они часто не справляются со сложными взаимодействиями и неопределенностями. Передовые подходы машинного обучения, такие как глубокое обучение и RL, обеспечивают повышенную точность и устойчивость. Методы глубокого обучения отлично справляются со сложными закономерностями и сценариями, но они требуют больших наборов данных с метками, значительной вычислительной мощности и создают проблемы с интерпретацией моделей [17]. Методы RL, которые учатся на основе данных и взаимодействий с окружающей средой, позволяют добиться высокой точности предсказаний. Тем не менее, они должны обеспечивать баланс между сложностью алгоритмов, доступностью обучающих данных и способностью к обобщению в различных сценариях реального мира.

Методы, основанные на глубоком обучении, в последнее время привлекают большое внимание к предсказанию траекторий движения беспилотных автомобилей. Искусственные нейронные сети стали очень эффективными в предсказании траекторий благодаря наличию больших наборов данных с метками для сценариев вождения и расширению вычислительных возможностей. Эти сети отлично справляются с обнаружением сложных закономерностей и взаимосвязей в обширных массивах данных.

Широкие исследования [20; 31; 33] были посвящены усовершенствованию архитектур моделей для предсказания траекторий. В этих моделях используются различные нейронные архитектуры [27], включая рекуррентные нейронные сети (RNNs, Recurrent Neural Networks), такие как сети с длинной кратковременной памятью (LSTMs, Long Short-Term Memory Networks) и рекуррентные блоки с управлением (GRUs, Gated Recurrent Units), а также конволюционные нейронные сети (CNNs, Convolutional Neural Networks), графовые нейронные сети (GNNs, Graph Neural Networks), генеративные адверсарные сети (GANs, Generative Adversarial Networks) и трансформаторные сети (Transformer Networks), для прогнозирования движения объектов. Эти модели высокоэффективны при улавливании временных и пространственных зависимостей в данных о траекториях, что позволяет делать точные предсказания будущих движений. Они специально разработаны для адаптации к сложности окружающей среды и качеству доступных сенсорных данных.

Несколько исследователей предложили модели, объединяющие архитектуры RNN и CNN для управления временной и пространственной информацией в целях предсказания траектории. В работе [30] авторы представляют модель кодера-декодера LSTM, дополненную конволюционным социальным объединением (convolutional social pooling). Этот подход учитывает взаимодействие между автомобилями, используя конволюционные социальные слои для более эффективного изучения параметров, связанных с взаимодействием, чем традиционные полностью связанные слои. При оценке на общедоступных наборах данных NGSIM US-101 и I-80 модель показала лучшие результаты по сравнению с полностью связанной сетью в отношении среднеквадратичной ошибки предсказания и отрицательного логарифмического правдоподобия истинных будущих траекторий. В исследовании также представлен качественный анализ предсказаний для различных сценариев движения. Однако в качестве ограничения отмечается,

что модель опирается только на треки автомобилей, поскольку в ней не учитываются данные с карты.

Авторы исследования [33] предложили новый метод, основанный исключительно на архитектуре на основе трансформаторов, который объединяет особенности карты с особенностями агента для предсказания траектории движения целевого агента. Они использовали векторное представление, ориентированное на агента, преобразуя все входные данные в локальные координаты каждой отдельной цели. Демонстрируя высокую эффективность при предсказании траектории движения одного агента, этот подход, тем не менее, имеет ограниченную масштабируемость и сложен в реализации при увеличении количества агентов в среде движения — обычный сценарий для автономного вождения, где обычно присутствует множество агентов.

В другом исследовании [31] ученые интегрировали кодеры Transformer с CNN для разработки модели предсказания траекторий на основе запросов движения. Этот подход использует кодер Transformer, основанный на запросах движения, для захвата временных зависимостей из исторических последовательностей траекторий. Кроме того, модель включает в себя модуль социального взаимодействия, который использует конволюционные сети и операции объединения (pooling) для извлечения семантической информации с дороги и кодирования траекторий соседних автомобилей в пространственный социальный тензор (tensor). Оценка наборов данных NGSIM US-101 и I-80 показала повышенную точность предсказаний по сравнению с предыдущими методами. Однако эффективность модели сильно зависит от качества пространственного социального тензора, неточности в котором могут понизить общую точность предсказания. Кроме того, в сценариях с плотным трафиком сложность обработки плотных пространственных взаимодействий может поставить под вопрос эффективность модели и ее работу в реальном времени.

Вместо того чтобы полагаться на оценки целей, авторы исследования [39] задействовали предложения траекторий в качестве опорных точек в своем подходе. Они ввели представление, ориентированное на агента, в котором использовались характеристики агента в полярных координатах, которые затем преобразовывались в характеристики Фурье. Несмотря на то, что этот метод избавляет от необходимости пересчитывать характеристики, ориентированные на агента, на каждом временном шаге, он ограничен коротким горизонтом

предсказания. В частности, предложенный метод предсказывает траектории окружающих агентов только на три секунды, что считается краткосрочным предсказанием. Это ограничение влияет на надежность модели принятия решений, которая зависит от него, поскольку может упустить важную информацию о будущих движениях агентов.

В работе [20] исследователи представили подход предсказания траекторий, ориентированный на запросы (Query-Centric). Сначала они использовали запросы без якорей (anchor-free queries), чтобы итеративно генерировать предложения траекторий. Затем модуль уточнения использовал эти предложения в качестве якорей и применял запросы, основанные на якорях, для дальнейшего улучшения траекторий. Хотя QCNet может столкнуться с ограничениями в сценариях, где сгенерированные якоря неточно отображают истинное положение дел, а ее эффективность может быть поставлена под сомнение в плотных и динамичных дорожных сценах, она, тем не менее, продемонстрировала превосходные предсказания. В частности, QCNet преуспела в бенчмарке Argoverse 2 Motion Forecasting Benchmark [40], который получил признание благодаря обширному набору данных, разнообразным сценариям и строгим метрикам оценки автономного вождения.

В заключение следует отметить, что предсказание поведения очень важно для систем автономного вождения, поскольку точное прогнозирование движения других участников дорожного движения улучшает процесс принятия решений в динамичной среде. Хотя традиционные методы эффективны, они часто не справляются со сложными взаимодействиями. В отличие от них, передовые методы, такие как глубокое обучение и обучение с подкреплением (RL), обеспечивают повышенную точность и адаптивность, но требуют значительных вычислительных ресурсов и больших массивов данных. Последние достижения направлены на интеграцию сложных нейронных архитектур, таких как RNNs, CNNs, GNNs, GANs и Transformers, эффективно отражающих как временные, так и пространственные зависимости. Однако каждый метод имеет свои ограничения, в частности, в масштабируемости, интерпретируемости и применимости в реальном времени. По мере развития исследований совершенствование этих моделей и разработка более надежных и масштабируемых решений будут играть ключевую роль в повышении эффективности систем автономного вождения.

2.2 Принятие решений

Планирование поведения или принятие решений включает в себя анализ окружающей среды, предсказания будущих движений других автомобилей, пешеходов и объектов в окружении и определение оптимального курса действий [6]. Действия предполагают принятие решений о различных маневрах, таких как уступка или следование за впереди едущим автомобилем, соблюдение полосы движения, смена полосы движения, слияние, обгон, уступка и проезд перекрестков.

Конечные автоматы (FSMs, Finite State Machines) были ранним и распространенным подходом к принятию решений в самоуправляемых автомобилях, начиная с DARPA Urban Challenge в 2007 году [11; 41; 42]. FSM основываются на математической модели вычислений, которая работает в рамках множества предопределенных состояний, переходя из одного состояния в другое на основе правил. Этот метод эффективно моделирует конкретные дорожные ситуации, но не справляется с динамическими сценариями и не может справиться с неопределенностью окружающей среды. Кроме того, FSM становятся неуправляемыми в сложных системах из-за необходимости иметь полностью связный граф переходов между состояниями, что приводит к трудностям при модернизации и модификации. Иерархические конечные автоматы (HFSMs, Hierarchical Finite State Machines)[43] были введены для решения этих проблем путем группирования состояний в суперсостояния, которые наследуют переходы для уменьшения избыточности. Однако HFSM по-прежнему страдают от ограниченной возможности повторного использования.

Деревья поведения (BTs, Behavior Trees) развились из HFSMs, предлагая более модульную, удобную в обслуживании и масштабируемую структуру для систем принятия решений. В отличие от FSMs, BTs имеют древовидную структуру, которая неявно обрабатывает переходы состояний, что снижает сложность. Несмотря на некоторые ограничения, BTs считаются мощным инструментом для детерминированного принятия решений в автоматическом вождении [44]. Однако моделирование оптимальных решений в неопределенных сценариях реального мира остается сложным и чреватым ошибками. Марковские процессы принятия решений (MDP) предлагают решение,

расширяя FSM стохастическими функциями перехода, что позволяет моделировать последовательные задачи принятия решений в условиях неопределенности. MDP позволяют агенту влиять на результаты через действия, основанные на вознаграждении, обеспечивая надежную основу для принятия решений в неопределенных средах, включая частично наблюдаемые ситуации, как в частично наблюдаемых марковских процессах принятия решений (POMDPs)[45—50]. Однако MDP требуют больших вычислительных затрат из-за рассмотрения всех возможных действий.

Традиционные методы принятия решений в автономном вождении сталкиваются со сложными сценариями, в частности с взаимодействием с непредсказуемыми водителями-людьми, и с проблемами, связанными с ростом числа правил/размеров. Эта проблема привела к внедрению методов, основанных на обучении и использующих большие массивы данных для разработки адаптивных стратегий вождения. Эти подходы на основе ИИ, такие как имитационное обучение и глубокое обучение с подкреплением (DRL, Deep Reinforcement Learning), обеспечивают гибкость при работе с различными сценариями, но страдают от ограниченной интерпретируемости и непредсказуемого поведения в новых ситуациях.

Глубокое обучение обеспечивает высокую точность и полностью использует данные об окружающей среде, но требует больших наборов данных и вычислительной мощности, а также создает проблемы с интерпретируемостью [16]. Обучение с применением подкрепления хорошо справляется с неопределенностью и динамикой, но сталкивается с рядом серьезных проблем [17; 18]. Эти проблемы включают проверку эффективности систем на основе RL, преодоление разрыва между симуляцией и реальностью, достижение эффективности выборки, разработку эффективных функций вознаграждения и интеграцию безопасности в процессы принятия решений для автономных агентов, что также может привести к проблемам стабильности и переборчивости (overfitting).

Для планирования поведения автономных систем было изучено несколько альтернативных методов. Нечеткая логика обеспечивает гибкость в управлении неопределенными данными, но становится все более сложной по мере добавления новых правил. Машины опорных векторов классифицируют намерения агентов на основе текущих данных, однако их возможности ограничены из-за неспособности учитывать историческую информацию.

Эволюционные методы, вдохновленные естественной эволюцией, используют такие механизмы, как мутация и отбор, для принятия решений. Искусственные нейронные сети (ANNs, Artificial Neural Networks) хорошо подходят для сложных сценариев, но им часто не хватает последовательных причинно-следственных объяснений, а глубокие нейронные сети особенно хорошо работают с большими наборами данных. Рекуррентные нейронные сети (RNNs), которые отражают динамику во времени, эффективны в сложных ситуациях, например, на переполненных перекрестках. Однако каждый из ранее упомянутых подходов имеет свои преимущества и сталкивается с определенными вызовами, что подчеркивает постоянную потребность в инновационных решениях для планирования поведения беспилотных автомобилей.

2.3 Планирование движения

Подходы к планированию движения для беспилотных автомобилей, изначально разработанные сообществом робототехников, эволюционировали для решения уникальных задач, связанных с дорожной средой с интенсивным движением и особыми правилами. Задача поиска оптимального пути в условиях ограничений является сложной и часто требует применения методов, требующих больших вычислительных затрат. Хотя точные решения возможны в определенных сценариях, таких как криволинейные траектории в среде без препятствий, большинство реальных случаев неразрешимы, что приводит к использованию численных методов аппроксимации.

Основные стратегии планирования движения [36] включают в себя геометрические методы, которые используют параметрические и полупараметрические кривые для определения траекторий; вариационные и оптимальные методы, которые ставят задачу как нелинейную оптимизацию в пространстве функций [51]; Подходы, основанные на выборке, такие как алгоритм RRT (Rapidly-exploring Random Tree) и его адаптации [52], которые инкрементально строят дерево достижимых состояний для определения осуществимых путей; и методы поиска графов, которые дискретизируют

пространство движения и применяют алгоритмы для поиска оптимальных путей.

Управление с прогнозирующими моделями (MPC, Model Predictive Control) — распространенная техника в рамках вариационных и оптимальных методов, используемых для планирования траектории движения беспилотных автомобилей. Этот подход хорошо подходит для динамичных и непредсказуемых сценариев вождения, когда условия не могут быть полностью предвидены. MPC решает эту задачу путем рекурсивного решения задач оптимизации траектории с конечным временем и использования динамических моделей для предсказания будущих состояний и параметров управления. Эта возможность позволяет MPC учитывать изменяющиеся условия при планировании. Значительное преимущество MPC заключается в его способности явно определять и постоянно обновлять динамические ограничения, основанные на модели автомобиля и условиях движения. Более того, надежные формулировки MPC позволяют эффективно управлять соображениями безопасности, такими как сохранение границ дороги и избежание столкновений, что делает его важным инструментом для обеспечения безопасного и эффективного управления транспортным средством в сложных условиях движения.

В методах выборки (sampling) пространство поиска дискретизируется, что приводит к трудностям при поиске решений в небольших или труднодоступных областях. Для устранения ограничений дискретизации используются методы оптимизации, позволяющие минимизировать функцию стоимости на основе множества ограничений на состояние и входные данные. Эти методы привлекательны тем, что быстро сходятся к локально оптимальным решениям; однако их применение ограничено тенденцией сходиться только к локальным минимумам. Методы поиска по графу пытаются преодолеть эту проблему, проводя глобальный поиск в дискретизированной версии пространства путей, созданного примитивами движения. В некоторых случаях фиксированная природа такой дискретизации графа может привести к неправильным или неоптимальным решениям. Методы инкрементального поиска могут быть ценными в таких сценариях, предлагая более адаптируемые и выполнимые пути для решения задач планирования движения, если такие пути существуют. Однако вычислительное время, необходимое для проверки полноты этих методов, может оказаться непрактичным для приложений реального времени. Несмотря на успешное применение подходов машинного обучения для решения

задач планирования траектории, они пока не отвечают строгим требованиям безопасности, предъявляемым к автомобильным приложениям, в первую очередь из-за сложности их проверки.

2.4 Комбинация принятия решений и планирования движения

В системе планирования беспилотных автомобилей, в зависимости от подхода к проектированию, принятие решений и планирование движения могут быть организованы как интегрированным, так и раздельным образом. В разделенной системе решения высокого уровня принимаются независимо от процесса планирования движения. Такой подход обеспечивает четкое разделение задач, позволяя модулю принятия решений сосредоточиться на выборе объективных маневров, в то время как модуль планирования движения генерирует целесообразные траектории для выполнения этих маневров. Раздельная схема завоевала популярность благодаря своей простоте и возможности оптимизировать и тестировать каждый компонент по отдельности. Однако по мере того, как беспилотные автомобили сталкиваются со все более сложными и динамичными условиями вождения, ограничения этого подхода становятся все более очевидными. Отсутствие плавной интеграции между принятием решений и планированием движения может привести к появлению невыполнимых траекторий, требующих повторного планирования с нуля с учетом низкоуровневых ограничений, и, кроме того, может стать причиной неоптимальных или небезопасных результатов, особенно в сценариях, требующих быстрой адаптации к непредвиденным препятствиям или внезапным изменениям окружающей среды. С другой стороны, интегрированное планирование задач и движений решает эту проблему, объединяя высокоуровневое планирование поведения с низкоуровневым планированием движений для получения выполнимых решений для задач с большим горизонтом, включающих геометрические ограничения и логические рассуждения. В последние годы интегрированному планированию задач и движений были посвящены обширные исследования [24—26].

Для планирования автономного маневра обгона в своем исследовании [8] авторы представили метод реактивного планирования, разделяющий задачу

на этапы планирования поведения и траектории, с использованием конечного автомата (FSM) на основе эвристических правил для планирования маневра, а затем планировщика траектории на основе управления с прогнозирующими моделями (MPC) для создания траекторий без столкновений. Однако использование простого FSM может оказаться недостаточным для сложных ситуаций. Разработка FSM для учета всех потенциальных возможностей сложной среды непрактична, а более простые FSM или множество правил могут либо пожертвовать производительностью из-за чрезмерной осторожности, либо поставить под угрозу безопасность из-за чрезмерного оптимизма.

В других исследованиях[45] задача принятия решения об обгоне была смоделирована как марковский процесс принятия решений (MDP), а обучение стратегии было основано на динамической нечеткой логике. Они рассмотрели сценарий, в котором беспилотный автомобиль окружают только четыре препятствия. Их MDP-модель состояла из девяти состояний: четырех относительных расстояний до четырех окружающих препятствий, четырех относительных скоростей четырех окружающих препятствий и идентификатора текущей полосы движения беспилотного автомобиля. Эксперименты моделировались в программе Simulation of Urban MObility (SUMO). Из-за вычислительных затрат, связанных с рассмотрением всех возможных действий, для упрощения вычислений был принят компромисс в количестве состояний.

В работе [53] планировался маневр обгона, включая решения и траектории в системе координации Frenet, путем построения семантического дерева решений с узлами, определяющими, каким транспортным средствам следовать, а какие — обгонять, генерируя несколько кандидатов на продольную и поперечную траекторию для каждого препятствия, которое предполагается обогнать. Затем из всех кандидатов траектории выбирается траектория с минимальной стоимостью. Предложенный подход обеспечивает хорошую производительность, но имеет ограничения по построению дерева, размеру и количеству сценариев, которые он может обрабатывать.

В статье [54] представлен интегрированный подход, комбинирующий принятие решений и планирование траектории для автономного изменения полосы движения и обгона, с акцентом на учет социального поведения окружающих автомобилей в рамках трех различных стилей вождения. Авторы используют модель игры Stackelberg для разработки алгоритма принятия

решений, который затем интегрируется с моделью потенциального поля, отражающей социальные характеристики транспортных средств, встроенной в модуль планирования движения. Для предсказания скорости и пути используется управление с прогнозирующими моделями (MPC). Подход представлен в виде итеративной оптимизационной задачи с замкнутым циклом и множеством ограничений.

В платформе автономного вождения с открытым исходным кодом Baidu Apollo [19], которая является известной платформой, поддерживающей разработку и тестирование алгоритмов планирования для самоуправляемых автомобилей, планировщик объединяет принятие решений с методами сглаживания пути и оптимизации скорости Piecewise-Jerk [51]. Он выбирает минимально затратный путь в системе координат Френе и оптимизирует его продольную скорость для траектории, свободной от столкновений. Однако он не полностью удовлетворяет потребность в пространственно-временных траекториях, необходимых для маневров обгона или смены полосы движения при столкновении с высокоскоростными динамическими препятствиями, поскольку приоритет отдается пространственному планированию траектории, а не временной корректировке профиля.

Для автоматической планировки маневра смены полосы движения авторы в своем исследовании [55] предложили основанный на правилах планировщик с использованием FMS, который управляет маневром смены полосы движения через пять состояний: переход на правую/левую полосу, возврат на левую/правую полосу и адаптивный круиз-контроль. При планировании траектории используется квадратичная программная оптимизация с управляющими функциями Ляпунова и управляющими барьерными функциями (CLF-CBF-QP, quadratic program optimization with control Lyapunov functions and control barrier functions). Проверенный в сценариях городских дорог и шоссе, предложенный метод достигает 62.46% успеха в городских условиях и 55.58% на шоссе. Однако в более сложных сценариях предложенный метод может столкнуться с проблемой, связанной с ростом числа правил/размеров.

В статье [7] для планирования автономного маневра смены полосы движения принятие решений было структурировано как частично наблюдаемый марковский процесс принятия решений (POMDP), включающий поведенческую модель, полученную из демонстрационных данных вождения.

Для оценки значений узлов действия использовалось моделирование Монте-Карло через дерево истории. Однако этот метод сталкивается с ограничениями, поскольку не соответствует требованиям реального времени, критичным для систем автономного вождения. Эти ограничения связаны с дорогостоящим обновлением убеждений и необходимостью реконструировать дерево истории для каждого шага планирования из-за непрерывного пространства наблюдений.

В другом исследовании [56] авторы представили схему планирования поведения при смене полосы движения с использованием подхода, основанного на выпуклой оптимизации. Сначала пространственно-временное изображение сценария движения (маневра) определяется путем вырезания полигонов на основе предсказаний траектории движения транспортных средств с препятствиями. Затем на этапе оптимизации траектории с помощью выпуклого квадратичного программирования интегрируются ограничения безопасности, в частности, время-до-столкновения и временной разрыв.

В других работах использовалось глубокое обучение с подкреплением (DRL) для автономных маневров по смене полосы движения. В исследовании [57] авторы использовали RL-агент, интегрированный с глубокой нейронной сетью, для принятия решений и планирования траектории в автоматизированной системе смены полосы движения. RL-агент обучен определять целевую полосу движения и принимать решение о том, оставаться ли на текущей полосе или переключиться на целевую полосу. Агент оптимизирует смену полосы движения с помощью глубокой нейронной сети, чтобы минимизировать общее время в пути. Унифицированная модель планирования траектории также генерирует опорную траекторию и профиль скорости при принятии решения о сохранении полосы движения или смене полосы.

В исследовании [58] авторы разработали структуру принятия решений и планирования траектории для выполнения маневров по смене полосы движения на шоссе, используя модуль принятия решений на основе глубокого обучения с подкреплением (DRL) для планирования поведения водителя. Затем они использовали управление с прогнозирующими моделями (MPC), которое выдает команды ускорения и рулевого управления для выполнения задач планирования продольного и поперечного движения. Для онлайн-эволюции модуля принятия решений DRL авторы использовали комбинацию модели

предсказания безопасного вождения и рациональной схемы. Кроме того, в статье [59] представлена стратегия автоматической смены полосы движения с использованием стратегии Proximal Policy Optimization (PPO) на основе глубокого обучения с подкреплением (DRL), в которой особое внимание уделяется безопасности, эффективности и комфорту. Подход использует высокоуровневую стратегию PPO для генерирования решений о смене полосы движения, определяя оптимальное время и исполнение на основе текущего контекста движения беспилотного автомобиля и его окружения, в то время как предопределенная модель обрабатывает управление нижнего уровня. Исследование показало, что беспилотный автомобиль, обученный PPO, может эффективно максимизировать накопленное вознаграждение и достичь 95%-99% успеха в плотном трафике, что подтверждает эффективность стратегии. Однако методы, основанные на DRL, могут быть недостаточно адаптированы к реальным дорожным сценариям, что указывает на необходимость дальнейших исследований для повышения их эффективности и обобщения.

Для планирования маневров автономной парковки для самоуправляемых автомобилей на платформе Apollo [19] был использован алгоритм Hybrid A*. Несмотря на то, что алгоритм хорошо справился с парковкой со статическими препятствиями, гибридный алгоритм A* не учитывает динамические препятствия, возникающие во время парковки, что может привести к столкновению или неполной траектории. Кроме того, из-за неинформативного поиска гибридный алгоритм A* генерирует небольшие фрагменты траектории с различными типами передач, что вносит накопленную ошибку траектории, и в результате автономный автомобиль может пропустить место парковки.

Другой подход к автоматической парковке был представлен в статье [60] на основе глубокого обучения с подкреплением (DRL), учитывающего только статические препятствия. Исследование освещает применение DRL в автоматизированной парковке, показывая, как различные модели и множество параметров влияют на производительность. Авторы подчеркивают, что, хотя DRL продемонстрировало успех в принятии решений в различных областях, включая симуляторы автономного вождения, при переносе этих алгоритмов в реальные сценарии остаются значительные проблемы. Несмотря на то, что статья дает ценное представление о потенциале и ограничениях DRL в автономном вождении, она не предлагает конкретных решений или достижений

для решения выявленных проблем, оставляя пробел в том, как эти улучшения могут быть реализованы в практических приложениях.

Глава 3. Адаптивное планирование маневров с помощью дерева поведения

Дерево поведения (BT, Behavior Tree) — один из основных подходов к принятию решений, традиционно используемый в искусственном интеллекте для игр, а в последнее время в робототехнике. BT представляют собой модульный процесс выполнения, который переключается между конечным множеством задач. Каждая задача может быть программой или подпрограммой, классифицируемой как условная задача или задача действия. Узлы потока управления определяют правила переключения между задачами. Эти узлы могут функционировать как последовательности, которые выполняют все свои дочерние задачи и возвращают успех только в случае успеха всех задач, или как узлы-селекторы (узлы обратного хода), которые выполняют свои дочерние задачи и возвращают успех в случае успеха любой из них.

Благодаря своей гибкости, реактивности и модульности деревья поведения (BT) являются высокоэффективными инструментами для планирования маневров автономных агентов. Они часто предпочтительнее конечных автоматов (FSM) при большом количестве переходов между состояниями. При увеличении сложности системы планирования размер дерева поведения растет пропорционально. Для решения этой проблемы было предложено несколько подходов [61—64], позволяющих частично или полностью изучать структуру дерева из окружающей среды, а не строить все дерево вручную. Однако построение дерева поведения с оптимальной адаптивной структурой все еще остается открытым вопросом. Подход, основанный на обучении с помощью эволюционного алгоритма, позволяет дереву поведения адаптироваться к специфическим потребностям планирования маневров беспилотного автомобиля.

В настоящей главе представлено адаптивное планирование маневров в беспилотных автомобилях с использованием обучаемого дерева поведения, которое развивается с помощью алгоритма генетического программирования. Представленный в этой главе подход был опубликован в статье [1].

Данная глава организована следующим образом. В разделе 3.1 представлена постановка задачи. Раздел 3.2 иллюстрирует предложенные методы, включая примитивы дерева поведения и генетическое

программирование. Далее, в разделе 3.3 представлены и проанализированы экспериментальные результаты, а в разделе 3.4 дано заключение.

3.1 Постановка задачи

Входные данные включают в себя конфигурацию беспилотного автомобиля $q(t) = [x, y, \theta, v, a]$, представляющую положение, угол поворота, скорость и ускорение автомобиля, а также состояния m динамических препятствий $B_i : i \in \{1, \ldots, m\}$, которые описываются их положениями, углами поворота и скоростями $[x_{bi}, y_{bi}, \theta_{bi}, v_{bi}]$. Кроме того, входные данные включают в себя информацию карты $\mathcal{M} \in M$, такую как опорные линии и полосы движения. Время T дискретизируется на Δt. Возможные действия определяются множеством $A = \{a_1, ..., a_n\}$, где $n = 3$, а действия включают в себя "KeepLane"для сохранения текущей полосы движения, "SwitchToRight"для перестроения на правую полосу движения и "SwitchToLeft"для перестроения на левую полосу движения.

Учитывая начальную конфигурацию q_0 и целевую область G, задача состоит в том, чтобы найти выполнимый план $\pi = [a_0, a_1, \ldots, a_T]$, используя основанный на правилах планировщик (дерево поведения) $D : Q \times B \times M \to A$. Следуя плану π, цель G должна быть достигнута из начальной конфигурации q_0, образуя траекторию $\tau = (q_0, q_1, \ldots, q_g)$. Время выполнения плана должно быть минимизировано, в частности, для выполнения сценариев обгона, при этом траектория $\tau(t)$ не должна приводить к столкновению с препятствием, $\tau(t) \cap \forall b_i(t) \in B = \varnothing$.

Задача адаптивного планирования маневров решается как интегрированная задача принятия решений и планирования движения и рассматривается как задача оптимизации. Алгоритм генетического программирования (GP, Genetic Programming) используется для поиска оптимальной структуры дерева поведения для принятия решений об обгоне и планирования движения с помощью оптимизатора Jerk. Структура обозначается символьной строкой x. Задача рассматривается как оптимизационная, при этом алгоритм генетического программирования ищет в пространстве генотипов G строку $x*$, обозначающую программу дерева

поведения, для которой оптимизируется функция приспособленности f:

$$f(x) \to max : find \overrightarrow{x}^* so\ that\ \forall \overrightarrow{x} \in G : f(\overrightarrow{x}) \leqslant f(\overrightarrow{x}^*) = f^* \qquad (3.1)$$

Язык программирования $L = \{F,T\}$ GP включает в себя два множества: множество функций F и множество терминалов T. F включает в себя все возможные функции в программе BT (узлы потока управления). В T входят все возможные условия и действия.

В GP пространство генотипов G включает строки, представляющие все программы BT, которые могут быть построены из элементов языка программирования L.

$P(t) \subset G$ обозначает состояние популяции в момент времени t. Время работы ГП определяется в терминах поколений. Популяцию составляют N особей $\overrightarrow{x_1}(t)$,$\overrightarrow{x_2}(t)$,...,$\overrightarrow{x_n}(t)$. Популяция GP развивается из начального состояния $P(t = 0)$ путем выбора μ родителей ($P_p(t)$) для варьирования и последующего выполнения *операции варьирования* с ними для получения λ-потомства от каждого родителя. Множество потомков (offspring) обозначается $P_c(t)$. Операция изменения может быть либо кроссовером, либо мутацией. После оценки пригодности нового потомства оно объединяется с популяцией и образует новое множество $P_a(t)$. Новое поколение $P(t + 1)$ является подмножеством $P_a(t)$. На протяжении поколений алгоритм GP ищет значения x, которые максимизируют функцию приспособленности f.

Для проведения экспериментов мы используем платформу автономного вождения Baidu Apollo (с открытым исходным кодом)[19]. Платформа состоит из нескольких подсистем, которые предоставляют подсистеме планирования необходимую информацию о маршрутизации, локализации, восприятии и предсказании окружающих объектов. Благодаря своей структуре, а также высокому параллелизму и низкой задержке, платформа представляет собой точный симулятор, подходящий для разработки и тестирования алгоритмов планирования для беспилотных автомобилей.

Процесс принятия решений о маневрах обгона на платформе Apollo интегрирован с планированием движения в подсистеме планирования. Подсистема планирования получает на вход: карту HD, местоположение и скорость беспилотного автомобиля, окружающие объекты, их предполагаемую 3D-форму, скорость, предсказание их будущей траектории и курса. При наличии медленно движущегося препятствия перед беспилотным автомобилем

в течение длительного периода времени в число кандидатов включаются пути из соседних полос, чтобы при необходимости можно было сменить полосу движения. Затем для каждой полосы-кандидата используется оптимизатор Piecewise-jerk [65] для оптимизации пути, после чего оптимизируется профиль скорости оптимизированного пути. Наконец, для определения траектории используется функция стоимости. При появлении динамических препятствий скоростной подход не гарантирует оптимального решения. В ответ на это ограничение планировщик EM разработан таким образом, чтобы выбирать маневр смены полосы движения, а не обгон для скоростных динамических препятствий.

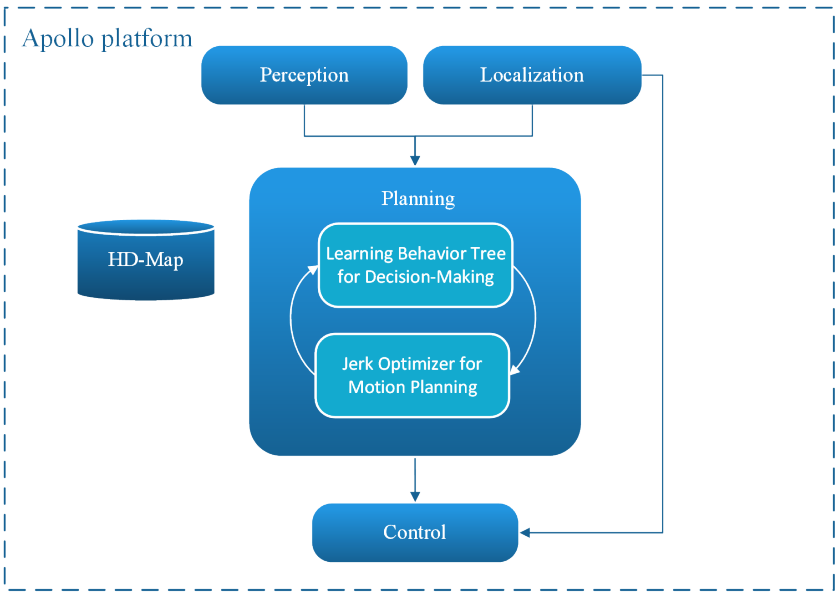


Рисунок 3.1 — Схема предлагаемого подхода для адаптивного планирования маневров. Планирование объединяет принятие решений с помощью обучающегося дерева поведения с планированием движения с помощью оптимизатора рывков. Структура дерева поведения адаптируется к условиям автономного вождения.

Мы интегрировали наш метод планирования обгона, а также технический вклад, позволяющий обгонять высокоскоростные динамические препятствия, в платформу Apollo (рисунок 3.1). Модуль планирования имеет обучаемое дерево поведения для принятия решений, интегрированное с планировщиком движения оптимизации Jerk. Структура дерева адаптируется к окружающей среде путем обновления и обучения оптимальной структуре для планирования обгона.

3.2 Методы

Структура дерева поведения маневра обгона изучается с помощью генетического программирования. ВТ определяет, будет ли беспилотный автомобиль обгонять движущийся впереди автомобиль (динамическое препятствие) с правой или левой стороны или же он сохранит текущую полосу движения. Сценарий включает несколько переменных: скорость переднего препятствия, количество динамических препятствий на правой и левой полосах, их скорость и начальное местоположение.

В платформе Apollo было разработано техническое решение для обгона динамических препятствий, которое было реализовано с помощью генетического программирования для оценки особей (структур дерева поведения) в популяции.

3.2.1 Примитивы дерева поведения

Фенотип — это программная версия генотипа. Чтобы оценить приспособленность особи ВТ, необходимо отображение между генотипом и фенотипом. Множество всех узлов действия, условия и управления ВТ определяет фенотипические примитивы. *KeepLane*, *SwitchToLeft* и *SwitchToRight* — это возможные действия. Управляющий узел может быть последовательностью или селекторным узлом. Каждая соседняя полоса разбита на три зоны с различной длиной s для отражения состояния занятости (рисунок 3.2). Расстояние $d(t)$ указывает на безопасное расстояние следования и зависит от текущей скорости беспилотного автомобиля. Для его определения используется «правило трех секунд»:

$$d(t) = \tau * v(t) \tag{3.2}$$

, где $\tau = 3\ s$. Для каждой зоны существует два состояния: $\{free, occupied\}$. Кроме того, скорость препятствия (km/h) рассматривается как диапазон и выражается в шести состояниях: [1,10], [11,20], [21,30], [31,40], [41,50], [51,60]. Таблица 1 содержит весь список условий, действий и описаний.

Таблица 1 — Примитивы дерева поведения. Возможные действия и состояния, их описание и алфавит, представляющий их.

Alphabet	Actions	Description
Y	SwitchToLeft	Switch from the current lane to the left lane.
Z	SwitchToRight	Switch from the current lane to the right lane.
X	KeepLane	Keep going on the current lane.
Alphabet	**Conditions**	**Description**
c	L1Free	Is Zone L1 free?
d	L1Occ	Is Zone L1 occupied?
e	L2Free	Is Zone L2 free?
f	L2Occ	Is Zone L2 occupied?
g	L3Free	Is Zone L3 free?
h	L3Occ	Is Zone L3 occupied?
i	R1Free	Is Zone R1 free?
j	R1Occ	Is Zone R1 occupied?
k	R2Free	Is Zone R2 free?
l	R2Occ	Is Zone R2 occupied?
m	R3Free	Is Zone R3 free?
n	R3Occ	Is Zone R3 occupied?
o	ObsSpeed1-10	Is occupant obstacle's speed in range [1,10]km/h ?
p	ObsSpeed11-20	Is occupant obstacle's speed in range [11,20]km/h?
q	ObsSpeed21-30	Is occupant obstacle's speed in range [21,30]km/h?
r	ObsSpeed31-40	Is occupant obstacle's speed in range [31,40]km/h?
s	ObsSpeed41-50	Is occupant obstacle's speed in range [41,50]km/h?
t	ObsSpeedMore50	Is occupant obstacle's speed more than 50 km/h?
u	EgoSpeed1-10	Is ego vehicle's speed in range [1,10] km/h?
v	EgoSpeed11-20	Is ego vehicle's speed in range [11,20]km/h?
w	EgoSpeed21-30	Is ego vehicle's speed in range [21,30]km/h?
x	EgoSpeed31-40	Is ego vehicle's speed in range [31,40]km/h?
y	EgoSpeed41-50	Is ego vehicle's speed in range [41,50]km/h?
z	EgoSpeedMore50	Is ego vehicle's speed more than 50 km/h?

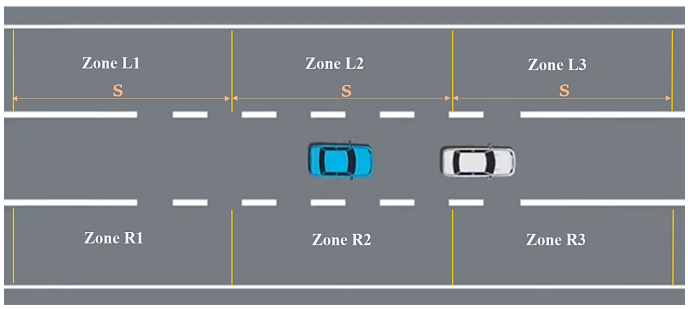


Рисунок 3.2 — Зоны двух соседних полос. Левая полоса разделена на три зоны L1, L2 и L3. Правая полоса также разделена на зоны R1, R2 и R3. Каждая зона имеет переменную длину s.

Состояние BT может быть одним из трех: SUCCESS, FAIL или RUNNING. Когда требуется выполнить действие, активируется состояние RUNNING. Если дерево находится в состоянии RUNNING в течение длительного периода времени без выполнения действия, состояние считается FAIL. Добавление состояния RUNNING к характеристикам дерева не только исключает деревья, которые приводят к неприменимым действиям, но и сохраняет реактивность дерева при изменении условий окружающей среды во время выполнения действия.

Символы, представляющие действия и условия (таблица 1), образуют терминальное множество $T = \{c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z, X,Y,Z\}$, которое отображает программу дерева поведения (фенотип) на алфавит генотипа. Узел последовательности представлен символом '&' с двумя круглыми скобками, содержащими поддеревья под узлом последовательности. Аналогично узел селектора представлен символом '/' с двумя круглыми скобками. Множество функций F — это множество двух управляющих функций:$\{\&, /\}$. Например, дерево поведения при обгоне, которое принимает решение об обгоне только тогда, когда одна из соседних полос свободна от препятствий (рисунок 3.3), представлено следующей строкой: '/(&(cegY)&(ikmZ)X)'.

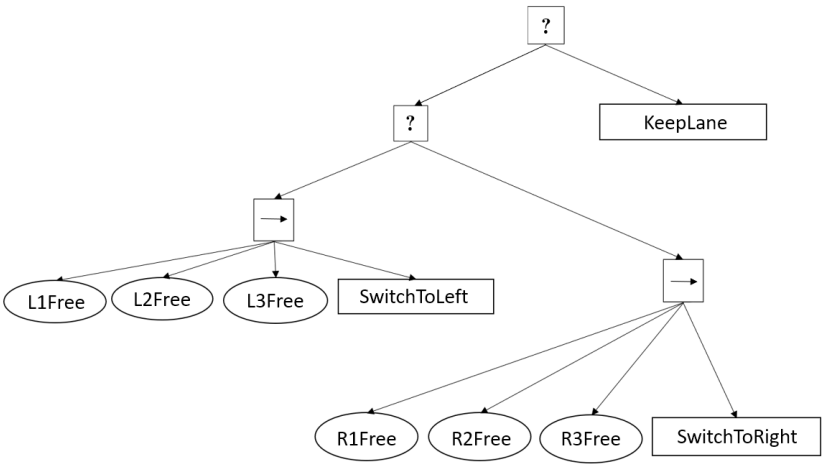


Рисунок 3.3 — Простое дерево поведения для маневра обгона. Дерево начинается с узла-селектора, который проверяет состояние SUCCESS своих дочерних узлов. Это заставляет второй узел-селектор проверить своих дочерних узлов. Первый дочерний узел (узел последовательности) оценивает три условия состояния зон левой полосы; если три свободные зоны L1, L2, L3 существуют, принимается решение о переключении на левую полосу, и дерево возвращает состояние RUNNING. Если одно из трех условий не выполняется, узел последовательности возвращает состояние FAIL, и селекторный узел продолжает проверку второго дочернего узла; если он также возвращает состояние FAIL, дерево окончательно решает оставить полосу.

3.2.2 Генетическое программирование

Для поиска оптимальной структуры дерева поведения мы используем схему выполнения GP, аналогичную той, что описана в [63]. Однако для задачи оптимизации планирования мы выбрали более подходящие параметры (таблица 2). Для выбора μ родителей для вариации используется метод выбора бинарного турнира. В этом методе выбираются две случайные особи, из которых выбирается та, у которой большее значение приспособленности, и добавляется в множество родителей. Турнирный отбор позволяет особям с низкими показателями сохранять свой генотип, что приводит к лучшему исследованию пространства генотипов и снижению вероятности застревания в локальном максимуме. Потомство λ создается путем выполнения операции кроссовера или мутации над выбранными родителями с определенной вероятностью (алгоритм 1).

Algorithm 1 Genetic Programming to learn optimal behavior tree's structure for overtaking planning

$INPUT : N, MaxGenerations, e(ElitesNumber),$
$Crossover - Percent.$
$OUTPUT : \overrightarrow{x}^{*} so\ that\ \forall \overrightarrow{x} \in G : f(\overrightarrow{x}) \leqslant f(\overrightarrow{x}^{*}) = f^{*}$
$t = 0$
$initialize : P(t = 0) = \{\overrightarrow{x_1}(0),\overrightarrow{x_2}(0),...,\overrightarrow{x_n}(0)\} \in G$
while $t < MaxGenerations$ **do**
  $evaluate : P(t) : \{f(\overrightarrow{x_1}(t)),f(\overrightarrow{x_2}(t),...,f(\overrightarrow{x_n}(t))\}$
  $P_p(t) = TournamentSelection(\mu,P(t))$
  $P_c(t) = \varnothing$
  for $i = 1,2,...,\mu$ **do**
    $rand = RandNumber(1 - 100)$
    $P_1 = P_p^i(t)$
    if $rand \leqslant Crossover - Percent$ **then**
      for $k = 1,...,\lambda/2$ **do**
        $P_2 = RandSelect(P_p(t))$
        $C_1,C_2 = CrossoverOperation(P_1,P_2)$
        $P_c(t) = P_c(t) \cup C_1 \cup C_2$
      end for
    else
      for $k = 1,...,\lambda$ **do**
        $C = MutationOperation(P_1)$
        $P_c(t) = P_c(t) \cup C$
      end for
    end if
  end for
  $simulate\&evaluate : P_c(t) : \{f(\overrightarrow{x_{c1}}(t)),f(\overrightarrow{x_{c2}}(t),...,f(\overrightarrow{x_{cn}}(t))\}$
  $P_a(t) = P_c(t) \cup P(t)$
  $P(t+1) = ElitistSelection(e,P_a(t))$
  $P(t+1) = P(t+1) \cup TournamentSelect(N - e,P_a(t))$
  $t \leftarrow t + 1$
end while

Таблица 2 — Параметры генетического программирования.

Parameter	Value
Population size (N)	20
Parents number(μ)	10
Parents selection method	Binary Tournament
Offspring number(λ)	4
Max depth	6
Crossover percent	40%
Mutation percent	60%
Mutation/ addition percent	40%
Mutation/ deletion percent	30%
Mutation/ mutation percent	30%
Elitist selection percent	10%

Операции по изменению выглядят следующим образом:

Кроссовер (Crossover) Эта операция берет две особи P_1, P_2 из выбранных родителей $P_p(t)$ и выполняет обмен поддеревьев для создания двух новых потомков C_1, C_2.

Мутация (Mutation) Эта операция позволяет индивидууму изменяться одним из трех способов с вероятностью: добавление, мутация или удаление. При добавлении в дерево добавляется случайный элемент из языка программирования L. При мутации случайный элемент из L мутирует в другой элемент из L. При удалении из дерева удаляется конечный элемент (условие или действие).

Процент мутации установлен на уровне 60%, чтобы улучшить раскрытие пространства поиска и избежать быстрого роста глубины дерева, вызванного операцией кроссовера. 70% операций мутации — это мутация или сложение, чтобы изучить все возможные действия и условия при поиске оптимального решения.

После вариации все потомки offspring ($P_c(t)$) оцениваются и объединяются с популяцией $P(t)$, образуя $P_a(t)$. После этого из $P_a(t)$ выбираются e особей с наивысшими значениями приспособленности для выживания в следующем поколении (Elitist selection), а остальное поколение ($N - e$) выбирается с помощью бинарного турнирного отбора без повторения какой-либо особи.

Все ранее выжившие особи оцениваются снова по новому сценарию на следующей итерации GP. Следующее уравнение объединяет предыдущее значение приспособленности, связанное с выжившими особями, с новым значением приспособленности, рассчитанным по сценарию новой итерации:

$$f(\overrightarrow{x}(t)) = \alpha * f(\overrightarrow{x}(t-1)) + (1-\alpha) * f(\overrightarrow{x}(t)) \tag{3.3}$$

где $\alpha = 1/2$. Таким образом, приспособленность программы BT будет определяться как интеграция ее приспособленности по сценариям.

Приспособленность каждого индивидуума (BT) в сценарии определяется двумя факторами: достижение цели без столкновений и время, необходимое для достижения цели по данной программе BT. Если индивид (BT) вызвал столкновение, ему присваивается отрицательное значение приспособленности. Если цель достигнута без столкновений, то приспособленность определяется как разница между временем, затраченным на выполнение действия *KeepLane* T^{a_0}, и временем, затраченным на выполнение выходного действия дерева T^{a_x}:

$$f(x) = T^{a_0} - T^{a_x} \tag{3.4}$$

В результате BT, который приводит к обгону и более быстрому достижению цели без столкновений, будет иметь высокое значение приспособленности. BT, который приводит к действию *KeepLane*, будет иметь значение, близкое к нулю.

3.2.3 Реализация на платформе Apollo

Модуль планирования Apollo включает в себя два основных планировщика: *Планировщик дорог общего пользования*, который обрабатывает сценарии движения по полосам, перекрестков и U-образных поворотов, и *Планировщик открытых пространств*, который обрабатывает сценарии парковки. Каждый планировщик инициализирует *менеджер сценариев* и обновляет его каждый цикл планирования. Менеджер сценариев определяет текущий сценарий, обрабатывает его и выполняет ряд предопределенных

задач для каждого сценария. Задачи могут быть либо решающими, либо оптимизирующими.

Чтобы продемонстрировать процесс планирования в Apollo, мы рассмотрим планирование сценария движения по полосе, в котором планировщик может принять решение об обгоне. Процесс планирования начинается с *Lane Change Decider*, который определяет, есть ли на дороге дополнительная полоса, доступная для смены в случае необходимости. Затем *Path Lane Borrow Decider* проверяет наличие переднего статического долгосрочного препятствия, принимая решение об обгоне, если оно есть. Затем с помощью оптимизатора пути Jerk находится оптимальный профиль пути в Frenet, а с помощью оптимизатора скорости Jerk — оптимальный профиль скорости в Frenet.

Path Lane Borrow Decider в Apollo предназначен для принятия решений об обгоне исключительно статичных передних препятствий и препятствий с очень низкой скоростью. Предлагаемый метод дерева поведения реализован в решателе BT, который добавляется к задачам сценария Lane-follow в Apollo перед *Path Lane Borrow Decider*. Все примитивные условия и действия BT запрограммированы в дешифраторе в виде функций. Дешифратор начинает с декодирования строки дерева, сгенерированной GP, из пространства генотипов в пространство фенотипов. Затем дешифратор запускает рекурсивную функцию выполнения дерева для выполнения задач дерева поведения на основе узлов потока управления. Для оценки дерева поведения также были разработаны дополнительная функция проверки столкновений и функция достижения цели.

3.3 Оценка и результаты

В этом разделе демонстрируются экспериментальные результаты работы алгоритма адаптивного планирования маневров с использованием дерева поведения. Предложенный алгоритм был оценен для планирования адаптивных маневров при обгоне путем проведения серии экспериментов в платформе моделирования автономного вождения Apollo. В подразделе 3.3.1 представлены среда моделирования и множество настроек эксперимента.

В подразделе 3.3.2 демонстрируются и анализируются результаты работы предложенного алгоритма на поколениях. В подразделе 3.3.3 демонстрируются и анализируются результаты работы предложенного алгоритма, дополненные предварительными знаниями.

3.3.1 Среда и сценарии симуляции

Экспериментальные сценарии обгона, использованные для оценки структуры каждого дерева поведения, проводились на платформе Apollo для автономного вождения. Исходным кодом Apollo является шестая версия открытой платформы Baidu Apollo. Исходный код был изменен, а в задания был добавлен решатель дерева поведения, что позволило протестировать маневр обгона, эффективность дерева поведения и его оценку на каждом цикле планирования.

Сценарий обгона включает в себя три полосы движения на скоростном шоссе, причем беспилотный автомобиль всегда находится на средней полосе (рисунок 3.4). Спереди имеется препятствие с начальным расстоянием 20 m. Скорость переднего препятствия задается случайным образом в диапазоне от 2.7 m/s до 5.5 m/s. Также существует пять препятствий, расположенных случайным образом на соседних полосах. Соседние препятствия движутся со случайными скоростями (от 2.7 m/s до 6.5 m/s). Эксперименты проводятся в режиме Apollo SimControl, который предполагает идеальное управление беспилотным автомобилем, следующим по запланированной траектории без ошибок. Данные о препятствиях публикуются с частотой 10 Hz по каналу Perception. Начиная с начального местоположения автономного агента расстояние до цели составляет 200 m.

3.3.2 Дерево адаптивного поведения без предварительных знаний

Мы проводим эксперименты предложенного алгоритма для первых 50 поколений начиная со случайной первой популяции $P(0)$.

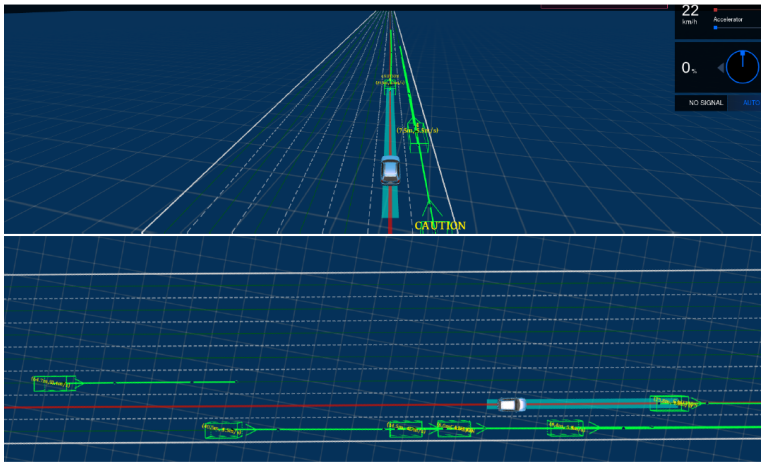


Рисунок 3.4 — Случайный сценарий обгона. Беспилотный автомобиль окружен пятью случайными динамическими препятствиями и одним — спереди.

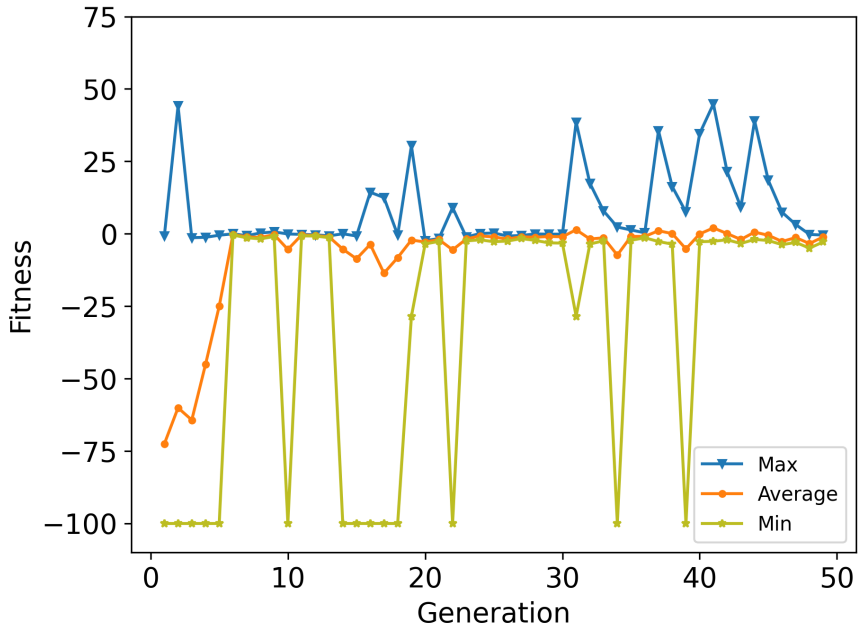


Рисунок 3.5 — Эволюция приспособленности за 50 поколений с помощью алгоритма адаптивного планирования маневров с использованием поведенческого дерева, показывающего максимальное (синий), среднее (оранжевый) и минимальное (зеленый) значения приспособленности. На графике видны первые улучшения, за которыми следуют периоды колебаний максимального значения приспособленности в процессе исследования пространства поиска оптимального решения, а также стабилизация среднего и минимального значений приспособленности, что свидетельствует о значительной конвергенции популяции в течение поколений.

Эволюция максимального, минимального и среднего значений функции приспособленности по поколениям показана на рисунке 3.5. Максимальная функция приспособленности демонстрирует немедленный всплеск в первых поколениях, достигая значения более 50, которое приобретается, когда дерево поведения приводит к выполнению маневра обгона. В последующих поколениях максимальная приспособленность колеблется из-за мутаций или кроссовера. В более поздних поколениях (около 30–50) максимальная приспособленность демонстрирует серию пиков и впадин, указывая на то, что алгоритм продолжает исследовать пространство поиска постоянно лучших решений или оптимального дерева поведения для маневра обгона в различных сценариях вождения.

Минимальная приспособленность имеет начальное падение с очень низкой отметки -100, как и на предыдущем графике, что указывает на то, что начальная популяция содержит очень плохие решения, которые генерируются случайным образом и вероятно приведут к авариям на дороге. Несмотря на периодические падения, минимальная приспособленность имеет тенденцию стабилизироваться немного ниже нуля после первых поколений, что является значением, полученным, когда дерево привело к принятию решения о сохранении текущей полосы движения, что говорит о том, что большая часть популяции работает относительно хорошо, обеспечивая безопасное вождение, даже если некоторые плохие особи сохраняются или периодически появляются вновь.

В целом стабилизация среднего и минимального значений приспособленности говорит о том, что популяция в основном сходится, а продолжающиеся колебания максимальной приспособленности означают, что предложенный алгоритм все еще исследует пространство поиска оптимального решения.

Чтобы более подробно рассмотреть результаты работы предложенного алгоритма, начиная со случайной первой популяции, мы анализируем эволюцию успешных деревьев, приводящих к безопасному поведению водителя, включая деревья, которые привели к безопасному маневру обгона (рисунок 3.6). Дерево поведения может быть либо успешным, либо неуспешным. Дерево считается успешным, если оно достигает своей цели без столкновения с каким-либо из окружающих препятствий, независимо от типа выходного действия (KeepLane, SwitchToLeft, SwitchToRight). $S(t) \subseteq P(t)$ обозначает множество успешных BT, $O(t) \subseteq S(t)$ обозначает множество успешных BT

с выходным действием обгона. Дерево считается неуспешным в следующих случаях: выходное действие ВТ привело к столкновению, время выполнения действия превысило заданный максимальный предел, а действие еще не выполнено, или выполнение ВТ завершилось, а действие не выполняется. $U(t) \subseteq P(t)$ обозначает множество неуспешных ВТ, а $P(t) = S(t) \cup U(t)$.

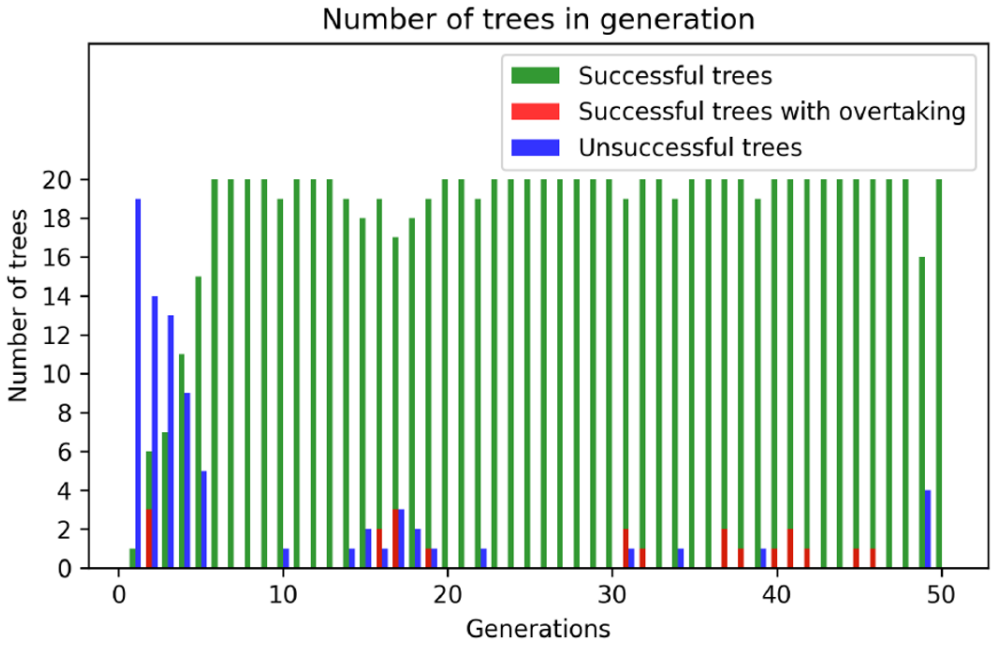


Рисунок 3.6 — Гистограмма успешных и неуспешных деревьев по поколениям, начиная с первого поколения и заканчивая 50-м. Начальная популяция — случайно сгенерированная популяция.

Результаты, представленные на рисунке 3.6, показывают, что количество успешных деревьев увеличивается с течением поколений, в то время как количество неуспешных деревьев значительно уменьшается. Видно, что эволюционирующая популяция, исследуя пространство поиска, нашла свой путь к созданию деревьев поведения с действием обгона, которое было успешным в некоторых сценариях. Начиная со случайной популяции задача поиска может занять больше времени, но такой алгоритм никогда не должен заканчиваться в локальном максимуме, поскольку функция мутации и метод отбора расширяют процесс поиска.

Выполнение 50 поколений алгоритма GP заняло почти 48 часов, что считается очень трудоемким. Время, необходимое для создания потомства и декодирования строки ВТ, относительно невелико, и им можно пренебречь по сравнению с временем, необходимым для каждого эксперимента по оценке

особи BT. Каждый эксперимент по оценке может длиться от 10 до 70 секунд, в зависимости от действия и валидности вывода дерева. Это можно объяснить тем, что каждый сценарий вождения выполняется в реальном времени на платформе Apollo, чтобы сделать алгоритм надежным и применимым к реальным задачам вождения.

3.3.3 Дерево адаптивного поведения с предварительными знаниями

Чтобы ускорить эволюцию структуры дерева поведения для адаптивного планирования обгона в поколениях, мы запустили предложенный алгоритм с начальной популяцией, содержащей 19 случайно сгенерированных особей и одно дерево эффективного поведения простой структуры (рисунок 3.3) для планирования маневра обгона из левой полосы, когда три зоны на левой полосе свободны от препятствий, или из правой полосы, когда три зоны на правой полосе свободны от препятствий, или для сохранения текущей полосы в случае любых других условий.

Эволюция максимального, минимального и среднего значений функции приспособленности в поколениях показана на рисунке 3.7. Максимальное значение функции приспособленности в целом увеличивается с течением поколений. В поколениях (5–10) он остается близким к нулю (деревья с единственным действием keep-lane), затем, исследуя пространство поиска с помощью мутации и кроссовера, он находит особей с более высокими значениями приспособленности и продолжает увеличиваться с колебаниями. Эти колебания происходят по мере того, как алгоритм ищет лучшие решения, которые являются последовательными и сохраняют высокую приспособленность при различных сценариях обгона. В первых нескольких поколениях средняя приспособленность постоянно улучшается, показывая, что популяция в целом становится лучше. Стабилизация средней приспособленности указывает на то, что популяция сходится.

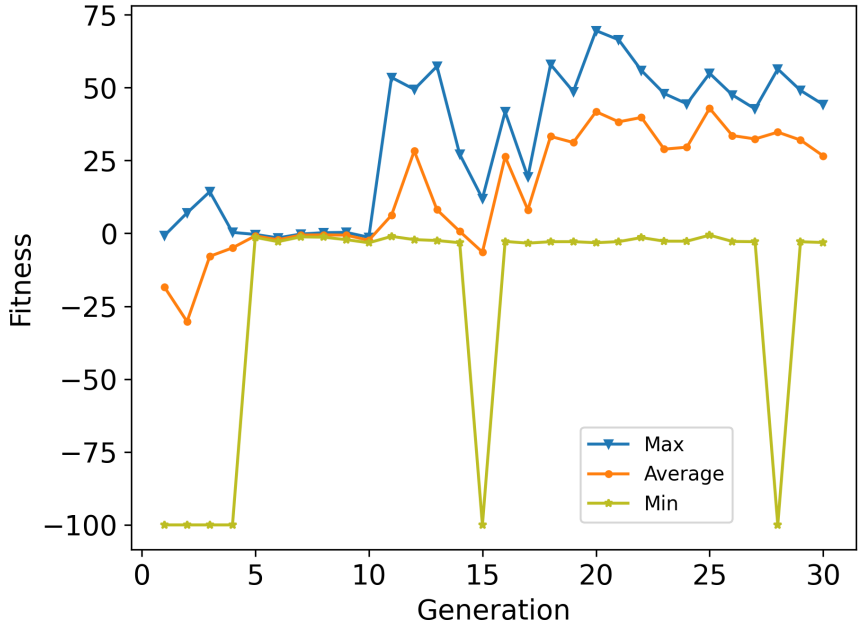


Рисунок 3.7 — Эволюция приспособленности за 30 поколений с помощью алгоритма Adaptive Maneuver Planning using Behavior Tree, показывающего максимальное (синий), среднее (оранжевый) и минимальное (зеленый) значения приспособленности. На графике видно увеличение минимального, среднего и максимального значений приспособленности с течением поколений, что свидетельствует о сближении популяции с оптимальной приспособленностью в процессе исследования пространства поиска оптимального решения.

Минимальная приспособленность в первых поколениях низка и достигает -100, что указывает на наличие плохого поведения отдельных особей, которое приводит к авариям на дороге, поскольку большая часть популяции была сгенерирована случайным образом. Однако после нескольких первых поколений минимальная приспособленность резко возрастает и достигает значения около нуля, что означает, что худшие особи дерева поведения приводят к соблюдению полосы движения, что также является безопасным поведением водителя. Периодические падения до очень низких значений минимальной приспособленности связаны с появлением новых, случайных особей путем мутации.

В целом стабилизация средних и минимальных значений приспособленности и увеличение максимальной приспособленности говорит о том, что популяция сходится к оптимальному решению и продолжает

исследовать пространство поиска оптимальной структуры дерева поведения, планируя безопасный маневр обгона в различных сценариях вождения.

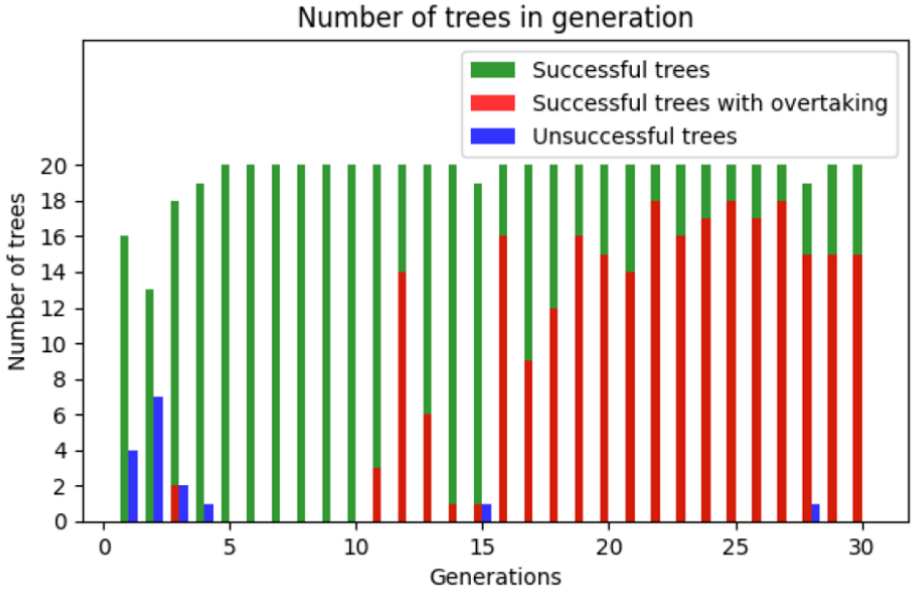


Рисунок 3.8 — Гистограмма количества успешных и неуспешных деревьев в поколениях от первого поколения до 30-го. Начальная популяция содержит одно простое эффективное дерево поведения — дерево поведения, планирующее безопасный маневр обгона на различных сценариях движения.

На рисунке 3.8 показана эволюция успешных деревьев, приводящих к безопасному поведению водителя, включая деревья, которые привели к безопасному маневру обгона за 30 поколений. Результаты показывают, что алгоритм адаптивного дерева поведения динамически сходится к оптимальному значению функции приспособленности. Алгоритм оптимизирует функцию приспособленности на протяжении поколений, развивая деревья, которые приводят к действию обгона, чтобы быстрее достичь цели, а также сохраняет разумный размер дерева благодаря ограничению на максимальную глубину дерева поведения.

Множество особей BT с длиной l больше 40 символов (включая скобки) обозначается $L(t) \subseteq P(t)$. В таблице 3 представлено процентное сравнение успешных деревьев, обгоняющих успешных деревьев и длины дерева в поколениях, когда поиск алгоритма начинается со случайного и с предварительного.

Таблица 3 — Сравнение алгоритма адаптивного дерева поведения с предварительным и случайным первым поколением.

	GP with prior			**GP with random**		
t	$S(t)\%$	$O(t)\%$	$L(t)\%$	$S(t)\%$	$O(t)\%$	$L(t)\%$
1	80	0	0	5	0	5
5	100	0	5	75	0	20
10	100	0	15	95	0	60
15	95	5	40	95	0	15
20	100	75	35	100	0	60
25	100	90	75	100	0	60
30	100	75	95	100	0	50

Было замечено, что в старших поколениях успешные деревья развили некоторые заметные паттерны, такие как ассоциация условий, относящихся к занятой зоне, с условиями, определяющими скорость препятствия в узле управления последовательностью, например, символы «jo», представляющие условия: «R1Occ» и «ObsSpeed1-10». Еще одно замечание — появление новых ветвей в успешных деревьях, представляющих сценарии, в которых одна или несколько зон заняты препятствием.

3.4 Заключение

В настоящей главе рассматривается задача интегрированного принятия решений и планирования движения для автономного маневра, а также представлен подход к адаптивному планированию маневра с использованием обучающегося дерева поведения. Были продемонстрированы и проанализированы экспериментальные результаты. Запуск предложенного алгоритма только для первых поколений алгоритма обучения GP дал многообещающие результаты для адаптивного планирования маневров благодаря его высокой гибкости, модульности и потенциалу. Несмотря на то, что время, необходимое для выполнения каждого эксперимента на платформе Apollo, является проблемой, мы можем преодолеть ее

в будущем, запустив несколько экземпляров Apollo параллельно для одновременной оценки многих структур деревьев поведения. Это можно сделать с помощью высокопроизводительного оборудования, например, суперкомпьютеров. Такое решение позволит значительно сократить время поиска оптимальной структуры BT. Еще одним улучшением алгоритма может стать использование более эффективного эволюционного алгоритма, расширение условий окружающей среды и включение информации из модуля предсказаний Apollo.

Глава 4. Обучение с подкреплением для адаптивного планирования маневров

По мере развития технологий автономного вождения такие специализированные задачи, как парковка, представляют собой уникальные проблемы, требующие точного и адаптивного принятия решений. Парковка включает в себя навигацию по сложным средам, таким как автостоянки, предполагающим статические и динамические препятствия, ограниченное пространство и непредсказуемую активность людей. Для преодоления этих проблем обучение с подкреплением (RL) предлагает действенное решение, позволяя автомобилю обучаться оптимальным маневрам парковки через взаимодействие с окружающей средой. Методы, основанные на RL, позволяют беспилотному автомобилю динамически адаптировать свои стратегии парковки, повышая безопасность и эффективность.

Для планирования маневра перпендикулярной парковки был использован алгоритм POLAMP [66] в качестве локального планировщика, который был обучен избегать динамических препятствий. Он состоит из алгоритма оптимизации ближайшей стратегии (PPO, Proximal Policy Optimization) [67] и стека фреймов. Модель велосипеда была интегрирована в симулятор POLAMP, а ограничения на ускорение использовались для того, чтобы автономный автомобиль двигался более реалистично и приближенно к динамике автомобиля Apollo.

В настоящей главе рассматривается применение обучения с подкреплением к задаче адаптивного планирования маневров парковки. Представленный подход был опубликован в статье [2]. Глава построена следующим образом. В разделе 4.1 представлена задача парковки как задача планирования маневров для беспилотных автомобилей. Она формулируется в терминах обучения с подкреплением путем определения соответствующего пространства состояний, пространства действий и функций вознаграждения. В разделе 4.2 представлена модель автомобиля, используемая в рамках данного подхода. В разделе 4.3 показано, как применяется обучение по расписанию (curriculum learning) для постепенного обучения RL-агента решению все более сложных задач, связанных с парковкой. Раздел 4.4 описывает интеграцию модели обучения с подкреплением для парковки в платформу автономного вождения Apollo.

В разделе 4.5 продемонстрировано, как эффективность подхода к парковке на основе RL оценивается с помощью симуляций. Наконец, в разделе 4.6 приводятся выводы.

4.1 Постановка задачи

Входные данные включают в себя конфигурацию беспилотного автомобиля $q(t) = [x, y, \theta, v, a]$, представляющую положение, угол поворота, скорость и ускорение автомобиля, а также состояния m динамических препятствий $B_i : i \in \{1, \ldots, m\}$, которые описываются их положениями, углами поворота и скоростями $[x_{bi}, y_{bi}, \theta_{bi}, v_{bi}]$. Кроме того, входные данные включают в себя информацию карты $\mathcal{M} \in M$, такую как геометрия парковочного места, границы полос. Время T дискретизируется на Δt.

Учитывая начальную конфигурацию q_0 и целевую конфигурацию q_g на парковочном месте, задача состоит в том, чтобы найти выполнимый план $\pi = [a_0, a_1, \ldots, a_T]$, используя планировщик на основе обучения с подкреплением (RL) $D : Q \times B \times M \to A$. Следуя плану π, цель q_g должна быть достигнута из начальной конфигурации q_0, образуя траекторию $\tau = (q_0, q_1, \ldots, q_g)$, при этом траектория $\tau(t)$ не должна приводить к столкновению с препятствием, $\tau(t) \cap \forall b_i(t) \in B = \varnothing$. Траектория движения для парковки делится на две части: одна — вперед, другая — задним ходом.

Мы рассматриваем частично наблюдаемый марковский процесс принятия решений (S, A, O, T, r, γ), где

– S — пространство состояний,
– A — пространство действий,
– O — пространство наблюдений,
– $T : S \times A \times S \longrightarrow R$ — модель перехода состояния,
– r — функция вознаграждения, и
– $\gamma \in [0,1)$ — коэффициент дисконтирования.

Мы предполагаем, что состояние не является непосредственно наблюдаемым, и изучаем стратегию $\pi(a|o)$, обусловленную наблюдениями $o \in O$.

Агент следует стратегии π. В дополнение к процессу принятия решений у нас есть система планирования $\Sigma = (S, A, T)$ и задача планирования $P =$

(Σ, s_0, g), которая обрабатывает последовательность действий $\{a_1, a_2, ..., a_n\}$. Мы предлагаем, чтобы стратегия π производила действия, тогда $a_1 = \pi(o_0), ..., g = \pi(o_{n-1})$. Предполагая, что стратегия параметризована θ, градиентный алгоритм оптимизирует θ, чтобы максимизировать ожидаемую будущую прибыль и минимизировать длину траектории:

$$\pi^* : \begin{cases} \pi^* = \underset{\pi(\theta)}{\operatorname{argmax}}\ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)|\pi\right] \\ \pi^* = \underset{\pi(\theta)}{\operatorname{argmin}}\ \|\{\pi(o_0), \pi(o_1), ..., \pi(o_{n-1})\}\| \\ \text{where } g = \pi(o_{n-1}) \end{cases} \tag{4.1}$$

Поскольку окружение частично наблюдается в каждый момент времени t, действие a_t генерируется из распределения $\pi^*(o_t)$ относительно наблюдения o_t. Наблюдение $o_t = (\Delta x_g, \Delta y_g, \Delta\theta, \Delta v, \Delta\gamma, \theta, v, \gamma, flag, l_1, l_2, l_3, ..., l_n)$, где $\Delta\theta$ — курсовой угол, Δv — разница между скоростью автомобиля и ограничением скорости на месте стоянки, $\Delta\gamma$ — разница между углом поворота автомобиля и ограничением угла поворота на месте стоянки, $flag$ — информация о текущей задаче (вперед, назад), добавленная для упрощения обучения, $l_1, l_2, l_3, l_4, ..., l_n$ — информация о лидарных данных, состоящая из n лучей, а $\Delta x_g, \Delta y_g$ показаны на рисунке 4.1.

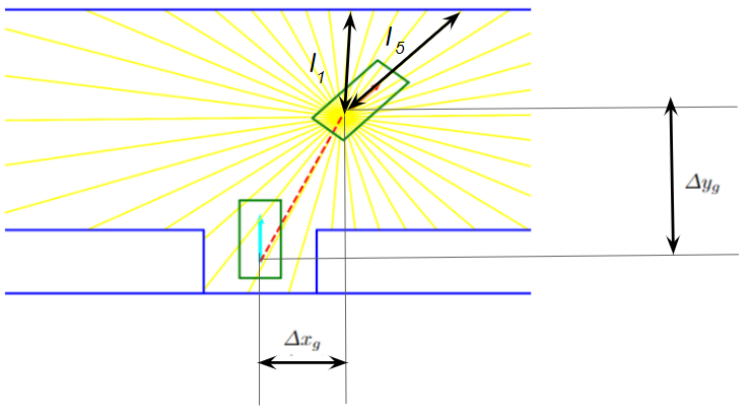


Рисунок 4.1 — Элементы наблюдений локального планировщика RL

Мы изменили действие агента на (a, ε), чтобы алгоритм RL сгенерировал соответствующий профиль скорости и ускорения для траектории, где a — линейное ускорение, а ε — угловое ускорение. Функция вознаграждения была изменена на:

$$R = w_r^T[r_{collision}, r_{timestep}, r_{distance}, r_{overspeeding}, r_{oversteering}, r_{action_a}, r_{action_{Eps}}], \quad (4.2)$$

где $r_{collision}$ равно -20 при столкновении агента с препятствиями и 0 в противном случае, r_{step} — постоянный шаг штрафа со значением 1, $r_{distance}$ — эвклидово расстояние до цели, $r_{overseeding}$, $r_{oversteering}$, равный -5, когда агент пытается превысить скорость, и $r_{action_a}, r_{action_\varepsilon}$ — штрафы за абсолютное значение линейного и углового ускорения, которые равны $-0.5 * a_t - 0.5 * \varepsilon_t$ в момент времени t.

Мы используем локальный планировщик POLAMP [66] под названием PPO [67]. Был взят gym - симулятор из статьи [66], в нем есть агент, представляющий собой нейронную сеть, которая принимает наблюдение o_t и возвращает ожидание и дисперсию линейного ускорения и угловой скорости. Во время обучения актор обновляет веса стратегии π с учетом градиента:

$$\Delta_w J(w) = E_{\pi_w} \Delta_w log \pi_w(s, a) A^{\pi_w}(s, a) \quad (4.3)$$

Критик используется для предсказания функции преимуществ $A^\pi(o_t, a_t)$, которая определяет, насколько предпочтительнее выбрать действие a_t, чем o_t. Кроме того, актор обновляет свои веса в зависимости от функции потерь:

$$\begin{aligned} L(s, a, w_k, w) = min(\pi_w(a|s)/\pi_{wk}(a|s) * A^{\pi_{wold}}(s, a), \\ clip(\pi_w(a|s)/\pi_{wold}, 1 - \varepsilon, 1 + \varepsilon) A^{\pi_{wold}}(s, a)) \end{aligned} \quad (4.4)$$

Веса критика обновляются путем оценки $A^\pi(s_t, a_t)$ по сгенерированным траекториям.

4.2 Модель автомобиля

Мы изменили динамическую модель агента на модель велосипеда с точкой отсчета в центре задней оси транспортного средства.

$$\begin{aligned} \dot{x_r} &= v * \cos(\theta) \\ \dot{y_r} &= v * \sin(\theta) \\ \dot{\theta} &= \frac{V * tan(\delta)}{L} \end{aligned} \quad (4.5)$$

Кинематические ограничения были изменены следующим образом:

$$\begin{aligned} -2\ m/s <= v <= 2\ m/s \\ -1.5\ m/s^2 <= a <= 1.5\ m/s^2 \\ -1.5\ rad/s^2 <= \varepsilon <= 1.5\ rad/s^2, \end{aligned} \tag{4.6}$$

а ограничения для места парковки (терминальные ограничения) таковы:

$$\begin{aligned} distance <= 0.5\ m \\ -0.5\ m/s <= v <= 0.5\ m/s \end{aligned} \tag{4.7}$$

4.3 Обучение по расписанию

Мы использовали подход, основанный на обучении по расписанию, чтобы научить агентов парковаться в критических ситуациях со статичными и динамическими препятствиями. Для обучения были выбраны четыре основных этапа (рисунок 4.2): обучение в пустой среде со статичными препятствиями, обучение в союзе, когда мы назначаем место парковки в качестве цели после того, как агент выполнил первую цель. Также существует несколько последовательных обучений в рамках каждого этапа. Например, чтобы агент научился достигать цели с ограничениями по расстоянию, курсовому углу и скорости, его сначала обучали только с ограничениями по расстоянию; в противном случае агенту будет сложно завершить эпизод.

Обучение агента проходило в четыре этапа, но количество обучений на каждом этапе могло варьироваться. Например, чтобы помочь агенту добраться до места парковки с учетом всех ограничений, сначала агент обучался в пустой среде с одним условием $distance <= 0.5m$, затем мы добавили условие угла $angle <= 15^\circ$ и условие скорости $speed <= 0.5m/s$.

На рисунке 4.3 изображена архитектура нейронной сети, используемой для обучения. Мы объединяем три последних наблюдения o_{t-1}, o_{t-2} и o_{t-3} с текущим наблюдением o_t. Поскольку используется $n = 39$ лидарных лучей, вектор наблюдений имеет длину 48. При объединении предыдущих наблюдений мы получаем вектор длиной $48 * 4 = 192$. Для обучения актора в нейронную

сеть подается партия из 8000. Для обучения критика действие агента было добавлено к входному вектору, поэтому его размер составляет 194.

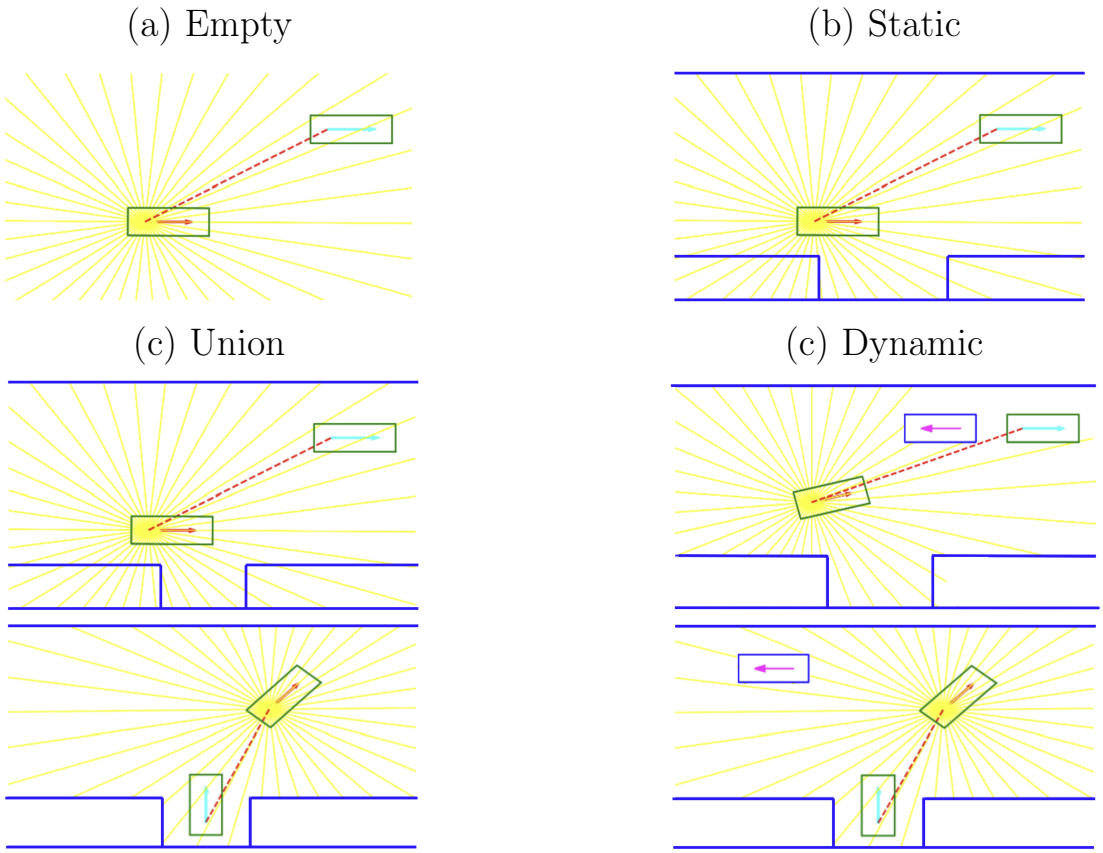


Рисунок 4.2 — Этапы обучения по расписанию

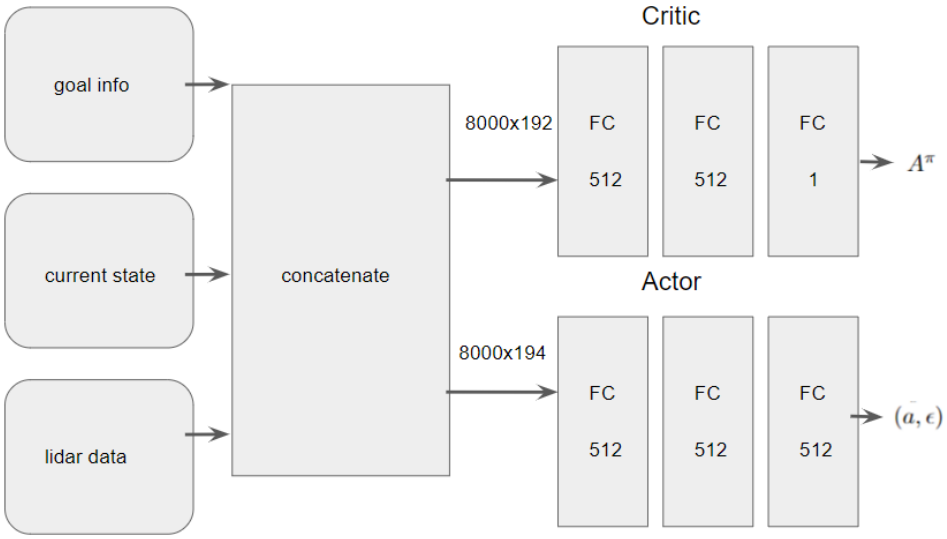


Рисунок 4.3 — Архитектура агента актор-критик (Actor critic architecture)

4.4 Реализация на платформе Apollo

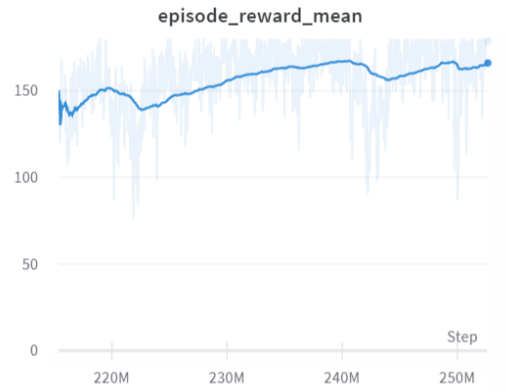
Планировщик POLAMP и симулятор Apollo взаимодействуют с разработанными темами Cyber, читателями и писателями. Прежде всего, для инициализации симулятора POLAMP требуется информация о статических препятствиях в окружении и границах задачи, куда может отправиться агент. Эту информацию собирает алгоритм определения региона интересов (ROI, Region Of Interest decider) в Apollo. Информация о динамических препятствиях поступает в ROI-дешифратор через модуль восприятия Apollo, который каждые 0,1 секунды публикует положение и вектор скорости динамических препятствий. Когда информация появляется в планировщике POLAMP, запускается симулятор POLAMP. После этого агент генерирует один эпизод в предположении равномерного прямолинейного движения динамических препятствий, сохраняет точки траектории в виде набора (x, y, курсовой угол, угол поворота, линейное ускорение, угловое ускорение) и предоставляет траекторию в Apollo через созданную тему.

Профили скорости и ускорения траектории были сглажены в Apollo, поскольку скорость в конечной точке каждой траектории POLAMP для движения вперед и назад может варьироваться от $-0{,}5$ до $0.5m/s$. Если конечная точка текущей траектории имеет индекс i, а конечная точка места парковки — индекс j, то новое линейное ускорение и скорость для этой точки рассчитываются следующим образом:

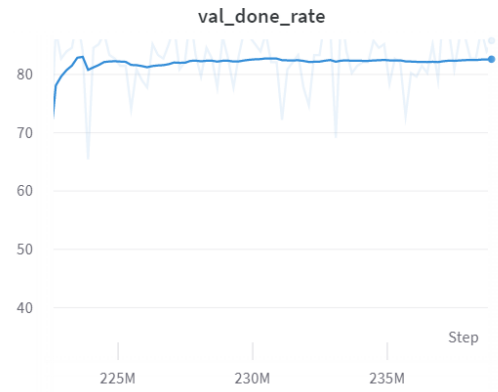
$$\begin{aligned} a(i-1) &= -v(i)/dt \\ v(i) &= 0 \\ a(i) &= v(i+1)/dt \\ a(j-1) &= -v(j-1)/dt \\ v(j) &= 0 \\ a(j) &= 0 \end{aligned} \tag{4.8}$$

4.5 Результаты

На рисунке 4.4 показаны результаты, полученные после обучения алгоритма на симуляторе POLAMP.

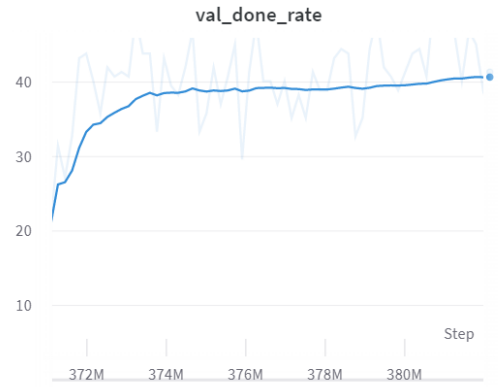


(a) static reward



(b) static val

(c) dynamic reward



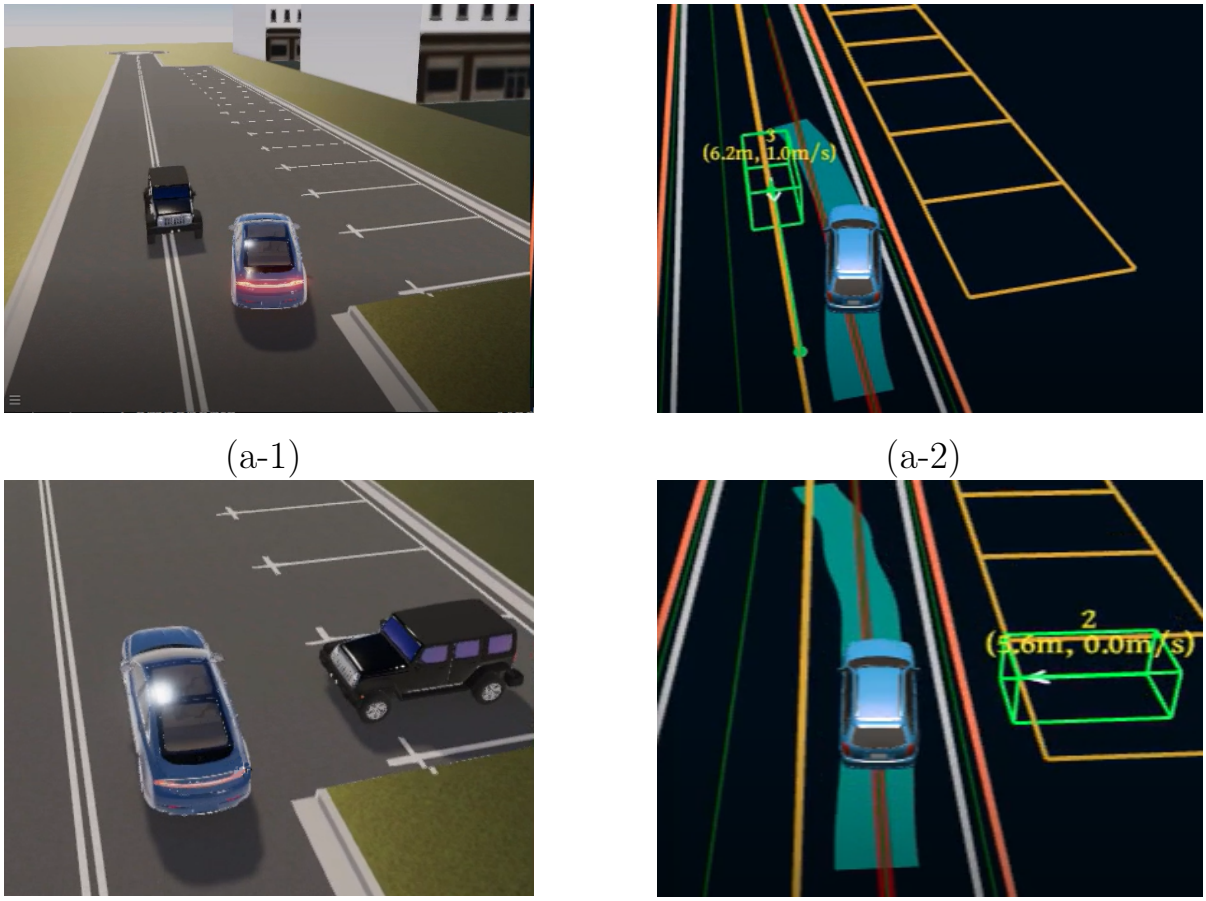
(d) dynamic val

Рисунок 4.4 — Результаты обучения POLAMP

Основные метрики: среднее вознаграждение за эпизод — среднее вознаграждение за всю партию (включает 8000 заданий) и val done rate — количество выполненных заданий на валидационном наборе данных (включает 600 заданий). На узких дорогах основными случаями, когда агент не завершает эпизод, являются ситуации, в которых агент должен зафиксировать направление вектора движения, чтобы достичь места парковки с учетом всех ограничений, что некритично для распространенных сценариев парковки.

Мы обучили и протестировали алгоритм с помощью платформы Apollo и симулятора SVL. Если столкновение с агентом возможно, значит, алгоритм

успешно научился избегать динамических препятствий. Если препятствий нет, алгоритм находит оптимальную траекторию, как показано на рисунке 4.5.



(a-1) (a-2)

(b-1) (b-2)

Рисунок 4.5 — Различные сценарии парковки с динамическими препятствиями были протестированы в симуляторе SVL (левые рисунки) и в Apollo (правые рисунки).

Среди оставшихся невыясненных случаев выполнения алгоритма беспилотный автомобиль прокладывает траекторию при движении задним ходом с некоторой ошибкой. Эта проблема возникает из-за того, что встроенный модуль Control не в состоянии отследить профиль скоростей и ускорений, предусмотренный алгоритмом POLAMP.

4.6 Заключение

В этой главе мы рассмотрели применение методов обучения с подкреплением (RL) для адаптивного планирования маневров парковки с

упором на алгоритм POLAMP. Мы сформулировали задачу парковки как частично наблюдаемый марковский процесс принятия решений (POMDP) и использовали алгоритм оптимизации ближайшей стратегии PPO в качестве основного RL-алгоритма. Метод был усовершенствован путем обучения по расписанию (curriculum learning), что позволило агенту постепенно осваивать все более сложные задачи парковки, включая динамическое избегание препятствий и точное маневрирование в ограниченном пространстве.

Интеграция этого подхода в платформу автономного вождения Apollo продемонстрировала практическую жизнеспособность метода. Планировщик на основе RL эффективно научился адаптироваться к изменениям в окружающей среде в реальном времени, успешно генерируя оптимизированные траектории, учитывающие как статичные, так и динамические препятствия. Экспериментальные результаты подтвердили эффективность подхода, показав способность агента избегать столкновений и эффективно парковаться в различных сценариях даже при наличии сложных ограничений.

Несмотря на многообещающие результаты, были выявлены некоторые ограничения, такие как периодические ошибки траектории во время маневров парковки задним ходом. Эти проблемы в основном связаны с расхождениями между модулями управления в Apollo и профилями ускорения и скорости, сгенерированными POLAMP. Последующие исследования должны быть направлены на совершенствование интеграции управления для обеспечения более плавной работы в сложных сценариях парковки.

В целом, данная глава вносит вклад в растущий объем исследований по применению RL для автономного вождения, демонстрируя, что адаптивное планирование маневров с использованием RL может значительно повысить безопасность и эффективность работы парковщиков. Рассмотренные здесь методы закладывают основу для дальнейших усовершенствований, в том числе для распространения фреймворков RL на более широкие задачи автономного вождения.

Глава 5. Быстрый эвристический поиск с помощью потоков (streams)

В настоящей главе мы исследуем алгоритм быстрого эвристического поиска в контексте автоматического планирования для беспилотных автомобилей, уделяя особое внимание интеграции потоковых сэмплеров в детерминированное планирование для повышения эффективности процесса поиска. Необходимость быстрого и надежного принятия решений в автономном вождении требует разработки алгоритмов поиска, которые могут справиться с динамической и неопределенной природой реальной среды. Потоки обеспечивают структурированный метод обработки непрерывных входных данных, что представляет существенную важность для планирования маневров и построения траекторий в условиях высокой динамики. Изложенный в этой главе подход был опубликован в статье [3].

Данная глава построена следующим образом. В разделе 5.1 представлена постановка задачи. В разделе 5.2 представлены используемые методы быстрого эвристического поиска с потоками, включая домен Maneuver-planning для планирования маневров обгона, смены полосы движения, уступки и удержания полосы движения, поток конфигураций (configuration Stream) для генерации новых конфигураций автомобиля и поток траекторий (trajectory Stream) для генерации новых траекторий для выполнения маневров. В разделе 5.3 изложены экспериментальные результаты и оценка предложенного подхода. Наконец, в разделе 5.4 представлены выводы.

5.1 Постановка задачи

Входные данные включают в себя конфигурацию беспилотного автомобиля $q(t) = [x, y, \theta, v, a]$, представляющую положение, угол поворота, скорость и ускорение автомобиля, а также состояния m динамических препятствий $B_i : i \in \{1, \ldots, m\}$, которые описываются их положениями, углами поворота и скоростями $[x_{bi}, y_{bi}, \theta_{bi}, v_{bi}]$. Кроме того, входные данные включают в себя информацию карты $\mathcal{M} \in M$, такую как опорные

линии и полосы движения. Время T дискретизируется на Δt. Возможные действия определяются множеством $A = \{a_1, ..., a_n\}$, где $n = 5$, а действия включают в себя "$Yield$" — уступить дорогу другому препятствию и замедлиться, "$Follow\ speed$" — следовать заданной скорости и ускориться, "$Change\ to\ left\ lane$" — перестроиться в левую полосу, "$Change\ to\ right\ lane$" — перестроиться в правую полосу, "$Collision\ check$" — проверить будущее столкновение с препятствием путем симуляции движения вперед. Множество выборок потока $St = \{st_1, ..., st_k\}$, где $k = 2$, а выборки потока включают в себя "Configuration stream"поток конфигураций, и "Trajectory stream"поток траекторий.

Учитывая начальную конфигурацию q_0 и целевую область G, задача состоит в том, чтобы найти выполнимый план $\pi = [a_0, a_1, \ldots, a_T]$, используя планировщик на основе эвристического поиска и выборки $D : Q \times B \times M \to A$. Следуя плану π, цель G должна быть достигнута из начальной конфигурации q_0, образуя траекторию $\tau = (q_0, q_1, \ldots, q_g)$. Время выполнения плана должно быть минимизировано, при этом траектория $\tau(t)$ не должна приводить к столкновению с препятствием, $\tau(t) \cap \forall b_i(t) \in B = \varnothing$.

Смена полосы движения, особенно на шоссе (рисунок 5.1 (a)), является критически важным маневром в автономном вождении, требующим безопасных и комфортных траекторий. Обгон движущегося препятствия — еще один критический маневр, включающий в себя две смены полосы движения, что добавляет сложности, особенно на дорогах с двусторонним движением и потенциальным встречным потоком транспорта (рисунок 5.1 (b)). В таких сценариях система планирования ставит на первое место безопасность и человекоподобное поведение водителя, чтобы свести к минимуму неудобства для других водителей.

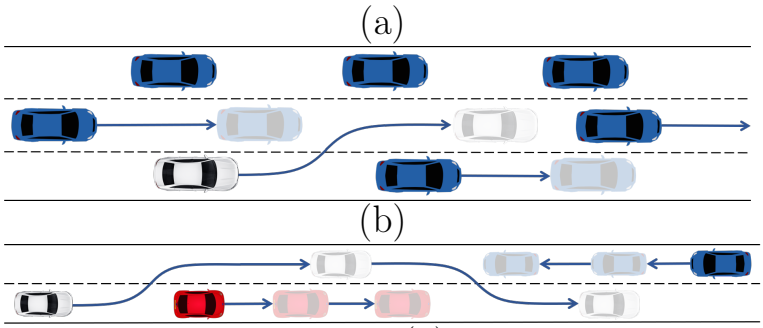


Рисунок 5.1 — Критические сценарии: (a) Смена полосы движения на шоссе. (b) Обгон движущегося на низкой скорости переднего препятствия на дороге с двусторонним движением.

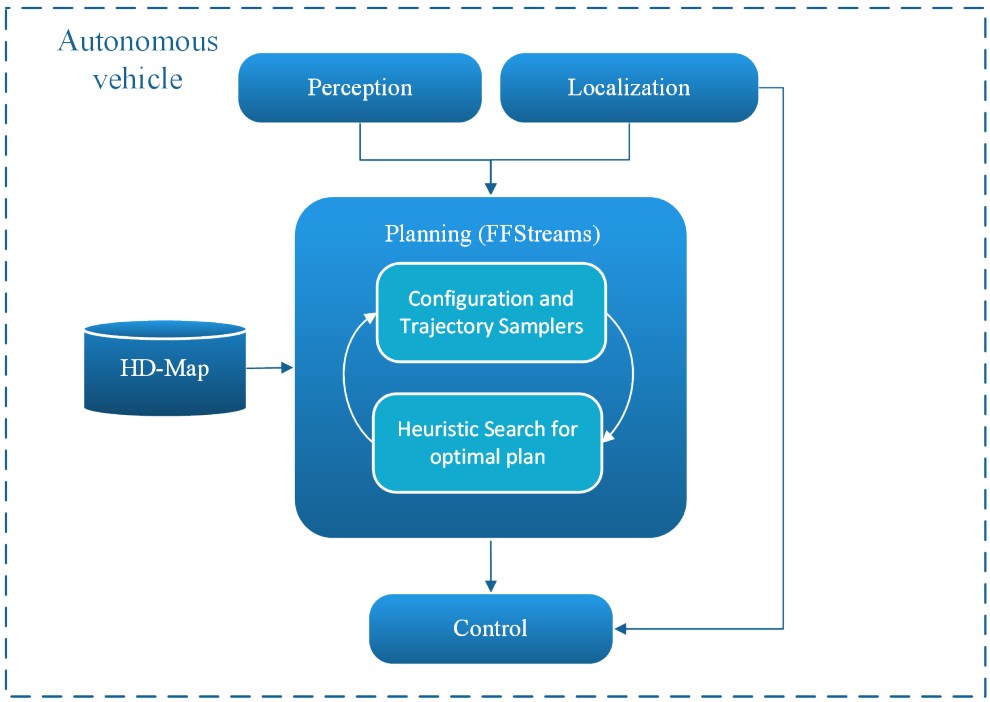


Рисунок 5.2 — Схема комбинированного принятия решений и планирования движения.

5.2 Методы

Для решения задач автономного вождения мы рассматриваем объединенную задачу принятия решений и планирования движения (рисунок 5.2). Она представлена как детерминированное планирование с использованием сэмплеров. Мы моделируем домен планирования с помощью PDDL 2.1, производим модификации с помощью Streams и используем эвристический поиск FastForward для итерационного поиска оптимального плана.

Мы предлагаем подход FFStreams, при котором процесс планирования чередуется с сэмплированием траекторий и поиском плана в PDDL. Часть планировщика, основанная на сэмплировании, использует потоки для генерации траекторий, jerk-оптимизированных для различных маневров (удержание полосы движения, смена полосы движения и обгон). В отличие от этого, часть, основанная на поиске, соответствует принятию решений на высоком уровне, проверяя траектории-кандидаты на столкновение и выбирая действие с минимальной стоимостью, соответствующее траектории. Наш подход позволяет планировать в высокоразмерных пространствах и

сложных средах благодаря сэмплированию и поиску оптимального плана через дискретизированное представление конфигурационного пространства задачи.

Для эвристического поиска FastForward мы используем планировщик Fast-Forward (FF)[68], реализованный на языке C. Мы использовали версию Metric-FF-v2.1 и внесли правки для расширения возможностей парсера по обработке более обширного ввода и для решения проблем сравнения, связанных с операциями над типом данных Float в C.

Моделирование планирования задач на языке PDDL позволяет интерпретировать процесс поиска плана решения. Потоки помогают осуществлять сэмплирование новых конфигураций и траекторий движения автомобиля с учетом кинематики беспилотного автомобиля. Поиск с помощью планировщика FF позволяет реализовать численные операции в домене и привязать стоимость к каждому действию. Мы используем алгоритм Incremental для итераций между процессами сэмплирования и поиска. Мы используем один домен для задач автономного вождения, включая маневры по сохранению полосы движения, уступке препятствия, смене полосы движения и обгону.

Планирование движения требует предварительного знания текущего состояния беспилотного автомобиля, а также состояния других препятствий, таких как координата, курс и скорость. Мы предполагаем, что восприятие состояния окружающих препятствий является идеальным. Кроме того, мы исходим из того, что все динамические препятствия движутся с постоянной скоростью.

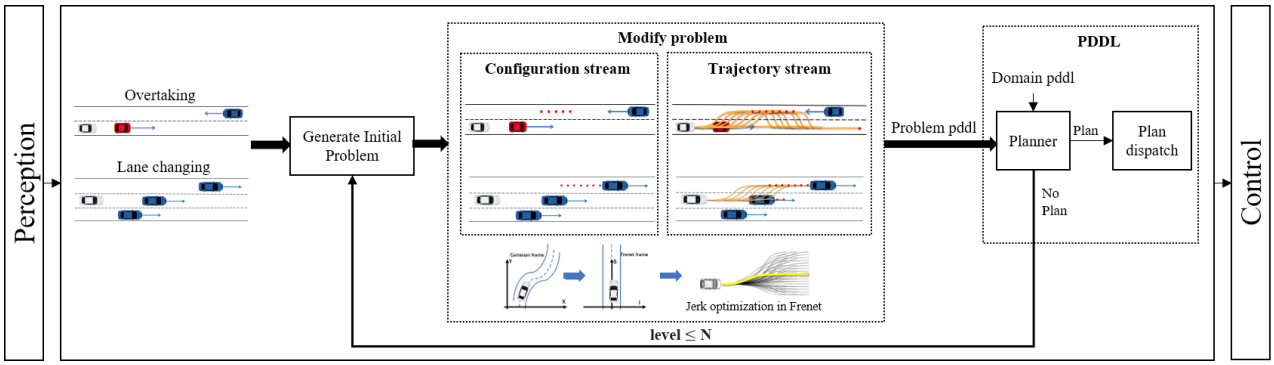


Рисунок 5.3 — Схема планировщика маневров FFStreams.

После наблюдения за состоянием динамических препятствий предложенный алгоритм планирования осуществляет итерационное сэмплирование новых конфигураций и возможных траекторий на N-уровнях,

обновляет задачу PDDL и ищет план. Как показано на рисунке 5.3 и иллюстрируется алгоритмом 2, задача планирования имеет на входе начальную конфигурацию эго q_0, и предикаты цели P_{goal}, которые включают движение вперед в одном из двух случаев: по текущей полосе в случае соблюдения скорости, уступки и маневра обгона; по соседней полосе в случае маневра смены полосы движения, начальное наблюдение в виде множества препятствий, их координат и скоростей $O(0)$, домен планирования Σ, и максимальное количество уровней N. Мы установили множество максимальных уровней: пять. Значение максимального уровня N было подобрано экспериментально в ущерб производительности и времени работы алгоритма.

Algorithm 2 Autonomous Maneuver Planning

$INPUT : q_0$,$O(0)$,$N(levels)$,P_{goal},Σ
$OUTPUT : traj = [q_1,...,q_k], [a_1, \ldots, a_k]$
$initialize$:
$t \leftarrow 0$
$traj \leftarrow \varnothing$
while $t \leqslant t_k$ **do**
  $Q(t) \leftarrow q_0$
  $\mathcal{P}(t) \leftarrow GenerateInitialProb(q_0,O(t),P_{goal})$
  for $l = 1,2,...,N$ **do**
    $ApplyApplicableStreams(\mathcal{P}(t))$
    $\mathcal{P}(t) \leftarrow \mathcal{P}(t) \cup st.cert$
    $\pi \leftarrow FFPlanner(\Sigma,\mathcal{P}(t))$
    if π **then**
      $a_t,q_1 \leftarrow PlanDispatch(\pi)$
      $traj \leftarrow traj \cup q_1$
    end if
  end for
  $q_0 \leftarrow q_1$
  $O(t) \leftarrow ObserveEnv()$
  $t \leftarrow t + 1$
end while

На каждом временном шаге t генерируется начальная задача PDDL; для N уровней вызываются все применимые потоки для обновления задачи

до нахождения плана. Конфигурации строятся из начальной конфигурации q_0, сэмплирования конфигураций на текущей и соседних дорожках; $Q(t) = [q_0,q_1,...,q_n]$. Первый поток, **Configuration stream**, генерирует новые конфигурации $[q_1,q_2,...,q_n]$ автономного автомобиля, включая его 2D-координату Френе и скорость. Сэмплирование различных скоростей новых конфигураций позволяет планировать ускорение и замедление траекторий, необходимых для различных маневров. Второй поток, **Trajectory stream**, отвечает за генерацию возможных дискретизированных траекторий $traj(q_i,q_j)$ между каждыми двумя конфигурациями q_i,q_j, строя граф связности конфигураций. Если кинематическая траектория существует, то она состоит из множества промежуточных конфигураций $[q_{i\_j\_1},q_{i\_j\_2},...,q_{i\_j\_s}]$, где s — константа, равная времени планирования, деленному на Δt.

После вызова каждого применимого потока его сертифицированные факты добавляются в задачу PDDL. В конце каждого уровня обновленная задача и домен передаются PDDL-планировщику FastForward для поиска плана π. При нахождении плана извлекается первое действие вместе с соответствующей траекторией. Действие применяется, и состояние эгомобиля обновляется до следующей конфигурации, а состояние препятствий — до нового наблюдения. Действие может быть одним из следующих: соблюдение скорости, уступка, перемещение в левую или правую полосу и обгон. Конечным результатом работы алгоритма является вся траектория от начального состояния до состояния цели $traj = [q_1,...,q_k]$, план следования и его множество действий: $\pi = [a_1, \ldots, a_k]$.

5.2.1 Домен планирования маневров

Представляя наш домен планирования на языке PDDL, мы определяем два типа объектов в нем: конфигурация автомобиля $-conf$ и препятствия $-obstacles$. Кроме того, множество предикатов, включая *(traj* $?q_1$ $?q_2$*)*, указывающий на существование траектории от конфигураций $?q_1$ до $?q_2$, *(next* $?q_1$ $?q_2$ $?q_{end}$*)*, указывающий на последовательность подконфигураций на траектории к конечной конфигурации $?q_{end}$, *idle*, указывающий, что процесс проверки столкновений простаивает, чтобы начать новую

проверку, *(checking_traj ?q_1 ?q_2 ?o)*, указывающий, что траектория между конфигурациями ?q_1 и ?q_2 проверяется с препятствием ?*o*, *(checked_traj ?q_1 ?q_2 ?o)*, указывающая, что траектория между двумя конфигурациями проверяется с препятствием ?*o* и является свободной от столкновений, *(ego_at ?q)*, указывающая, что беспилотный автомобиль в данный момент находится в конфигурации ?*q*, *(on_right_lane)* и *(on_left_lane)*, указывающие текущую полосу движения беспилотного автомобиля.

Функции домена планирования маневров включают в себя следующие изменяющиеся переменные: *(total_cost)* — общая стоимость, *(curr_time)* — текущее время, *(time_of_traj ?q_1 ?q_2)* — продолжительность следования по траектории, *(at_s ?q) (at_l ?q) (at_time ?q)* — координация френетов и время достижения определенной конфигурации, *(obst_at_s ?o) (obst_at_l ?o) (obst_at_speed ?o)* — координация френетов и скорость определенного препятствия, когда текущее время равно нулю.

Домен автономного планирования маневров состоит из 14 действий. Четыре действия по движению вперед по определенной траектории от первой конфигурации *?q_1* до второй конфигурации *?q_2*; соблюдение скорости, уступка, переход на левую полосу и переход на правую полосу. В предпосылках движущегося действия эгомобиль должен находиться в первой конфигурации *(ego_at ?q_1)*. Между двумя конфигурациями должна существовать траектория *(traj ?q_1 ?q_2)*; эта траектория свободна от столкновений после проверки со всеми существующими препятствиями ?*o*; *(checked_traj ?q_1 ?q_2 ?o)*. Эффект от такого действия следующий: эгомобиль уже не в первой конфигурации, а во второй *(ego_at ?q_2)*, текущее время увеличивается на время следования по траектории. В случае смены полосы движения «я» оказывается на соседней полосе (например, *(on_left_lane)*), а не на начальной. Пример действия по смене полосы движения на левую показан ниже.

Предсказание будущей траектории движения окружающих автомобилей очень важно для планирования движения. В данной работе мы реализуем предсказание в домене PDDL в действиях по проверке столкновений. Из-за ограничения линейности функций PDDL2.1 мы моделируем движение окружающих автомобилей как $\Delta x = V.\Delta t$, где V — постоянная скорость. Однако, поскольку планирование выполняется в реальном времени, любое изменение скорости окружающих автомобилей учитывается в каждом цикле планирования, и план модифицируется в соответствии с этим.

```
(:action change_lane_left
    :parameters (?q1 ?q2 - conf)
    :precondition (and (ego_at ?q1)
        (traj ?q1 ?q2)
        (> (at_l ?q1)(at_l ?q2))
        (forall (?o - obstacles)
            (checked_traj ?q1 ?q2 ?o))
        (on_init_lane) (idle))
    :effect (and (moved_forward)
        (ego_at ?q2)(not (ego_at ?q1))
        (on_left_lane) (not (on_init_lane))
        (increase (curr_time) (time_of_traj ?q1 ?q2))
        (increase (total_cost) 3)))
```

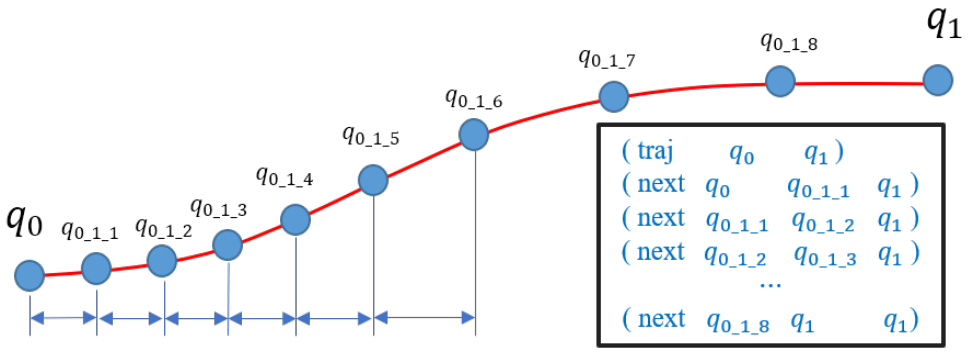


Рисунок 5.4 — Дискретизация траектории на постоянную Δt и преобразование в предикаты для проверки столкновений.

Каждая траектория дискретизируется на Δt и ассоциируется с конфигурациями (рисунок 5.4). Для проверки текущей траектории с существующим препятствием мы используем одно действие begin_check, восемь действий для восьми ситуаций проверки и одно действие end_check.

Действия по проверке столкновений перебирают все конфигурации траектории и в каждый момент времени t_i сравнивают продольную и поперечную координацию автомобиля в этой конфигурации с предсказанием координации препятствия. Мы продемонстрируем процесс проверки на примере одного действия из восьми. Восемь ситуаций проверки столкновений связаны с полосой препятствия, полосой «я», Δs и Δl в системе координат Френе (рисунок 5.5).

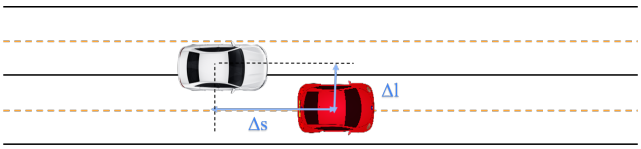


Рисунок 5.5 — Линейная проверка столкновений по Δs и Δl в системе координат Френе. Мы выделяем восемь случаев проверки линейного столкновения двух прямоугольников. Беспилотный автомобиль и препятствие находятся на одной полосе: автономный автомобиль впереди, автономный автомобиль сзади — мы проверяем расстояние по Δs, если оно безопасно. Беспилотный автомобиль и препятствие находятся на разных полосах: автономный автомобиль находится слева сзади, справа сзади, слева впереди или справа впереди препятствия — мы проверяем расстояние по Δs и Δl, если оно больше пределов безопасности.

5.2.2 Поток конфигураций (Configuration Stream)

Поток конфигурации (Conf) отвечает за генерацию новых конфигураций. Этот поток осуществляет сэмплирование новых конфигураций на текущей и соседних полосах с различными скоростями. Когда этот поток генерирует образец конфигурации на текущей полосе с более высокой скоростью, это помогает перемещаться по полосе с ускорением, а сэмплирование конфигурации движения с более низкой скоростью способствует замедлению, что необходимо, когда перед беспилотным автомобилем находится медленно движущееся транспортное средство. Сэмплирование на соседних полосах со скоростью, превышающей текущую скорость в пределах ограничений по ускорению, позволяет выполнять маневры по смене полосы движения и обгону. Поток Conf демонстрируется ниже.

```
(:stream conf
    :inputs (?q1)
    :domain (and (ego_at ?q1) (at_s ?q1) (at_l ?q1)
        (at_time ?q1))
    :outputs (?q2)
    :certified
        (and (at_s ?q2)(at_l ?q2) (at_time ?q2))  )
```

```
(:action begin_check
    :parameters(?first_q ?last_q -conf ?o -obstacles)
    :precondition (and
       (idle)(ego_at ?first_q)(traj ?first_q ?last_q)
       ( not (checked_traj ?first_q ?last_q ?o )))
    :effect (and
       (is_first ?first_q ?o)(is_last ?last_q ?o)
       (checking_traj ?first_q ?last_q ?o)(not(idle)))
       )

(:action check_forword_diff_lane_upper_bigger_delta_y
    :parameters (?most_first ?first ?after_first
        ?last - conf ?o - obstacles)
    :precondition (and (is_first ?first ?o)
        (is_last ?last ?o)
        (next ?first ?after_first ?last)
        (checking_traj ?most_first ?last ?o)
        (not (= ?first ?last))
        (> (at_l ?after_first)(obst_at_l ?o))
        (>= (- (at_l ?after_first)(obst_at_l ?o)) 3))
    :effect (and
        (not (is_first ?first ?o))
        (is_first ?after_first ?o)))

(:action end_check
    :parameters (?first_first ?element - conf
        ?o - obstacles)
    :precondition (and  (is_first ?element ?o)
        (is_last ?element ?o)
        (checking_traj ?first_first ?element ?o))
    :effect (and
        (not (is_first ?element ?o))
        (not (is_last ?element ?o))
        (not(checking_traj ?first_first ?element ?o))
        (checked_traj ?first_first ?element ?o )
        (idle)))
```

5.2.3 Поток траекторий (Trajectory Stream)

Поток траектории генерирует возможные траектории между двумя конфигурациями, соответственно. Траектория планируется в системе координации Frenet. Для оптимизации траектории мы минимизируем квадратичный рывок за интервал времени T [69]; сначала определяется множество квинтовых полиномов для бокового движения и квартовых полиномов для продольного движения, затем вычисляется их стоимость и выбирается траектория с минимальной стоимостью. Стоимость каждой траектории представляет собой взвешенную сумму двух затрат на продольное

и поперечное движение и задается уравнением:

$$C_{total} = k_{lat}C_l + k_{lon}C_s. \tag{5.1}$$

Стоимость траектории при боковом движении является функцией боковых рывков J_l, длительности траектории и боковой ошибки и задается уравнением:

$$C_l = k_j \sum J_l^2 + k_t T + k_d dl^2, \tag{5.2}$$

где k_j — вес рывка, T — временной интервал, а dl — боковая ошибка конечной точки траектории. Стоимость траектории на продольное перемещение является функцией продольных рывков, длительности траектории и продольной ошибки и задается уравнением:

$$C_s = k_j \sum J_s^2 + k_t T + k_d ds'^2, \tag{5.3}$$

где k_j — вес рывка, T — интервал времени, а ds' — ошибка продольной скорости в конечной точке траектории. Параметры оптимизации указаны в таблице 4. Ниже показан поток движения_траектории.

```
(:stream motion_trajectory
    :inputs (?q1 ?q2)
    :domain (and (at_s ?q1) (at_l ?q2) (at_time ?q1)
        (at_s ?q2) (at_l ?q2) (at_time ?q2) )
    :outputs (?q_{1_2_1} ... ?q_{1_2_24})
    :certified
        (and (traj ?q1 ?q2) (time_of_traj ?q1 ?q2)
        (next ?q1 ?q1_2_1 ?q2) ...
        (next ?q1_2_23 ?q1_2_24 ?q2)))
```

5.3 Эксперименты и результаты

Предложенный метод был проверен с помощью имитационных экспериментов в двух основных сценариях движения и реальных условиях: (1) одна полоса движения вперед и двунаправленная соседняя полоса с различными скоростями и начальными положениями препятствий; (2) многополосная автомагистраль; (3) три дополнительных сценария CommonRoad.

Таблица 4 — Оптимизация транспортного средства и траектории движения — Параметры SOFT и HARD

Parameter	Symbol	Value(soft)	Value(hard)
Maximum acceleration	a_{max}	$0.5[m/s^2]$	15 $[m/s^2]$
Jerk weight	k_j	0.1	0.08
Trajectory duration weight	k_t	0.1	0.9
Error weight	k_d	1.0	1.0
Maximum speed	v_{max}	33.33 $[m/s]$	
Maximum Curvature	K	1 $[1/m]$	
Time step	Δt	0.2 $[s]$	
Lane width	l	3.4 $[m]$	
Lateral weight	k_{lat}	1.0	
Longitudinal weight	k_{lon}	1.0	

Сценарий 1. Две полосы движения в разных направлениях

Сценарий включает в себя два препятствия: низкоскоростное динамическое препятствие на той же полосе, что и эгомобиль, расположенное на расстоянии 50 m впереди со случайной скоростью $[7m/s, 8m/s]$, и другое встречное препятствие на соседней полосе, движущееся в противоположном направлении. Встречное препятствие стартует из случайной позиции на оси x [50 m,350 m] со скоростью [4 $m/s, 12$ m/s]. Беспилотный автомобиль с начальной скоростью v_0, равной 10 m/s, стремится обогнать препятствие спереди. Он может либо ускориться, чтобы обогнать переднее препятствие до того, как его проедет встречное, либо уступить, регулируя скорость, пока встречное препятствие не проедет, а затем ускориться, чтобы обогнать его.

В 100 случайных сценариях наш метод с мягкими параметрами, оптимизированными для комфорта (таблица 4), достиг 92% успеха при обгоне. В 9% случаев обгон был завершен до проезда препятствия по соседней полосе, а в 83% — после проезда препятствия, что соответствует более безопасному и человекоподобному вождению. В 8% случаев планировщик не смог найти план в пределах максимального уровня. Критический сценарий показан на рисунке 5.6, демонстрируя запланированную траекторию, скорость, ускорение и профили рывков. Планируемые действия включают в себя сохранение полосы движения

и ускорение для обгона при наличии безопасной траектории, обеспечение плавности траектории и максимальный рывок в 0.8 m/s^3.

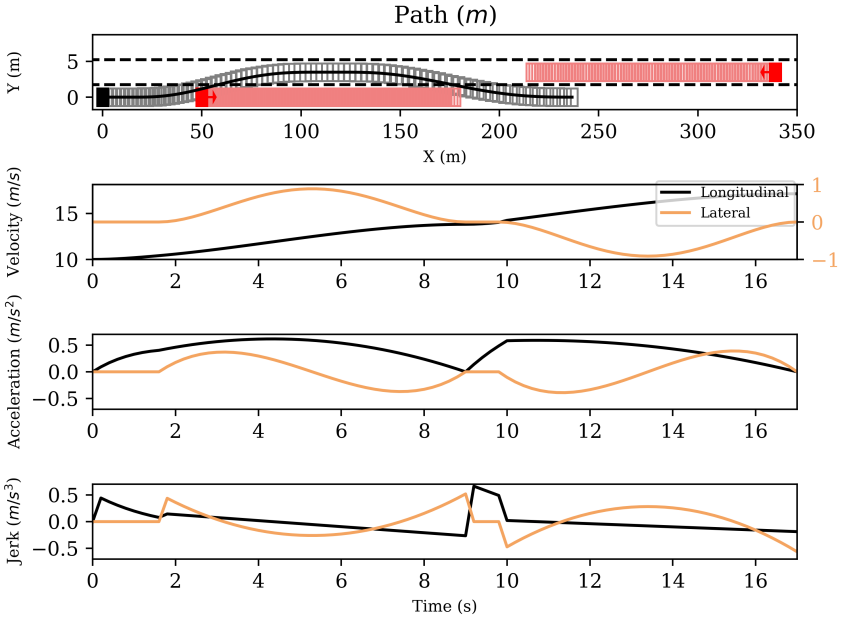


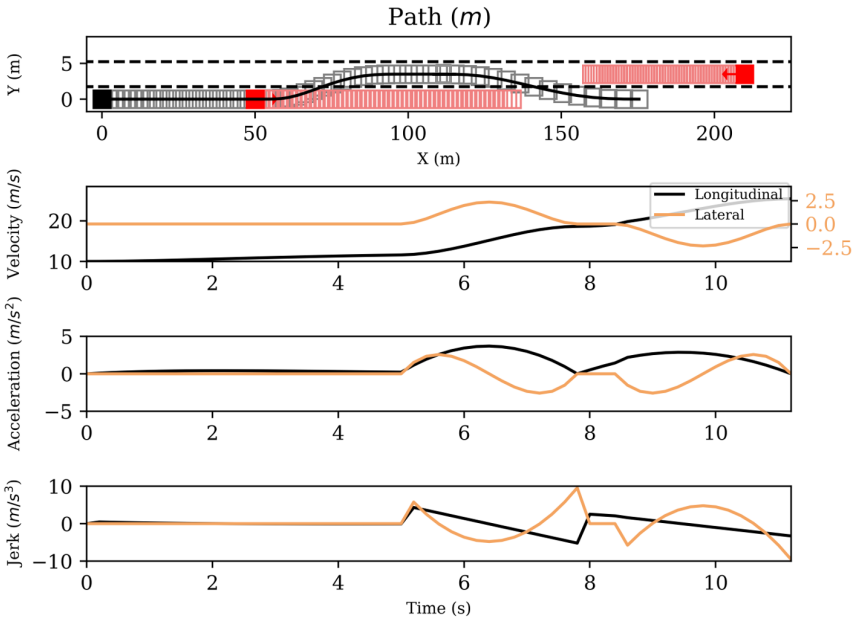
Рисунок 5.6 — Успешный обгон с помощью планировщика FFStreams с мягкими параметрами при критическом сценарии, когда скорость встречного препятствия составляет 7.22 m/s.

Мы сравнили FFStreams с открытыми планировщиками Monte Carlo Tree Search with Optimistic Exploration for Deterministic Systems (MCTS-OPD) и Interval-based Robust Planning (IRP)[70], реализованными в[71]. Мы сконфигурировали наш метод с жесткими параметрами оптимизации (таблица 4). Были выделены три варианта поведения: 1) обгон до того, как препятствие проедет ; 2) уступка до того, как препятствие проедет, затем обгон; и 3) следование по небезопасной траектории, ведущей к столкновению. Траектории оценивались с помощью Occupant's Preference Metric (OPM)[72], учитывающей комфорт на основе максимального бокового (lateral) и продольного (longitudinal) рывка.

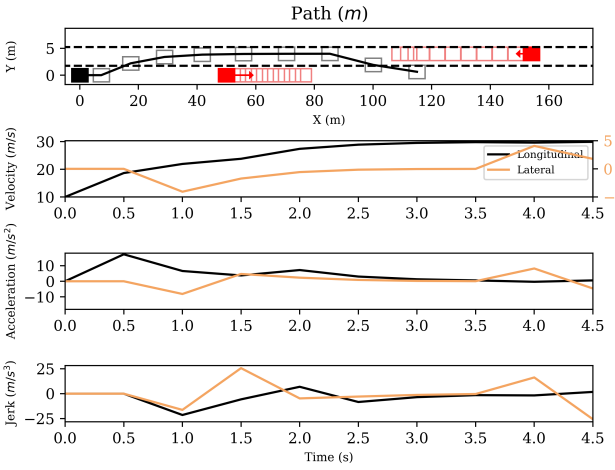
В 100 случайных экспериментах (таблица 5) FFStreams добился 94% успеха при обгоне, MCTS-OPD — 82%, а IRP — 62%. Неудачные попытки обгона в MCTS-OPD (18%) и IRP (38%) часто были результатом решений, принятых при высокой скорости встречного препятствия, что приводило к столкновениям. FFStreams, последовательно предоставлял траектории-кандидаты для сохранения полосы движения или уступки скорости

переднего препятствия, за исключением 6% случаев, когда не было найдено ни одного плана в пределах максимального уровня планирования. FFStreams превзошел в обгоне перед встречным автомобилем в 44% экспериментов, по сравнению с 40% для MCTS-OPD и 37% для IRP.

(a)



(b)



(c)

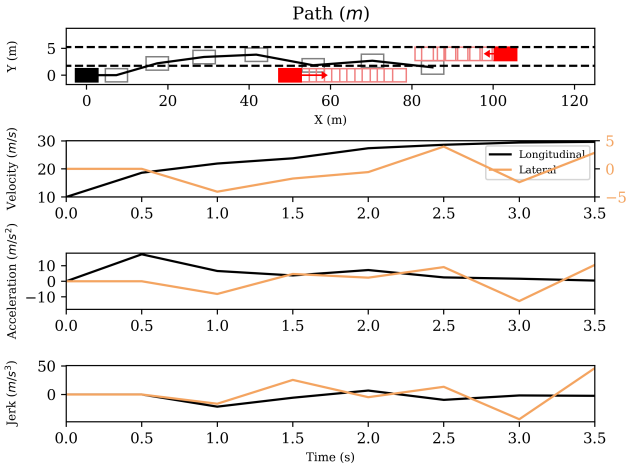


Рисунок 5.7 — (a) Успешный обгон, спланированный планировщиком FFStreams (жесткие параметры). (b) Успешный, но рискованный обгон был спланирован MCTS-OPD. В критическом сценарии скорость встречного препятствия составляет 10.7 m/s. (c) Успешный, но очень рискованный обгон был запланирован IRP, где скорость встречного препятствия составляет 6.0 m/s.

Таблица 5 — Результаты симуляций обгонов.

Method	Overtake	Yield &Overtake	Failure rate	Success rate	OPM
MCTS-OPD baseline	40%	42%	18%	82%	Extremely Aggressive Driver
IRP baseline	37%	25%	38%	62%	Extremely Aggressive Driver
FFStreams (hard params)	44%	50%	6%	94%	Extremely Aggressive Driver
FFStreams (soft params)	9%	83%	8%	92%	Normal Driver

По метрике OPM все три планировщика продемонстрировали крайне агрессивное поведение водителя. Таким образом, мы определяем планировщик FFStreams с мягкими параметрами оптимизации как самый безопасный, обеспечивающий обгон только тогда, когда это безопасно и удобно для пассажиров, достигая нормального поведения водителя по метрике OPM. На рисунке 5.7(a) представлен критический сценарий, в котором скорость эго составляет 7.5 m/s, а скорость встречного автомобиля — 4.47 m/s. Демонстрируется планируемая траектория с помощью FFStreams с жесткими параметрами, включая профили скорости, ускорения и рывка. Планируемые действия включают в себя сохранение полосы движения и ускорение для обгона переднего препятствия, когда существует безопасная траектория обгона без столкновений. Несмотря на экстремальное ограничение ускорения, запланированная оптимизированная траектория имеет максимальное абсолютное ускорение 4.3 m/s^2.

На рисунке 5.7 (b) запланированная MCTS-OPD траектория в критическом сценарии демонстрирует быстрое и небезопасное решение об обгоне с нереальными значениями, включая максимальное ускорение 15.0 m/s^2 и максимальный рывок 25.3 m/s^3, создавая тем самым высокий риск для пассажиров беспилотного автомобиля и других транспортных средств. На рисунке 5.7 (c) запланированная IRP траектория в критическом сценарии показывает непоследовательные и ненормальные решения, многократно чередующиеся между сменой левой и правой полос движения. Такие

решения предполагают высокий риск влияния на отношение водителей других транспортных средств. Кроме того, запланированная траектория демонстрирует нереалистичные значения ускорения и рывка, достигая максимального абсолютного ускорения в 15.0 m/s^2 и максимального абсолютного рывка в 52.5 m/s^3, что нецелесообразно для реальных приложений. Значения ускорения и рывка в MCTS-OPD и IRP существенно отличаются от поведения водителя-человека, что представляет угрозу безопасности. Кроме того, процесс принятия решений в этих методах отличается непоследовательностью, что может негативно повлиять на других водителей.

Сценарий 2. Многополосное шоссе

Чтобы показать эффективность нашего подхода, мы протестировали его на тех же экспериментах, которые были представлены в подходе CLF-CBF-QP [55], основанном на правилах, в условиях шоссе. Автономный автомобиль имеет начальную скорость $v_{ego}(0) = 29\ m/s$. Сценарий состоит из одного переднего препятствия, расположенного случайным образом $x_1(0) = [50\ m$,$65\ m]$, движущегося со случайной постоянной скоростью [26 m/s,32 m/s]. Кроме переднего препятствия, на соседней полосе в случайной координате x [$-85\ m$,$85\ m$] находятся четыре препятствия, движущиеся со случайными скоростями [26 m/s,32 m/s] и со случайными ускорениями [$-3\ m/s^2, 3\ m/s^2$], и дополнительное препятствие на третьей полосе, $x6(0) = [-85\ m$,$85\ m]$, которое переходит на соседнюю полосу в случайное время. Задача состоит в том, чтобы перестроиться в левую полосу. После проведения 50 экспериментов, в 88,00% экспериментов беспилотный автомобиль успешно сменил полосу движения. В 84,00% из них смена полосы произошла за 60 секунд (Таблица 6). Для сравнения, в методе CLF-CBF-QP этот показатель не превысил 56%. Мы также оценили траектории двух методов по метрике OPM [72]. Результаты показывают, что оба метода относятся к классификации типов движения общественного транспорта.

На рисунке 5.8 показан пример успешного эксперимента по смене полосы движения. В этом эксперименте одно препятствие меняет полосу движения, а планировщик меняет полосу движения, когда это безопасно. На рисунке 5.8 показаны профили скорости, ускорения и рывка; ускорение находится в диапазоне [$-0.5\ m/s^2, +0.5\ m/s^2$], а рывок — в диапазоне [-0.5 $m/s^3, +0.5\ m/s^3$], что обеспечивает комфортную езду. Результаты показывают, что планировщик FFStreams имеет более высокий процент успеха и превосходит

метод CLF-CBF-QP, основанный на правилах управления. Планировщик FFStreams работает с частотой ≈ 6 Hz.

Таблица 6 — Результаты симуляций смены полос движения.

Method	Lane-Change under 60s	OPM
FFStreams	84.00%	Public Transportation
CLF-CBF-QP	55.58%	Public Transportation

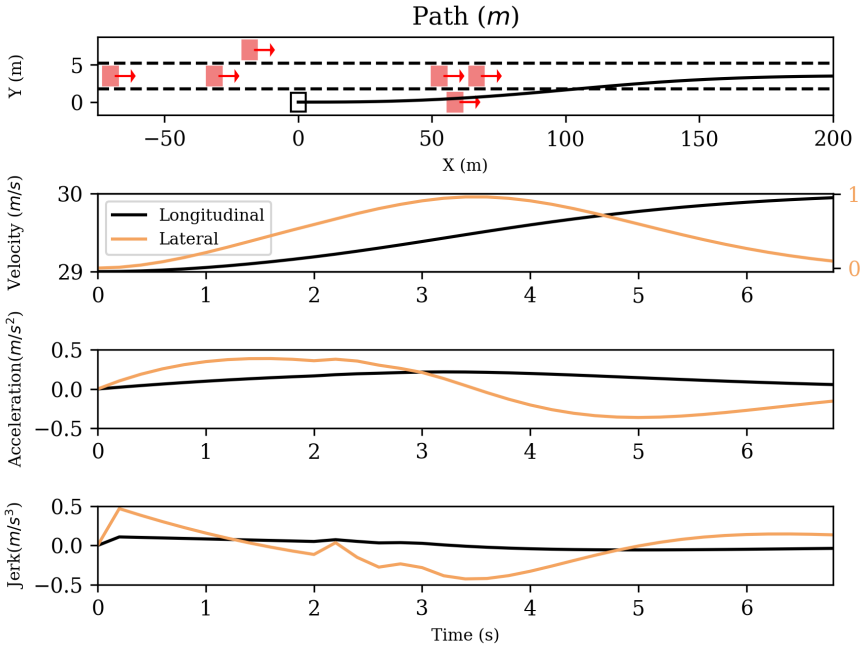


Рисунок 5.8 — Успешная смена полосы движения, когда беспилотный автомобиль меняет полосу движения, когда смена безопасна, и сливается с четырьмя препятствиями.

Три дополнительных сценария CommonRoad

Чтобы продемонстрировать эффективность нашего подхода, мы протестировали его на трех сценариях CommonRoad USA_US101-1_1_T-1, ESP_Monzon-2_1_T-1, ITA_Empoli-18_1_T-1, взятых из реальных ситуаций. Эти сценарии представляют собой различные проблемы, с которыми может столкнуться система автономного вождения. Первый включает в себя быструю смену полосы движения и слияние с быстро движущимися препятствиями. Второй требует следовать за замедляющимся автобусом, что требует корректировки скорости. В третьем случае необходимо следовать за ускоряющимся автомобилем, что требует адаптации скорости. Мы сравнили наш метод с планировщиком на основе поиска в CommonRoad, который

использует 2 697 примитивов движения и основан на библиотеке планирования на основе поиска (SBPL, Search-Based Planning Library)[73].

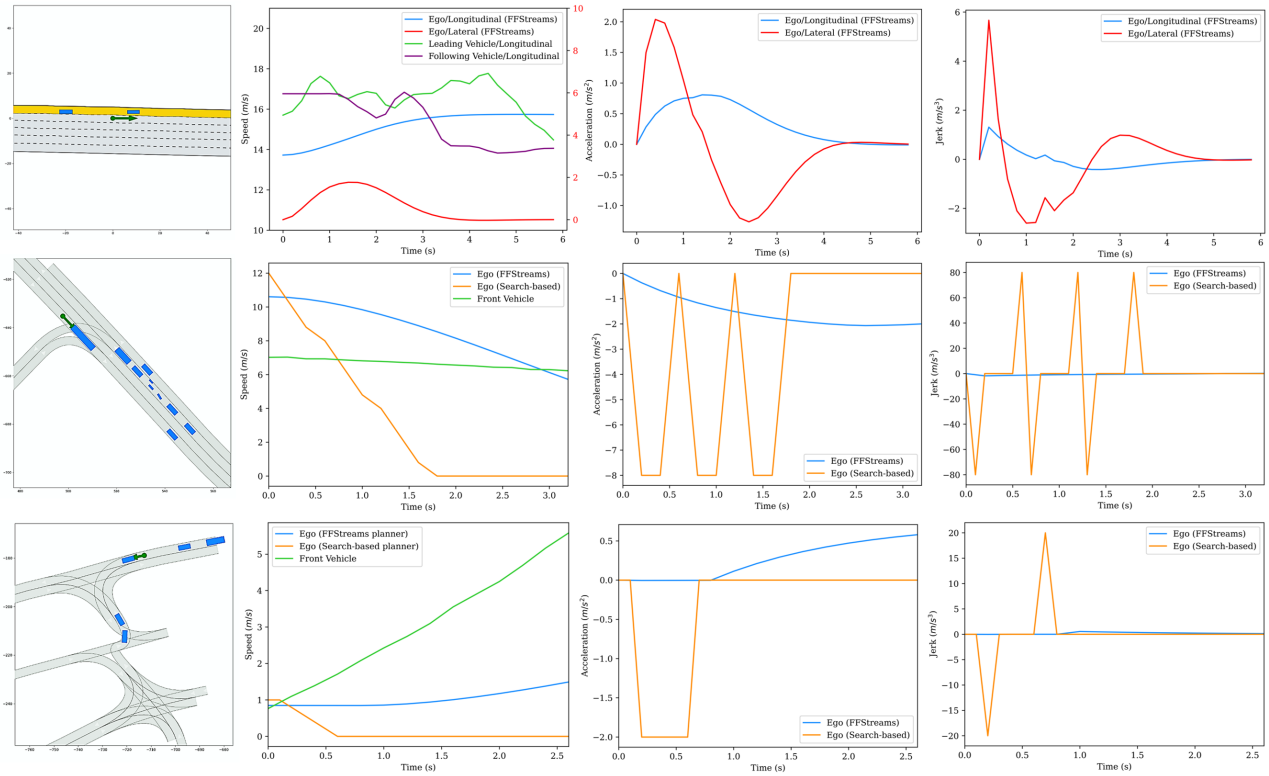


Рисунок 5.9 — Графики скорости, ускорения и рывка, сравнивающие FFStreams и планировщики на основе поиска в сценариях CommonRoad (USA_US101-1_1_T-1, ESP_Monzon-2_1_T-1, ITA_Empoli-18_1_T-1). Зеленые стрелки обозначают начальное положение беспилотного автомобиля, а синие прямоугольники — начальное положение препятствий.

На рисунке 5.9 видно, что FFStreams превосходит планировщик на основе поиска в различных сценариях. Во втором сценарии FFStreams избегает столкновения и регулирует скорость, чтобы следовать за автобусом, в то время как планировщик на основе поиска приводит к остановке беспилотного автомобиля. В последнем сценарии FFStreams ускоряется, чтобы следовать за лидирующим автомобилем, в то время как планировщик на основе поиска поддерживает нулевую скорость, что приводит к остановке беспилотного автомобиля. FFStreams превосходит по значениям ускорения и рывка, предлагая более комфортную траекторию. Его превосходство по критериям комфорта, безопасности и человекоподобности обусловлено способностью интегрировать семантические знания для человекоподобного поведения.

Чтобы оценить эффективность планирования, мы проанализировали среднее время работы в экспериментах с одиннадцатью препятствиями, сравнив

наш алгоритм с двумя базовыми: MCTS-OPD и IRP (рисунок 5.10). Хотя время работы увеличивается с ростом числа проверяемых препятствий, фокусировка на ограниченном числе значимых препятствий является практичной для выполнения автономных маневров обгона или смены полосы движения в различных сценариях.

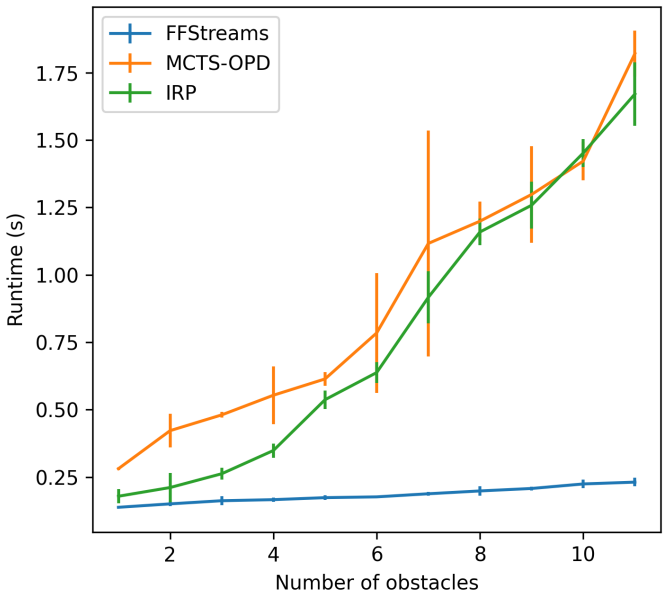


Рисунок 5.10 — Среднее время выполнения и стандартное отклонение для алгоритмов FFStreams, MCTS-OPD и IRP при наличии до 11 препятствий.

5.4 Заключение

В настоящей главе мы представили планировщик FFStreams, который демонстрирует эффективный подход к планированию маневров за счет интеграции планирования задач и движений, принятия решений на основе логики и кинематических ограничений. Эти элементы работают вместе, чтобы гарантировать, что система генерирует выполнимые и безопасные траектории, избегая при этом нежелательных или непрактичных результатов. В настоящее время планировщик поддерживает такие важные маневры, как удержание полосы движения, уступка, смена полосы движения и обгон, — все они жизненно важны для реальных операций с беспилотными автомобилями.

Ключевым преимуществом планировщика FFStreams является использование стандартного языка планирования, что облегчает сравнение с другими эвристическими поисковыми планировщиками. Эта особенность позволяет проводить сравнительный анализ его работы в различных сценариях, что дает ценные сведения о его сильных сторонах и областях, требующих улучшения.

Заглядывая вперед, можно выделить несколько направлений дальнейшей работы. Одним из перспективных направлений является расширение возможностей планировщика для работы с более сложными условиями движения, в частности, с разнообразными сценариями пересечения дорог. Планировщик FFStreams предлагает надежную основу для планирования маневров в беспилотных автомобилях, эффективно балансируя между логическим принятием решений и кинематическими ограничениями в реальном времени. Будущие разработки должны быть направлены на расширение сферы его применения и улучшение характеристик в более сложных условиях, что обеспечит его актуальность в условиях развития технологий беспилотных автомобилей.

Глава 6. Быстрый эвристический поиск с предсказанием траектории

В настоящей главе рассматривается интеграция быстрого эвристического поиска с моделями предсказания траекторий для улучшения возможностей принятия решений и планирования движения беспилотных автомобилей. Возрастающая сложность среды автономного вождения, особенно в динамических и многоагентных сценариях, требует применения передовых методов, позволяющих прогнозировать будущее поведение окружающих агентов и оптимизировать траекторию движения беспилотного автомобиля. Комбинируя эвристический поиск с предсказанием траектории, система планирования может предвидеть потенциальные риски и возможности, тем самым повышая общую эффективность и безопасность автономной навигации.

Центральное место в данной главе занимает представление модели предсказания траекторий QCNet, основанной на глубоком обучении и предназначенной для предсказания будущих траекторий движения транспортных средств с высокой точностью. QCNet играет важную роль в информировании системы планирования о возможных будущих движениях других агентов, позволяя автомобилю принимать обоснованные и упреждающие решения. Эта модель особенно ценна в условиях, когда быстрые и точные предсказания необходимы для предотвращения столкновений и безопасной навигации по сложным дорожным сценариям.

В дополнение к QCNet мы представляем FFStreams++ Decision-Making and Motion Planning Framework, улучшенную версию планировщика FFStreams, рассмотренного в предыдущей главе. FFStreams++ использует предсказания траекторий из QCNet для улучшения процесса принятия решений, интегрируя эти предсказания в эвристический поиск для определения оптимальных маневров с учетом будущих взаимодействий транспортных средств. Цель фреймворка — повысить производительность планировщика в реальном времени за счет эффективного сужения пространства поиска с использованием как эвристических методов, так и предсказаний.

Настоящая глава структурирована следующим образом. В разделе 6.1 представлена постановка задачи. В разделе 6.2 описываются методы, используемые для предсказания траектории и интеграции эвристического поиска, а затем дается подробная оценка системы с помощью серии

экспериментов и результатов (раздел 6.3). Мы оцениваем производительность подхода с помощью бенчмарка CommonRoad, а также в различных сценариях на перекрестках и шоссе, которые проверяют устойчивость системы в различных условиях вождения. В разделе 6.4 анализируются результаты, показывающие эффективность сочетания предсказания траектории с быстрым эвристическим поиском в реальных приложениях для автономного вождения. Наконец, в разделе 6.5 приводятся выводы.

6.1 Постановка задачи

Входные данные включают в себя конфигурацию беспилотного автомобиля $q(t) = [x, y, \theta, v, a]$, представляющую положение, угол поворота, скорость и ускорение автомобиля, а также состояния m динамических препятствий $B_i : i \in \{1, \ldots, m\}$, которые описываются их положениями, углами поворота и скоростями $[x_{bi}, y_{bi}, \theta_{bi}, v_{bi}]$. Кроме того, входные данные включают в себя информацию карты $\mathcal{M} \in M$, такую как опорные линии и полосы движения. Время T дискретизируется на Δt. Возможные действия определяются множеством $A = \{a_1, ..., a_n\}$, где $n = 5$, а действия включают в себя "$Yield$" — уступить дорогу другому препятствию и замедлиться, "$Follow\ speed$" — следовать заданной скорости и ускориться, "$Change\ to\ left\ lane$" — перестроиться в левую полосу, "$Change\ to\ right\ lane$" — перестроиться в правую полосу, "$Overtake$" — обогнать препятствие впереди. Множество выборок потока $St = \{st_1, ..., st_k\}$, где $k = 5$, а выборки потока включают в себя поток для каждого маневра, уступить дорогу впереди едущему препятствию или на перекрестке, следить за скоростью, перестроение в левую полосу, перестроение в правую полосу и обгон.

Учитывая начальную конфигурацию q_0 и целевую область G, задача состоит в том, чтобы найти выполнимый план $\pi = [a_0, a_1, \ldots, a_T]$, используя планировщик на основе эвристического поиска и выборки $D : Q \times B \times M \to A$. Следуя плану π, цель G должна быть достигнута из начальной конфигурации q_0, образуя траекторию $\tau = (q_0, q_1, \ldots, q_g)$. Время выполнения плана должно быть минимизировано, при этом траектория $\tau(t)$ не должна приводить к столкновению с препятствием, $\tau(t) \cap \forall b_i(t) \in B = \varnothing$.

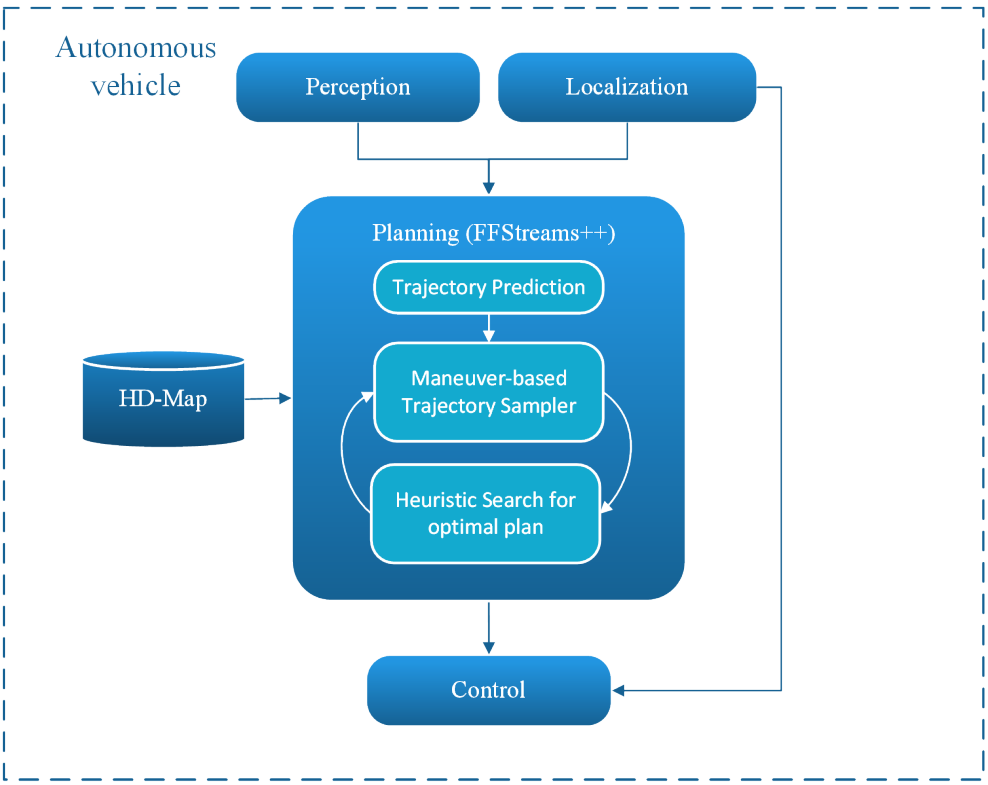


Рисунок 6.1 — Схема комбинированного принятия решений и планирования движения с предсказанием траектории.

Предложенный метод (рисунок 6.1) предназначен для планирования автономных маневров обгона, смены полосы движения, уступки, проезда перекрестков и незащищенных левых поворотов. Эти маневры являются критическими для автономного вождения, поскольку небольшая ошибка в принятии решения, планировании или предсказании траектории может привести к потенциально опасным ситуациям, включая столкновения и небезопасные маневры. На рисунке 6.2 показан критический маневр незащищенного левого поворота на несигнализированном перекрестке, когда намерения встречного транспортного средства неясны и маневр сопряжен с высоким риском. Рисунок 6.3 иллюстрирует критический маневр желательного обгона, когда впереди идущий автомобиль движется на низкой скорости, а другие окружающие автомобили — на высокой.

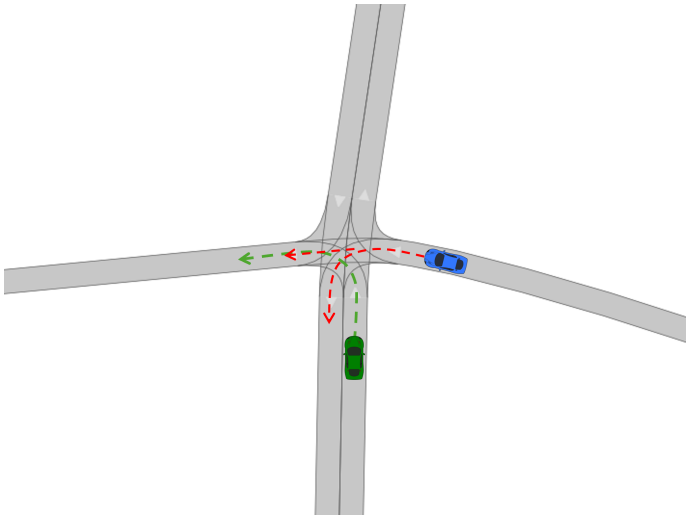


Рисунок 6.2 — Критический незащищенный маневр левого поворота на несигнализированном перекрестке.

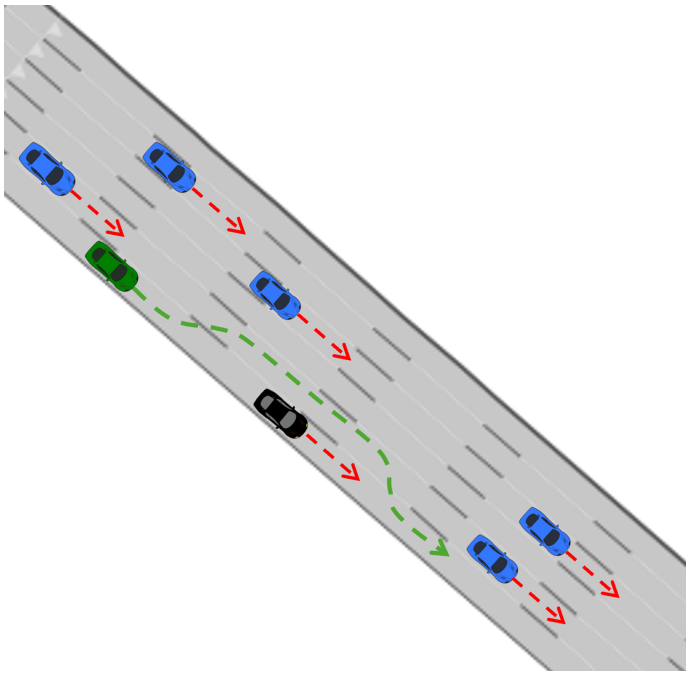


Рисунок 6.3 — Критический маневр обгона в сценарии движения по шоссе.

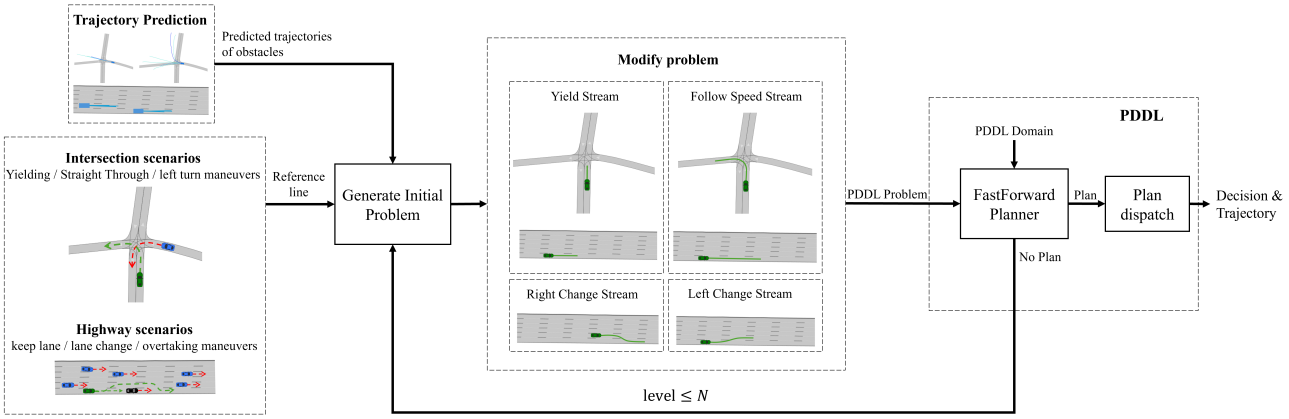


Рисунок 6.4 — Схема предлагаемого фреймворка FFStreams++ для интегрированного принятия решений и планирования движения с предсказанием траектории при автономном вождении в динамических средах. Система сочетает предсказание траектории с целевой опорной линией, полученной из сценариев проезда перекрестков и шоссе, для создания исходной задачи PDDL. Эта задача итеративно модифицируется путем рассмотрения различных потоков траекторий (Yield, Follow Speed, Right Change и Left Change Streams) до достижения оптимального плана. Задача формулируется в PDDL2.1, решается с помощью FastForward Planner, и полученный оптимальный план отправляется для принятия решений и управления траекторией движения автомобиля.

6.2 Методы

Мы предлагаем фреймворк FFStreams++ для решения интегрированной задачи принятия решений и планирования движения (рисунок 6.4). Мы интегрируем в FFStreams++ сеть предсказания траектории Query-Centric Network (QCNet). В следующих подразделах будет подробно описана каждая часть системы принятия решений и планирования движения (фреймворк) FFStreams++ и также модель предсказания траектории QCNet.

6.2.1 Модель предсказания траектории QCNet

Для предсказания траекторий движения окружающих автомобилей была использована сеть Query-Centric Network (QCNet)[20], предварительно обученная на бенчмарке Argoverse 2 motion forecasting. Множество сценариев, извлеченных из бенчмарка CommonRoad, находится в пределах компетенции предварительно обученной нейронной сети.

На вход сети подается информация о карте и состоянии препятствий на последних T временных шагах. Состояние включает в себя двумерное положение препятствий и их направление. Информация о карте включает в себя полигоны карты высокой четкости (например, полосы движения и переходы), где каждый полигон аннотирован точками сэмплирования и семантическими атрибутами (например, тип полосы движения). QCNet предсказывает будущие траектории K каждого препятствия на горизонте предсказания T' и выдает оценку вероятности для каждой траектории.

QCNet была настроена путем преобразования информации о карте из формата Lanelet2 в формат карты Argoverse HD и состояния агентов.

Наша модель предсказания была обучена предсказывать шесть возможных будущих траекторий для каждого окружающего препятствия и их вероятности. В планировщик FFStreams++ последовательно вводились две предсказанные траектории с наибольшей и наибольшей вероятностью для каждого препятствия.

Выбор горизонтов предсказания и планирования для алгоритма автономного вождения предполагает баланс нескольких факторов для обеспечения оптимальной работы. При выборе горизонта предсказания следует учитывать такие факторы, как сложность дорожной обстановки (город, шоссе), скорость автомобиля и возможности модели. Более длинные горизонты предсказания увеличивают неопределенность и могут снизить надежность предсказания. Короткие горизонты могут привести к потере ценной информации об окружающих автомобилях. При выборе горизонта планирования следует учитывать такие факторы, как сложность маршрута, вычислительные ресурсы и время отклика. Более длинные горизонты планирования обеспечивают более плавную и комфортную езду, но могут

быть менее отзывчивы к внезапным изменениям, поскольку требуют больших вычислительных мощностей и могут увеличивать время ожидания.

Мы предсказываем будущие траектории препятствий на горизонте предсказания в 5 секунд, который был выбран после оценки метрики Root Mean Squared Error (RMSE) на различных горизонтах предсказания (рисунок 6.5) в различных сценариях на шоссе CommonRoad. RMSE измеряет расстояние L_2 в метрах между истинной траекторией и наилучшей из предсказаний, усредненное по всем будущим временным шагам.

Мы также выбрали горизонт планирования в 5 секунд после изучения времени работы алгоритма, комфорта водителя и гладкости траекторий.

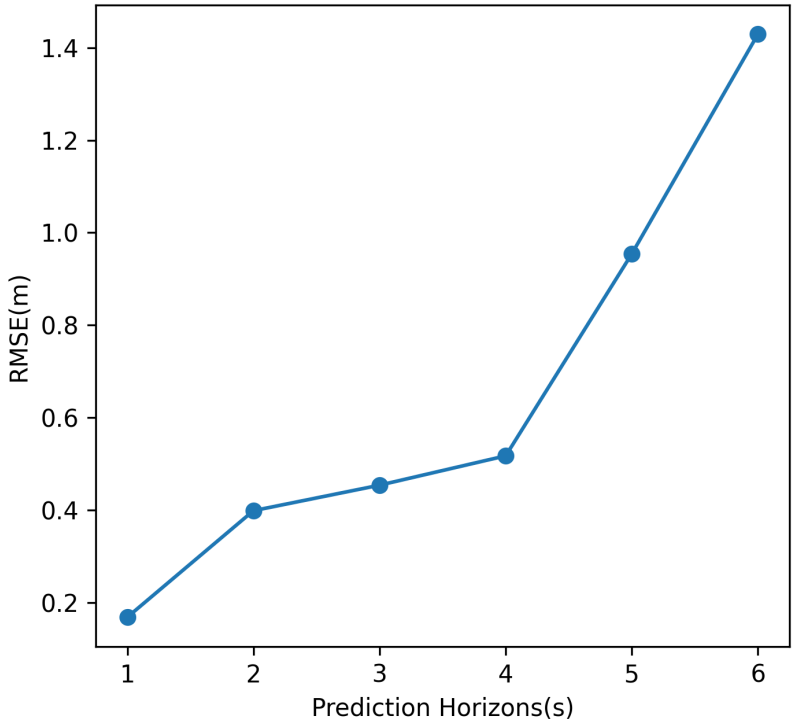


Рисунок 6.5 — Метрика RMSE модели предсказания для различных горизонтов предсказания.

6.2.2 Система принятия решений и планирования движения FFStreams++

Фреймворк FFStreams++ адаптирован для планирования маневров незащищенного левого поворота в дополнение к маневрам удержания в полосе, смены полосы движения и обгона, а также для учета более точных предсказаний траекторий окружающих препятствий для улучшения планируемого поведения и движения с учетом ускорений препятствий.

Фреймворк FFStreams++ представляет домен и задачу планирования на языке PDDL, редактирует задачу планирования с помощью потоков и ищет оптимальный план с помощью эвристического поиска FastForward. Потоки используются для генерации оптимизированной траектории на Frenet для каждого маневра, а также связанных с ними предикатов, которые будут добавлены к задаче на языке PDDL.

Домен PDDL и поиск FastForward

Мы формулируем задачу принятия решений и планирования движения как домен планирования PDDL2.1 Σ (представление предикатов, их типов и возможных действий) и задачу планирования PDDL2.1 $\mathcal{P}$ (представление объектов, предикатов начального состояния, предикатов цели, метрики оптимизации плана). Мы используем расширение PDDL2.1 для числового планирования, чтобы представлять различные предикаты числовыми значениями, а также для выполнения арифметических операций в PDDL для вычисления расстояния до препятствий при проверке столкновений.

В домене PDDL мы определяем два типа объектов: конфигурация автомобиля $-conf$ и препятствие $-obstacles$. Мы определяем множество предикатов: *(yield_traj* $?q_1$ $?q_2$*)*,*(keep_speed_traj* $?q_1$ $?q_2$*)*,*(overtake_traj* $?q_1$ $?q_2$*)*,*(left_traj* $?q_1$ $?q_2$*)*, и *(right_traj* $?q_1$ $?q_2$*)*, указывающие на существование траектории от конфигурации q_1 к конфигурации q_2 для уступки (замедления), следования со скоростью (ускорения), обгона, перехода на левую полосу или перехода на правую полосу, *(next* $?q_1$ $?q_2$ $?q_{end}$*)* указывает на

последовательность подконфигураций на траектории к конечной конфигурации q_{end}, *idle* указывает, что процесс проверки столкновений простаивает, чтобы начать новую проверку, *(checking_traj ?q_1 ?q_2 ?o)*, указывающая, что траектория между конфигурациями q_1 и q_2 находится на проверке с препятствием *o*, *(checked_traj ?q_1 ?q_2 ?o)* указывает, что траектория между двумя конфигурациями проверена препятствием *o* и свободна от столкновений, *(ego_at ?q)* указывает, что беспилотный автомобиль в данный момент находится в конфигурации *?q*, *(on_init_lane)*, *(on_second_lane)* указывает текущую полосу движения беспилотного автомобиля.

Функции домена планирования маневров включают следующие изменяющиеся переменные: *(total_cost)* — общая стоимость, *(curr_time)* — текущее время, *(time_of_traj ?q_1 ?q_2)* — продолжительность следования по траектории, *(at_x ?q) (at_y ?q) (at_time ?q)* — координаты и шаг по времени определенной конфигурации, *(obst_at_x ?o ?q) (obst_at_y ?o ?q)* — координаты определенного препятствия при определенной конфигурации беспилотного автомобиля, неявно связывая координаты препятствия с определенным шагом по времени.

Мы определяем множество типов T, объектов B, предикатов P и функций F следующим образом:

$$T = \{\text{"conf "obstacles"}\}.$$

$$B = B_{\text{conf}} \cup B_{\text{obstacles}}.$$

$$B_{\text{conf}} = \{q_1, q_2, \ldots, q_n\}.$$

$$B_{\text{obstacles}} = \{o_1, o_2, \ldots, o_m\}.$$

$$P = \{next(q_i,q_j), traj(q_i,q_j), idle(), is\_first(q_i,o_j),$$

$$is\_last(q_i,o_k), ego\_at(q_i), moved\_forward(),$$

$$checking\_traj(q_i,q_j,o_k), on\_init\_lane(),$$

$$checked\_traj(q_i,q_j,o_k), on\_second\_lane(),$$

$$there\_is\_front\_obs(), yield\_traj(q_i,q_j),$$

$$keep\_speed\_traj(q_i,q_j), overtake\_traj(q_i,q_j),$$

$$left\_traj(q_i,q_j), right\_traj(q_i,q_j)\}.$$

$$F = \{cost(), curr\_time(), time\_of\_traj(q_i,q_j),$$

$$at\_x(q_i), at\_y(q_i), at\_time(q_i),$$
$$obst\_at\_x(o_i, q_j), obst\_at\_y(o_i, q_j)\}.$$

Домен маневров состоит из множества действий A для маневров: *keep_speed* действие для сохранения полосы при ускорении, *keep_yield_speed* действие для сохранения полосы при замедлении, *left_change*/*right_change* действия для перемещения на соседнюю полосу, *overtake* действие для обгона переднего препятствия, если оно существует. Мы определяем множество действий A следующим образом:

$$A = \{keep\_speed(\bar{o}), keep\_yield\_speed(\bar{o}),$$
$$left\_change(\bar{o}), right\_change(\bar{o}), overtake(\bar{o})\}.$$

Ниже показан частичный домен PDDL.

```
(:predicates
    (next ?q1 ?q2 ?q_end - conf)
    (traj ?q1 ?q2 - conf)
    (idle)(is_first ?q - conf ?o - obstacles)
    (is_last ?q - conf ?o - obstacles)
    (ego_at ?q - conf)
    (checking_traj ?q1 ?q2 - conf ?o - obstacles)
    (checked_traj ?q1 ?q2 - conf ?o - obstacles)
    (moved_forward)
    (there_is_front_obs)
    (on_init_lane)(on_second_lane)(yield_traj ?q1 ?q2)
    (keep_speed_traj ?q1 ?q2)
    (overtake_traj ?q1 ?q2)
    (left_traj ?q1 ?q2)
    (right_traj ?q1 ?q2))
(:functions (cost)(curr_time)
    (time_of_traj ?q1 ?q2 - conf)
    (at_x ?q - conf)(at_y ?q - conf)
    (at_time ?q - conf)
    (obst_at_x ?o - obstacles ?q - conf)
    (obst_at_y ?o - obstacles ?q - conf)
)
```

После определения домена планирования и задачи планирования $(\Sigma, \mathcal{P})$ эвристический планировщик FastForward ищет план с минимальной стоимостью. Эвристика FF h^{FF} оперирует облегченной (relaxed) версией задачи планирования, в которой эффекты удаления действий игнорируются. Это означает, что если факт (предикат) становится истинным, то он остается истинным, что упрощает задачу. Эвристика FF строит граф расслабленного планирования (RPG, Relaxed Planning Graph) — многослойный граф,

```
(:action keep_speed
    :parameters (?q1 ?q2 - conf)
    :precondition (and
         (ego_at ?q1)
         (traj ?q1 ?q2)
         (keep_speed_traj ?q1 ?q2)
         (forall (?o - obstacles)
            (checked_traj ?q1 ?q2 ?o)
         )
         (on_init_lane)
         (idle)
       )
    :effect (and
         (ego_at ?q2)
         (not (ego_at ?q1))
         (increase (curr_time)  (time_of_traj ?q1 ?q2))
         (increase (cost)  5)
         (moved_forward)
        )
)
```

где каждый слой представляет собой множество фактов или действий, которые могут быть достигнуты или применены на данном шаге без учета эффектов удаления. Как только предикаты цели присутствуют в слое фактов, алгоритм прослеживает граф от целей назад, выбирая действия, которые достигают целей. Сумма затрат на эти действия дает эвристическую оценку. Руководствуясь этими эвристическими оценками, FF использует взвешенную стратегию поиска A* для исследования пространства состояний и поиска решения. Временные сложности для построения графа облегченного планирования и вычисления эвристики являются полиномиальными.

```
(:action left_change
    :parameters (?q1 ?q2 - conf)
    :precondition (and
         (ego_at ?q1)
         (traj ?q1 ?q2)
         (left_traj ?q1 ?q2)
         (forall (?o - obstacles)
            (checked_traj ?q1 ?q2 ?o)
         )
         (on_init_lane)
         (idle)
               )
    :effect (and
         (ego_at ?q2)
         (not (ego_at ?q1))
         (not (on_init_lane))
         (increase (cost)  1)
         (increase (curr_time)  (time_of_traj ?q1 ?q2))
         (moved_forward) )
        )
```

```
(:action keep_lane_yield
    :parameters (?q1 ?q2 - conf)
    :precondition (and
         (ego_at ?q1)
         (traj ?q1 ?q2)
         (yield_traj ?q1 ?q2)
         (forall (?o - obstacles)
            (checked_traj ?q1 ?q2 ?o)
         )
         (on_init_lane)
         (idle)
       )
    :effect (and
         (ego_at ?q2)
         (not (ego_at ?q1))
         (increase (curr_time)  (time_of_traj ?q1 ?q2))
         (increase (cost)  10)
         (moved_forward)
        )
)
```

Маневрирование потоками с оптимизацией френета и рывка

Мы используем поток для каждого маневра, скорости следования, уступки передним препятствиям или на перекрестке, перехода на левую соседнюю полосу и перехода на правую соседнюю полосу. Применимые потоки будут вызываться на каждой итерации алгоритма FFStreams++. Поскольку каждый поток имеет свои условия, не все из них применимы. Например, поток overtake_trajectory применим только при наличии переднего препятствия.

```
(:stream yield_stream
    :inputs (?q1)
    :domain (and (at_x ?q1) (at_y ?q2) (at_time ?q1))
    :outputs (?q2 ?q_{1_2_1} ... ?q_{1_2_24})
    :certified
        (and (yield_traj ?q1 ?q2)
        (time_of_traj ?q1 ?q2)
        (at_x ?q2) (at_y ?q2) (at_time ?q2)
        (next ?q1 ?q1_2_1 ?q2) ...
        (next ?q1_2_23 ?q1_2_24 ?q2)))
```

Для каждого потока маневров мы определяем $v_{desired}$ — желаемую конечную скорость траектории в соответствии с типом маневра. Чтобы сгенерировать траекторию-кандидат для каждого маневра, мы используем оптимизацию Френе для планирования траектории и интегрируем оптимизацию рывка в функцию стоимости для выбора эффективной и удобной траектории.

```
(:stream overtake_stream
    :inputs (?q1)
    :domain (and (at_x ?q1) (at_y ?q2) (at_time ?q1)
        (there_is_front_obs))
    :outputs (?q2 ?q_{1_2_1} ... ?q_{1_2_24})
    :certified
        (and (overtake_traj ?q1 ?q2)
        (time_of_traj ?q1 ?q2)
        (at_x ?q2) (at_y ?q2) (at_time ?q2)
        (next ?q1 ?q1_2_1 ?q2) ...
        (next ?q1_2_23 ?q1_2_24 ?q2)))
```

Мы определяем опорную траекторию как центральную линию желаемой полосы движения и начальное состояние беспилотного автомобиля как $[x_0,y_0,\theta_0,v_0,a_0]$, где $(x_0,y_0),\theta,v_0,a_0$ — это начальные декартовы координаты, угол поворота, скорость и ускорение беспилотного автомобиля. Затем мы преобразуем декартову систему координат в систему координат Френе, определяя состояние в системе Френе как:

$$x(t) = [s(t),s'(t),s''(t),s'''(t),l(t),l'(t),l''(t),l'''(t)] \tag{6.1}$$

где s — продольное положение, s' — продольная скорость, s'' — продольное ускорение, s''' — продольный рывок, l — поперечное положение, l' — поперечная скорость, l'' — поперечное ускорение и l''' — поперечный рывок.

При известном начальном состоянии Френе $[s_0,s'_0,s''_0,l_0,l'_0,l''_0]$ генерируется множество поперечных и продольных траекторий с использованием квинтовых полиномов для поперечного движения и квартовых полиномов для продольного движения. Из множества исключаются траектории, превышающие максимальное ускорение a_{max}, максимальную скорость v_{max} или максимальную кривизну κ. Затем для каждой траектории τ оценивается общая стоимость J, представленная в виде:

$$\begin{aligned} J(\tau) = J_{comfort}(\tau) + J_{efficiency}(\tau) + J_{lateral\_error}(\tau) \\ + J_{speed\_error}(\tau), \end{aligned} \tag{6.2}$$

где $J_{comfort}$ — затраты, связанные с комфортом пассажиров, включая минимизацию рывков для штрафования траекторий с большими значениями рывков, $J_{efficiency}$ — затраты, связанные со временем в пути, $J_{lateral\_error}$ — затраты, связанные с боковой ошибкой в конечной точке, и J_{speed_error} — затраты, связанные с ошибкой продольной скорости в конечной точке. Мы определяем ранее упомянутые затраты как:

$$J_{comfort} = \omega_j \int_0^T (s'''^2 + l'''^2)dt \tag{6.3}$$

$$J_{efficiency} = \omega_t.T \tag{6.4}$$

$$J_{lateral\_error} = \omega_{err}.(l(T) - l_{desired})^2 = \omega_{err}.e_l^2 \tag{6.5}$$

$$J_{speed\_error} = \omega_{err}.(s'(T) - s'_{desired})^2 = \omega_{err}.e_{s'}^2, \tag{6.6}$$

где ω_j,ω_t,ω_{err} — весовые коэффициенты для рывка, времени движения и стоимости ошибки, T — время движения, e_l — боковая ошибка, $e_{s'}$ — ошибка продольной скорости, $s'_{desired}$ — желаемая скорость в конечной точке, а $l_{desired}$ — желаемое боковое положение в конечной точке. Поскольку целью является следование по эталонной траектории, $l_{desired}$ принимается равным нулю. Параметры оптимизации указаны в таблице 7.

Таблица 7 — Параметры оптимизации френета и рывка

Parameter	Symbol	Value
Jerk weight	ω_j	0.1
Travel time weight	ω_t	0.1
Error weight	ω_{err}	1.0
Planning horizon	T	5.0 $[s]$
Time step	Δt	0.2 $[s]$
Maximum acceleration	a_{max}	2.0 $[m/s^2]$
Maximum speed	v_{max}	57.6 $[m/s]$
Maximum Curvature	κ	1 $[1/m]$

Общая стоимость составляет:

$$J(\tau) = \omega_j \int_0^T (s'''^2 + l'''^2)dt + \omega_t.T + \omega_{err}.(e_l^2 + e_{s'}^2). \tag{6.7}$$

Оптимальная траектория τ^* — это траектория, которая минимизирует общую стоимость:

$$\tau^* = argmin_\tau J(\tau). \tag{6.8}$$

6.3 Эксперименты и результаты

Мы протестировали предложенную схему принятия решений и планирования движения беспилотных автомобилей в сложных сценариях проезда перекрестков и обгона, чтобы оценить ее. На перекрестках существует несколько вариантов принятия решений, что повышает риск потенциальных конфликтов. При проезде перекрестков с возможными поворотами налево и направо и движении прямо, беспилотный автомобиль должен решить, двигаться ли ему дальше и проехать перекресток или уступить дорогу/остановиться для проезда других автомобилей. Решения должны быть точными и безопасными, что приведет к безопасной траектории движения без столкновений. В сценариях на шоссе, где окружающие препятствия движутся с очень высокой скоростью, а также при наличии медленно движущихся передних препятствий, планировщик должен следовать обычному поведению водителя: обгонять, когда обгон переднего препятствия безопасен.

Для оценки эффективности автономного автомобиля мы используем метрики безопасности и комфорта для пассажиров, где метрика безопасности связана с количеством экспериментов без столкновений, а комфорт определяется метрикой OPM [72], учитывающей комфорт на основе максимального поперечного и продольного ускорения и рывка. Мы сравнили наш метод с планировщиком на основе поиска в CommonRoad, который использует 2697 примитивов движения и основан на Search-Based Planning Library (SBPL)[73].

6.3.1 Бенчмарк CommonRoad

CommonRoad представляет собой ценную платформу для оценки и сравнительного анализа эффективности алгоритмов автономного вождения. Его всеобъемлющее и стандартизированное множество сценариев охватывает широкий спектр сценариев, включая городские перекрестки, шоссе и дороги, предназначенных для тестирования различных аспектов принятия решений и планирования движения.

Мы используем две широко распространенные карты CommonRoad: несигнализированный перекресток (DEU_Nuremberg-39, рисунок 6.2); и шоссе (USA_US101-22, рисунок 6.3). Кроме того, мы провели эксперименты в симуляторе CommonRoad, случайным образом генерируя препятствия, чтобы охватить широкий спектр сложных ситуаций, включая незащищенные левые повороты на перекрестках, удержание в полосе движения и обгон на шоссе.

6.3.2 Сценарии перекрестков

Мы проводим эксперименты в симуляторе CommonRoad на карте DEU_Nuremberg-39 (рисунок 6.2) с несигнальным перекрестком. Начиная с начальной позиции эгомобиля, показанной на рисунке, беспилотный автомобиль (зеленый автомобиль) может выполнить два возможных маневра: левый поворот, при котором он поворачивает налево с юга на запад, и прямой путь, при котором он проезжает перекресток прямо через него. С другой стороны, препятствие (синий автомобиль) выполняет левый поворот с востока на юг с различными возможными скоростями и начиная со случайной начальной позиции.

Таблица 8 — Результаты симуляций перекрестков

Method	Go straight Success rate	Left turn Success rate	OPM
FFStreams++ planner	89%	84%	Normal Driver
Search-based planner	75%	53%	Aggressive Driver

В сценариях перекрестков препятствие имеет начальное случайное положение на полосе движения и движется с высокой скоростью, следуя профилю скорости и ускорения, взятому из реалистичного сценария CommonRoad "DEU_Nuremberg-39_5_T-1". Беспилотный автомобиль имеет начальное положение [360.51,-30.79] и движется со случайной начальной скоростью [3.5 m/s, 11.5 m/s]. После проведения 100 экспериментов (таблица 8) по двум типам сценариев (поворот налево и движение прямо) в 89.00% экспериментов по сценарию «движение прямо» беспилотный автомобиль

успешно проехал прямо и миновал перекресток. В 84.00% экспериментов по сценарию «левый поворот» беспилотный автомобиль успешно выполнил незащищенный левый поворот. С другой стороны, планировщик, основанный на поиске, имел 75.00% успеха в экспериментах по сценарию «движение прямо» и только 53.00% в экспериментах по сценарию «левый поворот». Полученные результаты демонстрируют превосходство FFStreams++ над планировщиками, основанными на поиске. После оценки эффективности двух методов по метрике OPM траектории FFStreams++ были классифицированы как «нормальный водитель», а траектории на основе поиска — как «агрессивный водитель», что доказывает лучшую эффективность FFStreams++ и его близость к человекоподобному поведению водителя.

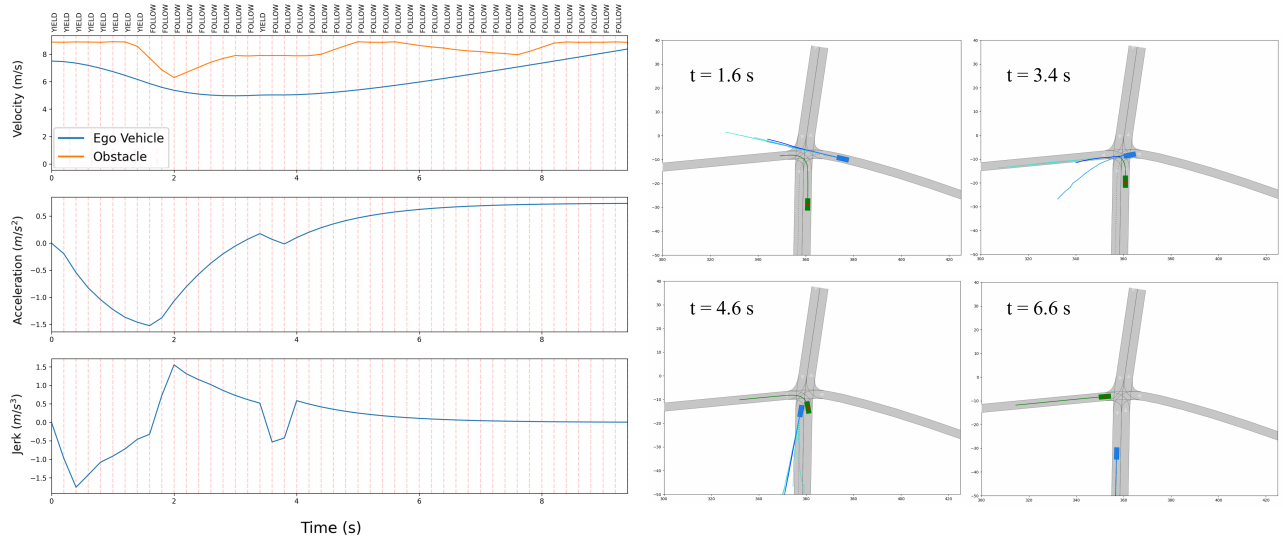


Рисунок 6.6 — Успешный эксперимент по планированию незащищенного левого поворота с помощью планировщика FFStreams++. Показаны профили скоростей, ускорений и рывков планируемых траекторий, а также решения на каждом временном интервале.

На рисунке 6.6 показан успешный эксперимент с незащищенным левым поворотом. В этом сценарии FFStreams++ уступает при приближении к перекрестку, и при более четком намерении окружающего автомобиля, представленном его предсказанием траектории, планировщик FFStreams++ принимает решение ускориться и выполнить левый поворот. На рисунке также показаны профили скорости, ускорения и рывка, а также решения, запланированные планировщиком FFStreams++, которые соответствуют ограничениям на ускорение и имеют оптимальные рывки, что приводит

к комфортной поездке для пассажиров. Планировщики на основе поиска, напротив, не смогли спланировать будущую безопасную траекторию.

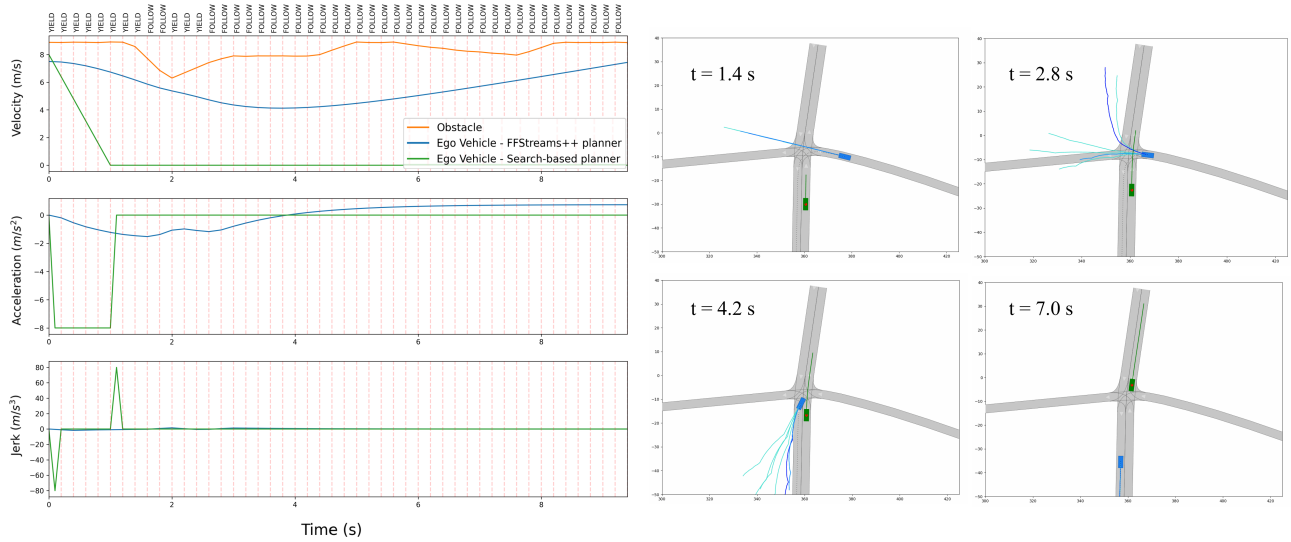


Рисунок 6.7 — Успешный эксперимент по проезду перекрестка, спланированный планировщиком FFStreams++. Демонстрируются профили скорости, ускорения и рывка планируемых траекторий, а также решение на каждом временном шаге.

На рисунке 6.7 показан успешный эксперимент по проезду перекрестков. В этом сценарии FFStreams++ снижает скорость (yield) при приближении к перекрестку, и при более четком намерении окружающего автомобиля, представленном его предсказанием траектории, планировщик FFStreams++ принимает решение ускориться и проехать перекресток. На рисунке также показаны профили скорости, ускорения и рывка, а также решения, запланированные планировщиком FFStreams++, которые придерживаются ограничений на ускорение и имеют оптимальные рывки, что делает езду комфортной для пассажиров. На рисунке также показана траектория, запланированная планировщиком Search-based, которая включает в себя остановку эгомобиля во избежание столкновения. Такое поведение далеко от нормального поведения водителя. С другой стороны, траектория FFStreams++ является плавной, а решения близки к реальному поведению человека при вождении.

6.3.3 Сценарии на шоссе

Мы проводим эксперименты в симуляторе CommonRoad на карте USA_US101-22 (рисунок 6.3). Сценарий состоит из четырех препятствий: 1) переднее препятствие на одной полосе с беспилотным автомобилем, движущееся с низкой скоростью $3.96m/s$ и следующее за профилями скорости и ускорения, полученными из одного из сценариев CommonRoad; 2) препятствие на левой соседней полосе, находящееся позади беспилотного автомобиля на случайном расстоянии $[25m, 50m]$, движущееся со случайной скоростью $[10m/s, 14m/s]$ и следующее профилю ускорения, извлеченному из одного из сценариев CommonRoad; 3) препятствие на левой соседней полосе, далеко впереди беспилотного автомобиля, вне зоны его интереса; 4) препятствие на левой соседней полосе, впереди автономного автомобиля, вне зоны интереса беспилотного автомобиля. В сценарии начальное положение беспилотного автомобиля по оси XY равно [22.041, − 18.688], его начальная случайная скорость равна $[12m/s, 14m/s]$, а начальное ускорение равно нулю.

Таблица 9 — Результаты симуляций шоссе

Method	Overtaking Success rate	Overtaking(after Waiting for Obstacle2 to pass)	OPM
FFStreams++ planner	92%	47%	Normal/Aggressive Driver
Search-based planner	34%	16%	Aggressive Driver

После проведения 100 экспериментов со сценариями обгона (таблица 9), в 92% экспериментов эгомобиль успешно завершил обгон переднего препятствия, запланированный планировщиком FFStreams++. В 47% из 92% успешных экспериментов по обгону, планировщик FFStreams++ планировал маневр обгона после ожидания прохождения соседнего препятствия Obstacle2, что отражает критерии безопасности планировщика. С другой стороны, планировщик на основе поиска успешно выполнил обгон только в 34% экспериментов, а в 16% планировщик на основе поиска планировал обгон после ожидания прохождения соседнего препятствия. После оценки эффективности

двух методов по метрике OPM в сценариях на шоссе, часть траекторий FFStreams++ была классифицирована как «нормальный водитель», а часть — как «агрессивный водитель», в то время как траектории на основе поиска все были классифицированы как «агрессивный водитель», что доказывает лучшую производительность FFStreams++ и более близкое поведение к поведению водителя-человека.

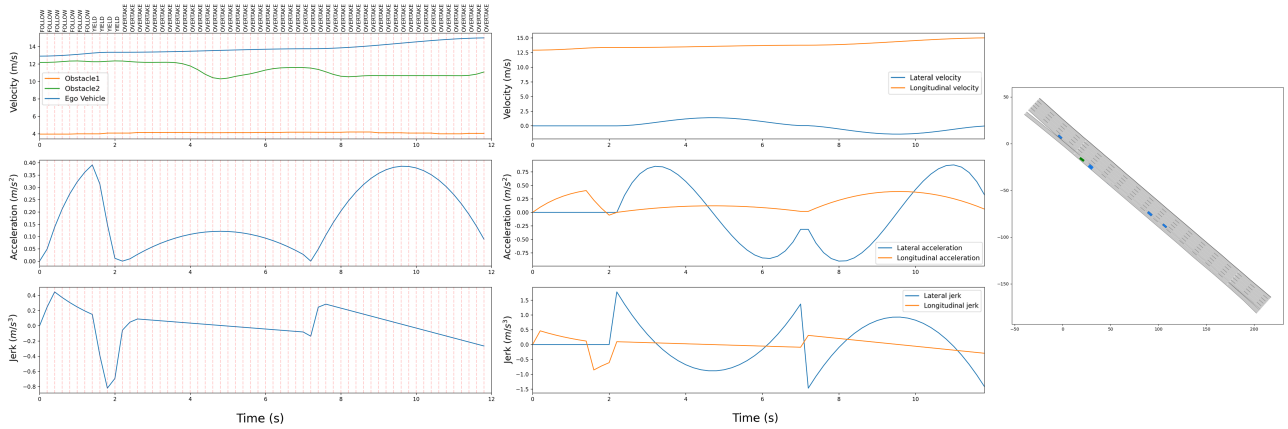


Рисунок 6.8 — Успешный эксперимент по маневру обгона. По сценарию, впереди беспилотного автомобиля находится препятствие Obstacle1, движущееся на низкой скорости, и еще три препятствия на левой соседней полосе. Соседнее препятствие позади беспилотного автомобиля – Obstacle2 – движется с большой скоростью. Планировщик FFStreams++ принимает решения и планирует безопасную траекторию для обгона переднего препятствия.

На рисунке 6.8 показан критический эксперимент по обгону в сценарии с препятствиями, движущимися на высокой скорости. FFStreams++ успешно спланировал решения и безопасную траекторию для выполнения маневра обгона. Демонстрируются профили запланированных абсолютной скорости, ускорения и рывка, а также поперечной и продольной скорости, ускорения и рывка. Планируемая FFStreams++ траектория имеет низкие значения ускорения, соответствующие ограничению на ускорение, и низкие оптимизированные значения рывка, что приводит к плавной и комфортной траектории для пассажиров. Планировщики на основе поиска, с другой стороны, не смогли спланировать будущую безопасную траекторию.

6.4 Анализ экспериментальных результатов

В сценариях проезда перекрестков наши экспериментальные результаты показывают, что процент неудач заметно выше в сценариях с левым поворотом по сравнению со сценариями с движением прямо, и оба показателя выше, чем в сценариях на шоссе. Это можно объяснить трудностями, связанными с точным предсказанием траекторий движения встречных автомобилей на перекрестках. Текущая модель предсказания в значительной степени опирается на исторические состояния препятствий и полигонов дороги. Однако она не учитывает в явном виде правила проезда перекрестков, такие как уступка перед перекрестком для наблюдения за состоянием окружающих автомобилей, которые важны для точного прогнозирования траектории в таких сложных сценариях. Кроме того, модель склонна отдавать предпочтение историческим данным перед геометрией дороги, что, как показано на рисунках 6.6 и 6.7, иногда приводит к предсказанию траекторий, которые отклоняются от границ дороги.

Это ограничение существенно влияет на процесс принятия решений, особенно когда возникает задержка в ожидании того, что встречный автомобиль замедлится перед перекрестком. Такие задержки могут привести к тому, что система планирования примет ряд нерешительных решений, что может нарушить плавность общего процесса принятия решений. Например, система может чередовать решения уступить (замедлиться) и следовать (ускориться) целевой скорости в течение нескольких циклов планирования, прежде чем окончательно примет решение о безопасном маневре. Как показано на рисунке 6.7, решение «следовать» принимается в моменты времени $[1.6s, 1.8s]$, «уступать» — в моменты времени $[2.0s, 2.2s, 2.4s]$, а затем решение «следовать» последовательно принимается до конца сценария. Несмотря на эти сложности, способность фреймворка к перепланировке в течение менее 200 миллисекунд позволяет ему адаптироваться к изменениям предсказаний и в конечном итоге безопасно проезжать перекрестки по запланированным траекториям.

Предложенная система FFStreams++ демонстрирует высокую способность адаптироваться и генерировать плавные траектории, даже когда приходится часто менять решение между уступкой и следованием (ускорением).

Эта адаптивность достигается за счет оптимизированных траекторий, которые минимизируют рывки и придерживаются строгих ограничений на максимальное ускорение, обеспечивая плавное и безопасное реагирование в динамически меняющихся условиях.

В сценариях на шоссе предсказания траекторий движения препятствий значительно более точны. Такая точность во многом объясняется тем, что препятствия на шоссе обычно сохраняют курс, если только они явно не собираются сменить полосу движения. Снижение вероятности внезапного изменения курса упрощает задачу предсказания, что приводит к более надежным результатам планирования.

Что касается процента неудач, то предложенная система демонстрирует 8% неудач в сценариях обгона и 11% и 16% в сценариях проезда перекрестков. В целом, процент неудач системы колеблется от 9% до 16%, что мы считаем приемлемым, учитывая сложный характер сценариев, включая обгон на шоссе и проезд несигнализированных перекрестков с высокими факторами риска.

Устойчивость системы к различным кинематическим характеристикам и типам транспортных средств в различных сценариях гарантируется несколькими ключевыми особенностями:

- Проверка столкновений: выполняется путем симулирования каждого будущего временного шага движения, где 2D-проекции автомобилей на предсказаниях траекторий сверяются с 2D-проекцией беспилотного автомобиля на сэмплировании траекторий с помощью библиотеки CommonRoad Drivability Checker.
- Гибкость параметров: параметры оптимизации Frenet и Jerk могут быть настроены на максимальные значения ускорения и кривизны, характерные для конкретного беспилотного автомобиля, что обеспечивает адаптивность системы.
- Постоянство эффективности: система стабильно обеспечивает надежную работу в различных сценариях, включая шоссе и городские условия.

В целом, предложенная система демонстрирует высокую степень надежности и адаптивности, что делает ее подходящей для автономного вождения как на городских перекрестках, так и в сценариях на шоссе.

6.5 Заключение

В этой главе мы представили фреймворк FFStreams++ — новый и передовой подход для интеграции принятия решений и планирования движения в системах автономного вождения. Объединяя методы сэмплирования с методами поиска, FFStreams++ позволяет генерировать траектории, специфичные для маневра, и обновлять состояние исходной задачи PDDL с помощью Streams. Затем он ищет оптимальный план с помощью эвристического поиска FastForward, гарантируя, что рассматриваются только выполнимые траектории. Такой подход повышает эффективность и результативность процесса планирования, объединяя принятие решений и планирование движения, что позволяет находить оптимальные решения даже в динамичных и неопределенных условиях.

Ключевым новшеством фреймворка FFStreams++ является интеграция сети Query-Centric Network (QCNet) для точного прогнозирования траекторий. Благодаря предсказанию движения и ускорения окружающих препятствий, QCNet усиливает возможности планировщика по принятию решений, позволяя ему предвидеть будущие состояния и планировать безопасные и эффективные маневры. Наша экспериментальная оценка с использованием среды моделирования CommonRoad подчеркивает эффективность FFStreams++ в обработке сложных сценариев движения, таких как незащищенные левые повороты и обгоны. Результаты демонстрируют способность системы безопасно и надежно работать в различных динамических условиях движения от городских условий до автомагистралей.

Данная работа значительно продвигает технологию беспилотных автомобилей, предлагая надежное и масштабируемое решение для предсказания траектории и планирования движения. FFStreams++ доказала свою адаптивность, способность выполнять сложные маневры и устойчивость к непредсказуемым условиям окружающей среды.

Заглядывая вперед, можно выделить несколько направлений для будущих исследований. Повышение точности модели предсказаний QCNet и расширение возможностей фреймворка по управлению дополнительными маневрами, такими как слияние, выезд и управление приоритетами на перекрестках, будут иметь решающее значение для дальнейшего применения фреймворка.

Кроме того, интеграция семантики светофоров и информации о пешеходных переходах позволит FFStreams++ принимать решения с учетом контекста в условиях плотной городской застройки. Эти усовершенствования расширят универсальность и надежность фреймворка, гарантируя, что он останется мощным инструментом для систем автономного вождения, работающих в самых разных реальных условиях.

Заключение

В данной диссертации исследованы различные методы автоматического планирования в беспилотных автомобилях с упором на интеграцию детерминированных и вероятностных подходов, адаптивное планирование маневров, обучение с подкреплением и быстрый эвристический поиск с предсказанием траектории. Решая уникальные задачи предсказания движения, принятия решений и планирования движения в динамичных и неопределенных условиях, исследование вносит значительный вклад в развитие возможностей автономных систем вождения. Универсальность и устойчивость системы гарантирует, что она останется мощным инструментом для автономных систем вождения, работающих в самых разных условиях реального мира. Основные результаты работы заключаются в следующем.

1. адаптивное планирование маневров обгона с использованием подхода «дерево поведения» , включающего генетическое программирование для динамической адаптации к новым условиям вождения.
2. обучение с подкреплением (RL) для адаптивного планирования маневров при парковке, где использование обучения на основе curriculum learning позволило системе постепенно обучаться сложным задачам вождения. Основанная на RL система оказалась эффективной в сценариях, требующих адаптивного принятия решений, позволяя беспилотному автомобилю постоянно совершенствовать свои стратегии планирования и выполнения путем проб и ошибок.
3. метод FFStreams - подход к быстрому эвристическому поиску с использованием сэмплеров потоков для эффективной генерации траекторий. Благодаря объединению потока конфигурации и потока траекторий, система планирования успешно сократила пространство поиска, обеспечив при этом создание выполнимых траекторий. Этот подход оказался особенно полезен в сценариях обгона и смены полосы движения, требующих быстрого и надежного планирования, что было продемонстрировано в ходе экспериментов.
4. метод FFStreams++, объединяющий быстрый эвристический поиск и предсказание траектории с помощью модели QCNet для адаптивного планирования маневров. Интеграция предсказания траектории

улучшила процесс принятия решений, обеспечив точное предсказание окружающих препятствий в реальном времени. Экспериментальные результаты, в частности, в бенчмарке CommonRoad, а также в сценариях проезда перекрестков и шоссе, показали, что фреймворк FFStreams++ стабильно превосходит другие методы как по безопасности, так и по адаптивности.

В заключение следует отметить, что данная диссертация продемонстрировала, что сочетание различных методологий планирования - от детерминированного и вероятностного планирования до адаптивного обучения и быстрого эвристического поиска - позволяет системам беспилотных автомобилей достичь надежных, эффективных и масштабируемых возможностей планирования движения. Будущие исследования могут расширить эти результаты за счет усовершенствования моделей предсказания, интеграции элементов, учитывающих контекст, таких как семантика трафика, и усовершенствования системы планирования для обработки еще более широкого спектра сценариев движения. Эти достижения будут способствовать дальнейшему развитию технологии беспилотного вождения, обеспечивая более безопасную и надежную работу автомобиля во все более сложных условиях.

Список сокращений и условных обозначений

S	пространство состояний, см. (1.3)
A	пространство действий, см. (1.3)
$\mathcal{T}(s,a)$	функция перехода состояний, см. (1.3)
Σ	пространство планирования, см. (1.3)
$\mathcal{P}$	задача планирования
π	план, см. (1.7)
G	цель
T	множество типов объектов, см. (1.4)
P	множество предикатов, см. (1.4)
F	множество функций, см. (1.4)
B	множество объектов, см. (1.4)
$\Pr(s' \mid s, a)$	вероятность достижения состояния s', см. (1.5)
$\mathbb{R}^+$	множество всех положительных действительных чисел
$\Pr(o \mid s, a)$	вероятность наблюдения o после выполнения действия a в состоянии s
R	функция вознаграждения, см. (1.6)
γ	фактор дисконтирования, см. (1.6)
$\pi(* \mid s)$	стратегия агента, см. (1.6)
$J(\pi)$	функция полезности, см. (1.9)
$Q^\pi(s,a)$	Q-функция (функция полезности) для стратегии π, см. (1.11)
$Q^*(s,a)$	Функция полезности для оптимальной стратегии, см. (1.12)
f	функция приспособленности, см. (3.1)
T	множество терминалов дерева поведения, см. (3.1)
F	множество функций дерева поведения, см. (3.1)
L	язык программирования $L = \{F,T\}$
G	пространство генотипов
$d(t)$	безопасное расстояние следования, см. (3.2)
$v(t)$	скорость беспилотного автомобиля, см. (3.2)
C_{total}	общая стоимость траектории, см. (5.1)
C_l	стоимость траектории при боковом движении, см. (5.1)
C_s	стоимость траектории на продольное перемещение, см. (5.1)

J_l	функция боковых рывков, см. (5.2)
J_s	функция продольных рывков, см. (5.3)
T	временной интервал, см. (5.2)
k_j	вес рывка, см. (5.2)
J	общая стоимость траектории, см. (6.2)
$J_{comfort}$	затраты, связанные с комфортом пассажиров, см. (6.2)
$J_{efficiency}$	затраты, связанные со временем в пути, см. (6.2)
$J_{lateral\_error}$	затраты, связанные с боковой ошибкой в конечной точке, см. (6.2)
J_{speed_error}	затраты, связанные с ошибкой продольной скорости в конечной точке, см. (6.2)
ω_j	весовый коэффициент для рывка, см. (6.7)
ω_t	весовый коэффициент для времени движения, см. (6.7)
ω_{err}	весовый коэффициент для стоимости ошибки, см. (6.7)

Публикации автора по теме диссертации

1. *Jamal, M.* Adaptive maneuver planning for autonomous vehicles using behavior tree on apollo platform / M. Jamal, A. Panov // Artificial Intelligence XXXVIII: 41st SGAI International Conference on Artificial Intelligence, AI 2021, Cambridge, UK, December 14–16, 2021, Proceedings 41. — Springer. 2021. — С. 327—340.
2. *Gorbov, G.* Learning adaptive parking maneuvers for self-driving cars / G. Gorbov, M. Jamal, A. I. Panov // International Conference on Intelligent Information Technologies for Industry. — Springer. 2022. — С. 283—292.
3. *Jamal, M.* FFStreams: Fast Search With Streams for Autonomous Maneuver Planning / M. Jamal, A. Panov // IEEE Robotics and Automation Letters. — 2024.
4. *Jamal, M.* Maneuver Decision-Making with Trajectory Streams Prediction for Autonomous Vehicles / M. Jamal, A. Panov // arXiv preprint arXiv:2409.10165. — 2024.

Список литературы

5. Motion planning for autonomous driving: The state of the art and future perspectives / S. Teng [и др.] // IEEE Transactions on Intelligent Vehicles. — 2023.
6. A survey of autonomous driving: Common practices and emerging technologies / E. Yurtsever [и др.] // IEEE access. — 2020. — Т. 8. — С. 58443—58469.
7. *Gonzalez, D. S.* Towards human-like prediction and decision-making for automated vehicles in highway scenarios : дис. ... канд. / Gonzalez David Sierra. — Université Grenoble Alpes, 2019.
8. Planning for safe abortable overtaking maneuvers in autonomous driving / J. Palatti [и др.] // 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). — IEEE. 2021. — С. 508—514.
9. Decision-making technology for autonomous vehicles: Learning-based methods, applications and future outlook / Q. Liu [и др.] // 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). — IEEE. 2021. — С. 30—37.
10. *Buehler, M.* Junior: the Stanford entry in the urban challenge / M. Buehler, K. Iagnemma, S. Singh // The DARPA urban challenge: autonomous vehicles in city traffic. — 2009. — Т. 56. — С. 91—123.
11. Making bertha drive—an autonomous journey on a historic route / J. Ziegler [и др.] // IEEE Intelligent transportation systems magazine. — 2014. — Т. 6, № 2. — С. 8—20.
12. Optimization-based tactical behavior planning for autonomous freeway driving in favor of the traffic flow / H. Bey [и др.] // 2019 IEEE Intelligent Vehicles Symposium (IV). — IEEE. 2019. — С. 1033—1040.
13. *Artuñedo, A.* A decision-making architecture for automated driving without detailed prior maps / A. Artuñedo, J. Godoy, J. Villagra // 2019 IEEE Intelligent Vehicles Symposium (IV). — IEEE. 2019. — С. 1645—1652.

14. *Dong, C.* Intention estimation for ramp merging control in autonomous driving / C. Dong, J. M. Dolan, B. Litkouhi // 2017 IEEE intelligent vehicles symposium (IV). — IEEE. 2017. — С. 1584—1589.

15. *Isele, D.* Interactive decision making for autonomous vehicles in dense traffic / D. Isele // 2019 IEEE Intelligent Transportation Systems Conference (ITSC). — IEEE. 2019. — С. 3981—3986.

16. *Schwarting, W.* Planning and decision-making for autonomous vehicles / W. Schwarting, J. Alonso-Mora, D. Rus // Annual Review of Control, Robotics, and Autonomous Systems. — 2018. — Т. 1, № 1. — С. 187—210.

17. *Dulac-Arnold, G.* Challenges of real-world reinforcement learning / G. Dulac-Arnold, D. Mankowitz, T. Hester // arXiv preprint arXiv:1904.12901. — 2019.

18. Deep reinforcement learning for autonomous driving: A survey / B. R. Kiran [и др.] // IEEE Transactions on Intelligent Transportation Systems. — 2021. — Т. 23, № 6. — С. 4909—4926.

19. Baidu Apollo team (2017), Apollo: Open Source Autonomous Driving, howpublished = https://github.com/ApolloAuto/apollo, note = Accessed: 2019-02-11.

20. Query-Centric Trajectory Prediction / Z. Zhou [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2023. — С. 17863—17873.

21. *Althoff, M.* CommonRoad: Composable benchmarks for motion planning on roads / M. Althoff, M. Koschi, S. Manzinger // 2017 IEEE Intelligent Vehicles Symposium (IV). — IEEE. 2017. — С. 719—726.

1. *Jamal, M.* Adaptive maneuver planning for autonomous vehicles using behavior tree on apollo platform / M. Jamal, A. Panov // Artificial Intelligence XXXVIII: 41st SGAI International Conference on Artificial Intelligence, AI 2021, Cambridge, UK, December 14–16, 2021, Proceedings 41. — Springer. 2021. — С. 327—340.

2. *Gorbov, G.* Learning adaptive parking maneuvers for self-driving cars / G. Gorbov, M. Jamal, A. I. Panov // International Conference on Intelligent Information Technologies for Industry. — Springer. 2022. — С. 283—292.

3. *Jamal, M.* FFStreams: Fast Search With Streams for Autonomous Maneuver Planning / M. Jamal, A. Panov // IEEE Robotics and Automation Letters. — 2024.

4. *Jamal, M.* Maneuver Decision-Making with Trajectory Streams Prediction for Autonomous Vehicles / M. Jamal, A. Panov // arXiv preprint arXiv:2409.10165. — 2024.

22. *Committee, O.-R. A. D. (* Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles / O.-R. A. D. (Committee. — SAE International, 2021.

23. A Review of Decision-Making and Planning for Autonomous Vehicles in Intersection Environments / S. Chen [и др.] // World Electric Vehicle Journal. — 2024. — Т. 15, № 3. — С. 99.

24. *Wong, C.* Adaptive task planning and motion planning for robots in dynamic environments / C. Wong. — University of Strathclyde (United Kingdom), 2020.

25. Representation, learning, and planning algorithms for geometric task and motion planning / B. Kim [и др.] // The International Journal of Robotics Research. — 2022. — Т. 41, № 2. — С. 210—231.

26. *Migimatsu, T.* Object-centric task and motion planning in dynamic environments / T. Migimatsu, J. Bohg // IEEE Robotics and Automation Letters. — 2020. — Т. 5, № 2. — С. 844—851.

27. *Bharilya, V.* Machine learning for autonomous vehicle's trajectory prediction: A comprehensive survey, challenges, and future research directions / V. Bharilya, N. Kumar // Vehicular Communications. — 2024. — С. 100733.

28. A dynamic Bayesian network for vehicle maneuver prediction in highway driving scenarios: Framework and verification / J. Li [и др.] // Electronics. — 2019. — Т. 8, № 1. — С. 40.

29. Vehicle trajectory prediction by integrating physics-and maneuver-based approaches using interactive multiple models / G. Xie [и др.] // IEEE Transactions on Industrial Electronics. — 2017. — Т. 65, № 7. — С. 5999—6008.

30. *Deo, N.* Convolutional social pooling for vehicle trajectory prediction / N. Deo, M. M. Trivedi // Proceedings of the IEEE conference on computer vision and pattern recognition workshops. — 2018. — С. 1468—1476.

31. Motion Query-based Multimodal Vehicle Trajectory Prediction for Autonomous Driving / H. Jiang [и др.] // 2023 7th CAA International Conference on Vehicular Control and Intelligence (CVCI). — IEEE. 2023. — С. 1—6.

32. *Cheng, J.* Forecast-mae: Self-supervised pre-training for motion forecasting with masked autoencoders / J. Cheng, X. Mei, M. Liu // Proceedings of the IEEE/CVF International Conference on Computer Vision. — 2023. — С. 8679—8689.

33. Motion transformer with global intention localization and local movement refinement / S. Shi [и др.] // Advances in Neural Information Processing Systems. — 2022. — Т. 35. — С. 6531—6543.

34. *Angulo, B.* Policy Optimization to Learn Adaptive Motion Primitives in Path Planning With Dynamic Obstacles / B. Angulo, A. Panov, K. Yakovlev // IEEE Robotics and Automation Letters. — 2023. — Т. 8, № 2. — С. 824—831.

35. Reachability-based decision-making for autonomous driving: Theory and experiments / H. Ahn [и др.] // IEEE Transactions on Control Systems Technology. — 2020. — Т. 29, № 5. — С. 1907—1921.

36. *Villagra, J.* Decision-Making Techniques for Autonomous Vehicles / J. Villagra, F. Jimenez. — Elsevier, 03/2023.

37. *Fox, M.* PDDL2. 1: An extension to PDDL for expressing temporal planning domains / M. Fox, D. Long // Journal of artificial intelligence research. — 2003. — Т. 20. — С. 61—124.

38. Optimal control of Markov decision processes with incomplete state estimation / K. J. Astrom [и др.] // Journal of mathematical analysis and applications. — 1965. — Т. 10, № 1. — С. 174—205.

39. A Novel Lane-Change Decision-Making With Long-Time Trajectory Prediction for Autonomous Vehicle / X. Wang [и др.] // IEEE Access. — 2023. — Т. 11. — С. 137437—137449.

40. Argoverse 2: Next generation datasets for self-driving perception and forecasting / B. Wilson [и др.] // arXiv preprint arXiv:2301.00493. — 2023.

41. Junior: The stanford entry in the urban challenge / M. Montemerlo [и др.] // Journal of field Robotics. — 2008. — Т. 25, № 9. — С. 569—597.

42. Autonomous driving in urban environments: Boss and the urban challenge / C. Urmson [и др.] // Journal of Field Robotics. — 2008. — Т. 25, № 8. — С. 425—466.

43. *Harel, D.* Statecharts: A visual formalism for complex systems / D. Harel // Science of computer programming. — 1987. — Т. 8, № 3. — С. 231—274.

44. *Olsson, M.* Behavior trees for decision-making in autonomous driving / M. Olsson. — 2016.

45. A fuzzy-inference-based reinforcement learning method of overtaking decision making for automated vehicles / Q. Wu [и др.] // Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering. — 2022. — Т. 236, № 1. — С. 75—83.

46. Autonomous planning and control for intelligent vehicles in traffic / C. You [и др.] // IEEE Transactions on Intelligent Transportation Systems. — 2019. — Т. 21, № 6. — С. 2339—2349.

47. *Brechtel, S.* Probabilistic MDP-behavior planning for cars / S. Brechtel, T. Gindele, R. Dillmann // 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC). — IEEE. 2011. — С. 1537—1542.

48. *Ulbrich, S.* Probabilistic online POMDP decision making for lane changes in fully automated driving / S. Ulbrich, M. Maurer // 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013). — IEEE. 2013. — С. 2063—2067.

49. High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning / B. Mirchevska [и др.] // 2018 21st International Conference on Intelligent Transportation Systems (ITSC). — IEEE. 2018. — С. 2156—2162.

50. Multi-lane Cruising Using Hierarchical Planning and Reinforcement Learning / K. Rezaee [и др.] // 2019 IEEE Intelligent Transportation Systems Conference (ITSC). — IEEE. 2019. — С. 1800—1806.

51. Autonomous driving trajectory optimization with dual-loop iterative anchoring path smoothing and piecewise-jerk speed optimization / J. Zhou [и др.] // IEEE Robotics and Automation Letters. — 2020. — Т. 6, № 2. — С. 439—446.

52. Research on local path planning based on improved RRT algorithm / C. Zong [и др.] // Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering. — 2021. — Т. 235, № 8. — С. 2086—2100.

53. Semantic-level maneuver sampling and trajectory planning for on-road autonomous driving in dynamic scenarios / H. Li [и др.] // IEEE Transactions on Vehicular Technology. — 2021. — Т. 70, № 2. — С. 1122—1134.

54. An integrated framework of decision making and motion planning for autonomous vehicles considering social behaviors / P. Hang [и др.] // IEEE transactions on vehicular technology. — 2020. — Т. 69, № 12. — С. 14458—14469.

55. Rule-based safety-critical control design using control barrier functions with application to autonomous lane change / S. He [и др.] // 2021 American Control Conference (ACC). — IEEE. 2021. — С. 178—185.

56. Maneuver identification for interaction-aware highway lane change behavior planning based on polygon clipping and convex optimization / M. Schmidt [и др.] // 2019 IEEE Intelligent Transportation Systems Conference (ITSC). — IEEE. 2019. — С. 3948—3953.

57. *Li, S.* Combining decision making and trajectory planning for lane changing using deep reinforcement learning / S. Li, C. Wei, Y. Wang // IEEE Transactions on Intelligent Transportation Systems. — 2022. — Т. 23, № 9. — С. 16110—16136.

58. Evolutionary decision-making and planning for autonomous driving based on safe and rational exploration and exploitation / K. Yuan [и др.] // Engineering. — 2024. — Т. 33. — С. 108—120.

59. Automated lane change strategy using proximal policy optimization-based deep reinforcement learning / F. Ye [и др.] // 2020 IEEE Intelligent Vehicles Symposium (IV). — IEEE. 2020. — С. 1746—1752.

60. *Du, Z.* Trajectory planning for automated parking systems using deep reinforcement learning / Z. Du, Q. Miao, C. Zong // International Journal of Automotive Technology. — 2020. — Т. 21. — С. 881—887.

61. *Fu, Y.* A reinforcement learning behavior tree framework for game AI / Y. Fu, L. Qin, Q. Yin // 2016 International Conference on Economics, Social Science, Arts, Education and Management Engineering. — Atlantis Press. 2016. — С. 573—579.

62. *Pereira, R. d. P.* A framework for constrained and adaptive behavior-based agents / R. d. P. Pereira, P. M. Engel // arXiv preprint arXiv:1506.02312. — 2015.

63. Learning Behavior Trees with Genetic Programming in Unpredictable Environments / M. Iovino [и др.] // arXiv preprint arXiv:2011.03252. — 2020.

64. Learning behavior trees for autonomous agents with hybrid constraints evolution / Q. Zhang [и др.] // Applied Sciences. — 2018. — Т. 8, № 7. — С. 1077.

65. Optimal Vehicle Path Planning Using Quadratic Optimization for Baidu Apollo Open Platform / Y. Zhang [и др.] // 2020 IEEE Intelligent Vehicles Symposium (IV). — 2020. — С. 978—984.

66. *Angulo, B.* Policy optimization to learn adaptive motion primitives in path planning with dynamic obstacles / B. Angulo, A. Panov, K. Yakovlev // IEEE Robotics and Automation Letters. — 2022. — Т. 8, № 2. — С. 824—831.

67. Proximal policy optimization algorithms / J. Schulman [и др.] // arXiv preprint arXiv:1707.06347. — 2017.

68. *Hoffmann, J.* FF: The fast-forward planning system / J. Hoffmann // AI magazine. — 2001. — Т. 22, № 3. — С. 57—57.

69. Optimal trajectory generation for dynamic street scenarios in a frenet frame / M. Werling [и др.] // 2010 IEEE International Conference on Robotics and Automation. — IEEE. 2010. — С. 987—993.

70. Interval Prediction for Continuous-Time Systems with Parametric Uncertainties / E. Leurent [и др.] // 2019 IEEE 58th Conference on Decision and Control (CDC). — 2019. — С. 7049—7054.

71. *Leurent, E.* rl-agents: Implementations of Reinforcement Learning algorithms / E. Leurent. — 2018. — https://github.com/eleurent/rl-agents.

72. Self-driving like a human driver instead of a robocar: Personalized comfortable driving experience for autonomous vehicles / I. Bae [и др.] // arXiv preprint arXiv:2001.03908. — 2020.

73. *Likhachev, M.* Search-based Planning Library (SBPL) including a collection of graph searches and planners that utilize these graph searches / M. Likhachev. — 11.2018.

Список рисунков

Список таблиц

Приложение А

Акт о внедрении и использовании результатов исследования

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
«ИНТЕГРАЦИЯ НОВЫХ ТЕХНОЛОГИЙ»

АКТ

об использовании результатов диссертационной работы

Жамал Маис

г. Москва 17.09.2024

на тему «Разработка и исследование методов и алгоритмов адаптивного планирования маневров беспилотного автомобиля», представленной на соискание ученой степени кандидата технических наук

Результаты диссертационной работы «Разработка и исследование методов и алгоритмов адаптивного планирования маневров беспилотного автомобиля» обладают высокой актуальностью и представляют практический интерес для решения задач принятия решений и планирования движения в беспилотных технологиях.

В диссертации Жамал М. разработаны методы адаптивного планирования маневров автономного автомобиля для задач, связанных с парковкой, перестроением и обгоном динамических препятствий. Данные методы основаны на передовых достижениях в таких областях, как детерминированное планирование с выборкой, обучение с подкреплением и планирование пути. Кроме того, был создан программно-алгоритмический инструментарий, предназначенный для решения задачи принятия решений для беспилотного автотранспортного средства (АТС) в сложной и изменяющейся среде, что позволило использовать как обучаемые, так и классические подходы к планированию пути в системе управления.

Результаты, полученные в диссертационной работе, были практически использованы при выполнении научно-исследовательской работы (НИР) «Исследование и разработка комплекта аппаратно-программных средств для решения задач локализации и планирования пути беспилотного транспортного средства в условиях дорог общего пользования». В ходе выполнения этой НИР был разработан набор экспериментальных программных решений для

автоматического управления беспилотным АТС на базе Kia Soul в условиях дорог общего пользования, с учетом кинематических и динамических параметров шасси АТС, информации о дорожной инфраструктуре, получаемой из HD-карт и заданных скоростных ограничений.

Благодаря использованию экспериментальных программных реализаций, на данных АТС удалось значительно повысить качество автономного вождения и надежность системы автоматического управления при обгоне статических и динамических препятствий, перестроении в другие полосы движения и при проезде перекрестков. В частности, при проезде перекрестков были достигнуты следующие метрики качества: средняя длина траекторий 44 м., время проезда 27 сек., метрика AOL - 1,13 рад/м, что на 10% улучшает характеристики по сравнению с результатами полевых испытаний при ручном управлении.

Генеральный директор ***Т.В. Михайлова***

Общество с ограниченной ответственностью «Интеграция новых технологий» ИНТЕГРАНТ г. Москва ОГРН 1197746284604 ИНН 7714453083