МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный университет»

ИНЖЕНЕРНАЯ ШКОЛА

Кафедра Автоматизации и управления

Карманова Светлана Вячеславовна

РАЗРАБОТКА СИСТЕМЫ ПЛАНИРОВАНИЯ ГЛАДКИХ ПРОСТРАНСТВЕННЫХ ТРАЕКТОРИЙ ДВИЖЕНИЯ МОБИЛЬНОГО РОБОТА НА ОСНОВЕ НЕЧЕТКОЙ ЛОГИКИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по направлению подготовки бакалавров
15.03.06 – Мехатроника и робототехника
профиль «***Мехатроника и робототехника***»

г. Владивосток
2018

Студент _____
(подпись)
«20» июня 2018 г.

Руководитель выпускной квалификационной работы (проекта) д.т.н.
(должность, ученое звание)
_____ Юхимец Д.А.
(подпись) (ФИО)
«21» июня 2018 г.

«Допустить к защите»

Руководитель ОП к.т.н.
(ученое звание)
_____ Кацурин А. А.
(подпись) (ФИО)
«27» июня 2018 г.

Зав. кафедрой д.т.н., профессор
(ученое звание)
_____ Филаретов В. Ф.
(подпись) (ФИО)
«27» июня 2018г

Защищена в ГАК с оценкой отлично

Секретарь ГАК
_____ Мурсалимов Э. Ш.
(подпись) (ФИО)
«4» июля 2018 г.

В материалах данной выпускной квалификационной работы не содержатся сведения, составляющие государственную тайну, и сведения, подлежащие экспортному контролю.

Директор ИШ ДВФУ _____ А.Т. Беккер

Сведений, содержащих государственную тайну, нет.
Экизов А. Мирзаева

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный университет»

Инженерная школа
Кафедра автоматизации и управления

УТВЕРЖДЕНО
Руководитель ОП к.т.н.
(ученая степень, должность)
Кацурин А.А.
(подпись) (ФИО)
« 3 » октября 2017 г.

Заведующий кафедрой к.т.н., профессор
(ученая степень, звание)
Филаретов В.Ф.
(подпись) (ФИО)
« 3 » октября 2017 г.

З А Д А Н И Е
на выпускную квалификационную работу

Студенту **Кармановой Светлане Вячеславовне** (Фамилия, Имя, Отчество) Группа **Б3421** (номер группы)

1. Наименование темы **Разработка системы планирования гладких пространственных траекторий движения мобильного робота на основе нечеткой логики**

2. Основания для разработки

3. Источники разработки **Yukhimets, D. A. Method of spatial path planning for mobile robot in unknown environment / D.A. Yukhimets, A. V. Zuev, A.S. Gubankov. // Proceedings of the 28th DAAAM International Symposium. – Vienna, Austria: - 2017. – С. 258 – 267.**

4. Технические требования (параметры) **Разрабатываемая система управления должна обеспечивать безопасное движение мобильного робота по сформированной гладкой траектории в неизвестном окружении, предусматривать автоматический выбор стратегии движения и учитывать динамические ограничения мобильного робота.**

5. Дополнительные требования **Отсутствуют.**

6. Перечень разработанных вопросов **1. Изучение систем планирования траекторий движения мобильного робота.**
2. Реализация системы планирования гладких траекторий движения мобильного робота в условиях неизвестного окружения для трехмерного случая.
3. Разработка нечеткой системы для комбинации алгоритмов обхода препятствий в вертикальной и горизонтальной плоскостях.
4. Математическое моделирование для исследования разработанного метода.

7. Перечень графического материала (с точным указанием обязательных плакатов)

Графический материал отсутствует.

КАЛЕНДАРНЫЙ ГРАФИК ВЫПОЛНЕНИЯ РАБОТЫ

№ п/п	Наименование этапов дипломного проекта (работы)	Срок выполнения этапов дипломного проекта (работы)	Примечание
1	Написание введения и постановки задачи	До 15.02.2018	
2	Написание первой главы ВКР (обзор существующих подходов)	До 28.02.2018	
3	Реализация алгоритма планирования гладких траекторий в неизвестном окружении для двумерного случая	До 15.03.2018	
4	Реализация алгоритма планирования пути в неизвестном окружении для трехмерного случая	До 30.03. 2018	
5	Доработка исходного алгоритма элементами нечеткой логики	До 15.04. 2018	
6	Написание второй главы ВКР (описание разработанного метода)	До 30.04.2018	
7	Написание третьей главы ВКР (описание результатов моделирования)	До 15.05.2018	
8	Доработка ВКР в соответствии с замечаниями руководителя	До 30.05.2018	
9	Проверка ВКР на Антиплагиат	До 05.06.2018	
10	Подписи ВКР у руководителя ВКР, руководителя ОП, заведующего кафедрой	До 10.06.2018	
11	Подготовка к защите (подготовка доклада и презентация в Power Point)	До 19.06.2018	
12	Защита ВКР в ГЭК	04.07.2018	

Дата выдачи задания «2» октября 2017 г.

Срок представления к защите июнь 2018 г.

Руководитель ВКР _____ (подпись) Юхимец Д.А. (ФИО)

Студент _____ (подпись) Карманова С.В. (ФИО)

Аннотация

В работе рассматривается метод формирования пространственных траекторий движения мобильных роботов в неизвестном окружении с применением элементов нечеткой логики. Траектории в горизонтальной плоскости представляют собой сплайны Безье третьего порядка, которые формируются на основе показаний бортовых датчиков. В вертикальной плоскости поддерживается постоянная высота над поверхностью движения. Нечеткая логика обеспечивает возможность достижения цели, которая находится на вершине возвышенности с резким уклоном. Использование датчиков приближения позволяет автоматически выбирать стратегию обхода препятствий с учетом динамических ограничений мобильного робота.

Результаты моделирования подтверждают эффективность и малую вычислительную сложность метода.

Введение

В настоящее время наблюдается активное развитие сферы применения мобильных роботов. Для широкого применения мобильных роботов в повседневной жизни необходима разработка алгоритмов планирования траекторий движения, которые обеспечили бы безопасное взаимодействие роботов с людьми, а также методов, позволяющих взаимодействовать роботам в составе мультиагентных систем. В таких случаях достаточно значимыми и обособленными являются задачи, которые роботы способны выполнять в автономном режиме без контроля оператора в окружении, априорной информации о котором недостаточно: транспортировка грузов различного назначения, осмотр зданий, поисковые операции и многие другие задачи. В процессе выполнения этих задач от системы управления роботом требуется формирование траекторий на основе данных, получаемых непрерывно в процессе движения от датчиков, интегрированных в робота. Такие траектории позволяют роботу двигаться быстро и безопасно в неизвестном окружении, содержащем препятствия.

Задача формирования траекторий движения мобильных роботов в среде, содержащей препятствия – это одна из главных проблем в разработке СУ. На данный момент предложено и модифицировано множество методов, которые можно классифицировать по различным критериям, но, в общем случае, говорят о методах, требующих полной или частичной априорной информации об окружении и о тех, что не требуют информации о местоположении препятствий. Первая группа методов преимущественно представляет собой алгоритмы поиска оптимального пути на графе [1, 2], также существуют генетические алгоритмы [3-5] и некоторые другие методы [6-8]. Часто методы первой группы так или иначе могут быть сведены к теории графов. В случае, когда мобильный робот передвигается в частично известном окружении или среде, где находятся динамические препятствия, маршрут также необходимо

планировать динамически, основываясь на показаниях бортовых датчиков и виде предварительно спланированного пути [9, 10]. Недостаток таких подходов состоит в высокой вычислительной сложности и необходимости разработки эвристик формирования траекторий. В дополнение к вышесказанному траектории, формируемые таким образом, представляют собой набор базовых точек, через которые необходимо пройти роботу. Отсюда, возникает необходимость дополнительного использования интерполяционных алгоритмов, способных обеспечить гладкость траекторий.

Одними из представителей второй группы являются методы на принципе потенциальных полей [11, 12], которые позволяют мобильному роботу двигаться в сложном неизвестном окружении. Однако, подходы такого рода не позволяют управлять траекторией робота в процессе его движения. Также, они могут привести к резкому изменению направления движения, которое не может быть точно отработано СУ робота.

Несмотря на существование большого числа методов формирования траекторий, каждый из них подразумевает решение проблемы в плоскости. Это значительно уменьшает функциональные способности мобильных роботов, передвигающихся в пространстве (подводные, летательные аппараты). Возможность пространственного движения позволяет роботам в некоторых случаях обойти препятствия сверху, без изменения направления движения в горизонтальной плоскости. Однако, при движении в вертикальной плоскости, направление лучей зоны чувствительности бортовых датчиков робота непрерывно изменяется, что может исказить представление о расположении препятствий и привести к генерации неверных траекторий, когда используются традиционные методы.

В некоторых работах рассмотрены методы формирования траекторий в пространстве. В [13] описан гибридный метод для формирования траектории летательного аппарата в гористых местностях. В этом методе сначала формируется оптимальная траектория движения в плоскости на основе сеточного представления окружения робота. Результирующая дискретная

траектория сглаживается с помощью метода потенциальных полей. Главный недостаток описанного метода – это необходимость знать карту расположения препятствий в пространстве и неспособность обходить встречающиеся препятствия сверху.

В [14] описан метод преодоления препятствий в вертикальной плоскости на основе информации от бортовых датчиков. Однако, в данной работе угол тангажа робота предполагается неограниченным, то есть всегда можно обойти препятствие сверху. Предположение применимо только к некоторым летательным аппаратам: тем, у которых есть вертикальные подруливающие устройства.

Таким образом, существует задача планирования пространственного маршрута движения мобильного робота в неизвестном окружении, включающем в себя препятствия, на основе показаний датчиков. Этот метод позволит использовать все преимущества пространственного движения.

В настоящей работе данная задача решена на основе метода формирования гладких траекторий в неизвестном окружении, предложенного в [15], с элементами нечеткой логики.

1 Обзор существующих подходов к решению задачи

Обычно, когда говорят о планировании движения, подразумевают поиск бесконфликтного пути для перемещения твердого тела или кинематической конструкции в трехмерной среде. Задача планирования пути мобильного робота включает в себя четыре основных аспекта. Во-первых, спланированный путь должен представлять собой непрерывную кривую в конфигурационном пространстве (число степеней свободы) объекта, проложенную от заданной начальной точки к заданной конечной точке. Во-вторых, путь должен обеспечивать движение робота с обходом всевозможных препятствий. В-третьих, искомый путь должен удовлетворять всем установленным кинематическим и динамическим ограничениям. И в-четвертых, выбранный путь из возможных, удовлетворяющих первым трем пунктам путей, должен быть в некотором смысле оптимальным.

В большинстве случаев, задача планирования пути допускает бесконечное множество решений (хотя может не существовать ни одного), поэтому часто данная задача формулируется, как поиск оптимального пути, который имеет минимальную длину среди всех возможных путей и обеспечивает движение объекта управления на максимальном расстоянии от препятствий.

По полноте информации об окружающей среде алгоритмы планирования пути преимущественно делятся на две категории, а именно: глобальное и локальное. Глобальное или офлайн планирование требует полноты априорных знаний об окружающей среде и ее неизменности. Следовательно, задача планирования может быть решена заблаговременно: ещё до того, как робот начнет движение [16]. Такая концепция отличается от локального планирования, где необходимо лишь частичное знание окружения. Среда, в которой функционирует робот, может быть динамической, а, следовательно, возможно проявление недостатка информации об окружении.

Принимая это во внимание, мобильный робот должен достигать цели, обладая малым количеством информации об окружающей его среде – как минимум знать своё текущее местоположение и конечный пункт.

Современный алгоритм планирования пути должен интегрировать в себе преимущества различных существующих методов и быть свободен от недостатков отдельных методов. Очень важно комбинировать глобальные и локальные методы. Такой подход позволит планировать офлайн маршрут на основе глобальной информации об окружении, а, благодаря онлайн методам, робот сможет избегать столкновений с препятствиями, которые являются динамическими или не были обнаружены заранее.

Существует множество проблем, которые необходимо рассматривать при построении системы навигации робота. Специфика проблем в общем случае определяется функциями робота и целью, которая перед ним ставится. Большинство предложенных подходов строится на идее поиска кратчайшего пути из начальной точки к целевой. В последнее время разработчики фокусируют свое внимание на уменьшении вычислительной стоимости и повышении гладкости траектории движения робота [17]. Другие текущие проблемы включают автономную навигацию робота в сложном окружении, с учетом движущихся препятствий, навигацию нескольких роботов, взаимодействующих в рамках одной системы, и повышение естественности движения роботов. На рисунке 1.1 показаны сопутствующие проблемы планирования пути.

Математически хорошо проработаны алгоритмы поиска пути, когда характеристики окружающей робота среды предполагаются точно или частично известными. Для этого используется дискретная математика (теория графов) и линейное программирование. Наиболее известны и распространены задачи поиска оптимального пути с помощью графа (например, алгоритм Дейкстры и его модификации). Кроме этого используют и другие алгоритмы (пересечения линий, волнового фронта, эвристические A^*, D^*).

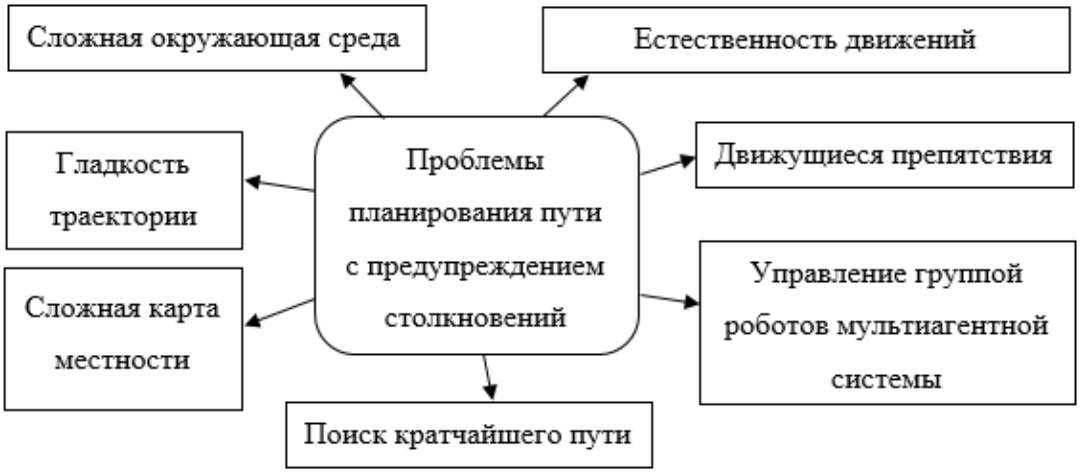

Рисунок 1.1 - Проблемы при планировании пути

Поиск пути в заранее неизвестном окружении формализовать гораздо сложнее, так как нет заранее составленной карты (формального описания окружения). В таком случае не представляется возможным описать пространство, как точно определенную математическую структуру, а, следовательно, не может быть найден и оптимальный путь в привычном понимании слова. Длина спланированного пути обусловлена конкретным выбором на каждом шаге, поскольку нельзя предугадать, приведет ли принятое решение к возникновению комбинации препятствий, выйти из которой возможно только вернувшись на предыдущий шаг, или нет.

В литературе описано большое число разнообразных методов планирования пути, в которых применяются различные эвристики, вытекающие, по большей части, из специфики поставленной задачи. В данной главе будут рассмотрены основные подходы к решению задачи планирования пути: методы с использованием карты окружающей среды или ее описания с помощью графа, методы на основе разбиения карты и методы, построенные на принципе потенциальных полей.

1.1 Алгоритмы глобального планирования

1.1.1 Алгоритмы планирования пути по карте

Все состояния робота характеризуются точкой в пространстве конфигураций. Вводят понятие пространства свободных конфигураций (C_{free}) – множество тех состояний объекта управления, проекция которых на рабочее пространство не пересекается со множеством препятствий. Граф отражает состояния, в которых может находиться робот, где каждый узел представляет одно состояние робота (координата положения, угол ориентации, скорость). Таким образом, алгоритмы движения по карте позволяют определить набор маршрутов в свободном пространстве C_{free}.

Существует ряд методов, которые отличаются преимущественно выбором узлов соответствующего графа. Подобные методы обычно используют в случаях полноты априорной информации об окружающей среде и ее неизменности. Ниже рассмотрим некоторые из предложенных методов: алгоритм Дейкстры, метод, основанных на использовании графа взаимовидимых позиций, A*, метод Монте-Карло (на основе выборки) и др.

Алгоритм Дейкстры в качестве входных данных получает взвешенный граф с неотрицательным весом ребер. Принцип работы метода можно охарактеризовать следующим образом: вершине графа ставится в соответствие метка, которая представляет собой стоимость пути в эту вершину из начальной. Работа алгоритма выполняется пошагово. На каждом из шагов в рассмотрение берется одна вершина и ставится цель уменьшить значение метки смежных узлов. После перебора всех вершин работа алгоритма завершается. На выходе получаем наикратчайший путь в данном графе.

При использовании графа видимости (графа взаимовидимых позиций) за узлы принимают множество вершин препятствий, ребра же, являющиеся доступными переходами между позициями, образуются такими парами узлов, что соединяющая их прямая не пересекает препятствия. Задача поиска пути

решается с помощью алгоритмов поиска в графах (например, Дейкстры). Строится в пространствах с полигональными препятствиями, то есть необходимо, чтобы препятствия представляли собой многоугольники в рабочем пространстве размерностью 2D либо многогранники в трехмерном случае. При движении в неизвестном окружении робот может в ходе перемещения дополнять граф видимости новыми вершинами по мере того, как они будут открываться. Данный подход может быть использован, если у робота достаточно широкий угол обзора. Рассматриваемый алгоритм имеет ряд недостатков, обусловленных тем, что часть спланированного пути совпадает с краями препятствий, а значит присутствует определенная опасность столкновения. Ещё одним значительным минусом является быстрый рост сложности графа с увеличением количества препятствий. Последняя проблема решается путем применения метода динамической диаграммы видимости [18] или концепции т-вектора (traversability vector) [19]. Данные подходы направлены на определение того, какие из препятствий в полной окружающей среде необходимо учитывать во время движения робота, тем самым они позволяют сократить сложность графа взаимовидимых позиций.

Другой подход описан в методе структурирования свободного пространства (free space structuring method) [20]. Он приводит к построению допустимого пути, максимально удаленного от препятствий. Применяя данный метод в двумерном пространстве, выделяют отрезки, соединяющие вершины препятствий и не пересекающиеся с препятствиями. Такие отрезки принято называть свободными звеньями (free links). Из свободных звеньев собираются выпуклые многогранники, описывающие области свободного пространства. Далее эта информация представляется в виде специального графа (maklink), а спланированный путь в обход препятствий строится алгоритмами поиска путей в графе. Метод структурирования свободного пространства дает более безопасный путь с точки зрения столкновения с препятствиями, но приводит к увеличению общей длины пути.

Алгоритм A^* [21] представляет собой эвристическое уточнение алгоритма Дейкстры. В отличие от классического метода производится перебор только наиболее вероятных вершин. Для этого принимают во внимание не только стоимость прохождения по пути от текущей вершины к находящейся по соседству, но и эвристическую функцию приоритета (в простейшем случае – это расстояние от рассматриваемой точки до конечной точки), что значительно сужает область поиска. Таким образом, при обходе препятствия робот будет выбирать наиболее выгодные, с точки зрения заданной эвристики, позиции. Причем выбранная эвристика оказывает влияние на время работы алгоритма: благодаря эффективно подобранной эвристике может достигать высокой производительности по времени. Алгоритм A^* всегда находит оптимальный путь, если эвристическая функция не переоценивает расстояние от узла до цели.

Суть алгоритма на основе выборки заключается в том, что из пространства свободных конфигураций выбирается N конфигураций, которые используются как контрольные точки для создания плана движения (roadmap) [22]. План движения соединяет две точки, если отрезок, образуемый этими точками, полностью лежит в пространстве свободных конфигураций. Для того, чтобы проложить маршрут к цели из начального положения, их добавляют в план движения. Если отрезок, образуемый точками, не принадлежит или частично принадлежит C_{free}, то в план движения добавляются дополнительные контрольные точки.

Данный алгоритм часто используют в многомерных пространствах конфигураций, так как скорость его работы не зависит от размерности пространства. Если рассматривать применение алгоритма в двумерном случае, то составленный план движения будет представлять собой граф видимости, описанный выше. В подобной ситуации достаточно оставить в графе видимости, только являющиеся касательными к препятствиям в обеих точках ребра.

Методы на основе графов удобно использовать в статической окружающей среде, поскольку построение графа подразумевает использование полной информации об окружающей среде, которая должна быть известна заранее. Наличие динамических препятствий в этих методах трудно интерпретируется и не позволяет строить граф.

Большинство методов, направленных на поиск кратчайшего пути, решают задачу таким образом, что траектория движения робота из начальной точки до достижения конечной представляет собой набор прямых линий. Очевидно, что полученная таким образом ломаная траектория не может быть точно отработана СУ робота. Более того, для увеличения точности подхода к целевой точке приходится снижать скорость, увеличивая тем самым время работы.

1.1.2 Алгоритмы на основе пространственной декомпозиции

Клеточная декомпозиция при планировании движения предполагает разбиение свободных областей сцены на множество простых ячеек (клеток) на основе тех или иных методов декомпозиции, определение смежности полученных ячеек и формирование графа связности, который в дальнейшем может использоваться для навигации в сцене. Данный подход редуцирует вычислительно сложную задачу планирования пути в евклидовом пространстве до типовой задачи поиска пути в графе. Выделяют приближенную и точную клеточные декомпозиции [23].

Приближенная клеточная декомпозиция реализуется с помощью сетки, покрывающей весь объем пространства конфигураций, причем ячейки сетки имеют одинаковый размер. Каждая клетка характеризуется статусом занятости: она может быть свободна (доступна), заполнена или же частично заполнена объектами сцены [24]. Маршрут в данном случае строится на основе анализа смежных доступных ячеек, выбора направлений для распространения и проверки принадлежности объекта управления доступным ячейкам сетки.

Таким образом, вместо исходной точной геометрической модели сцены используется ее упрощенное дискретное представление, что упрощает процесс определения маршрута следования. Сетку достаточно просто создавать использовать и поддерживать, но для таких алгоритмов очень важными характеристиками являются разрешение сетки и ограничения на виды переходов (эти параметры влияют на точность и быстродействие алгоритма). С уменьшением размера клеток возрастает объем необходимых вычислений, что является существенным недостатком метода сеток. Для борьбы с данной проблемой были созданы методы регулярной адаптивной декомпозиции, в числе которых дерево квадрантов в двумерном случае или дерево кубов в трехмерном [25-27].

Альтернативу приближенной клеточной декомпозиции составляет точная. Здесь окружающую среду делят на клетки, используя грани препятствий, что позволяет устранить проблемы, обусловленные погрешностью дискретизации пространства. Известны такие подходы точной клеточной декомпозиции, как метод вертикального разбиения или трапецеидальная декомпозиция [28], триангулированное разбиение [29], а также декомпозиция Морса [30].

Как правило, при решении задачи планирования по методам на основе клеточной декомпозиции или на основе графов необходим дополнительный этап построения пути, состоящий в поиске путей на графе. Можно использовать разные стратегии поиска относительного оптимального решения. Эти стратегии делят на классические (неинформированные) и эвристические (информированные). К неинформированным стратегиям поиска относятся: поиск в ширину, поиск по критерию стоимости, поиск в глубину, поиск с ограничением глубины, поиск в глубину с итеративным углублением и двунаправленный поиск [31].

Эвристические стратегии поиска (типичный пример – A^*) не имеют строгого обоснования, однако позволяют обнаружить приемлемое решение задачи в большинстве практически значимых (ценных) случаев.

Особенностью эвристической стратегии являются эвристические правила, которые формулируются для конкретной задачи. Эти правила способствуют повышению эффективности алгоритма планирования по сравнению с соответствующей классической стратегией.

В целом, недостатком любого из описанных выше видов клеточной декомпозиции является ограниченность количества вариантов направления, вызванная структурой сетки. На борьбу с данным ограничением направлены такие подходы, как D* (динамический A*) [32] или Theta* [33].

Алгоритмы типа D* разделяют карту на сегменты-ячейки, каждый сегмент содержит указатель на оптимальное направление движения и стоимость перехода на соседние ячейки. Когда алгоритм обнаруживает препятствия или иные изменения карты, то он добавляет новые сегменты, одновременно перестраивая маршрут. Таким образом, алгоритм D* представляет собой усовершенствованный с помощью перепланировки алгоритм A*. Алгоритм D* имеет ряд преимуществ перед A*, а именно: использование меньшего количества клеток, за счет того, что планируется не весь путь до цели, применение более эффективных расширений, сокращение количества расширенных узлов и меньшее время, необходимое для вычислений. Динамическая перестройка применяется также в алгоритмах Lifelong Planning A* (LPA*) [34], Focused D* (сочетает идеи A* и D*) [32] и D* Lite (построен на основе LPA*) [35].

1.1.3 Методы с использованием потенциальных полей

Общая идея методов потенциальных полей состоит в движении робота вдоль векторных линий векторного поля, потенциальная функция которого отражает конфигурацию препятствий и их форму, а также цель движения. Указанный подход подходит как для двумерного, так и для трехмерного случаев. Существующие методы различаются потенциальной функцией, по

типу которой можно выделить виртуальное силовое поле (virtual force field, VFF) [36], ньютоновское потенциальное поле [37], супербиквадратное потенциальное поле и гармоническое векторное поле [38].

Наиболее же известным является метод искусственных потенциалов (artificial potential field, APF) [39]. Его алгоритм основывается на простом (что упрощает контроль процесса в режиме реального времени), но эффективном принципе. Векторное поле разделяется на две составляющие: цель движения представляется притягивающим векторным полем, препятствия же – отталкивающим. Сложение векторных полей позволяет решить две задачи: движение к заданной целевой точке и обход препятствий. В свою очередь отталкивающее векторное поле есть сумма составляющих, каждое из которых описывает отдельное препятствие. Достоинством алгоритма является малая вычислительная стоимость, однако существует вероятность попадания робота в локальный минимум. Для того, чтобы не допустить этого, данный алгоритм часто комбинируют с диаграммой Вороного [40], где за узлы разбиения принимают все локальные минимумы. Также присутствует неопределенность в построении потенциальных функций, связанная с выбором коэффициентов функции. Эти проблемы в совокупности ограничивают широкое применение метода искусственных потенциалов в решении практических задач.

Стоит упомянуть, что такой подход применим в условиях статического окружения. В работе [41] представлена модификация метода искусственных потенциалов для динамической среды, когда препятствия и целевая точка могут изменяться.

Метод виртуального силового поля сочетает идеи сеток и метод потенциального поля. Указанный алгоритм применим для локального планирования, но обладает недостатками, обусловленными существенными изменениями свойств отталкивания препятствий, возникающих из-за дискретизации. То есть данный метод не позволяет планирование прохождения узких областей (например, дверной проем), а также приводит к колебательному движению в областях, наподобие коридоров.

Метод гистограммы векторного поля (vector field histogram, VFH) [42] способен преодолеть описанные недостатки. В настоящее время он является одним из наиболее популярных для локального планирования, и используется в режиме управления реального времени. Работа алгоритма осуществляется в три основных этапа: составление сетки на двумерной декартовой плоскости, описывающей препятствия вокруг робота, перестроение сформированной гистограммы на одномерную полярную. Далее осуществляется выбор наиболее подходящего сектора с низкой плотностью препятствий, и вычисляется угол поворота в этом направлении.

Существуют некоторые улучшенные модификации метода VFH: VFH+ [43] и VFH* [44]. Метод VFH+ учитывает размер робота, динамические ограничения и соответствующее расширение размеров препятствий, увеличивает гладкость спланированной траектории. Метод VFH* считается улучшением VFH и VFH+. Данный подход, учитывая глобальную информацию об окружении, выбирает наилучшее направление движения с использованием алгоритма A*. VFH* подходит для работы с неточной информацией, благодаря использованию вероятностного подхода, и считается быстрым и надежным в среде с большим количеством препятствий, однако существуют неразрешимые ситуации при наличии U-образных и симметричных препятствий.

1.2 Алгоритмы локального планирования

Практически все рассмотренные алгоритмы (за исключением D*, LPA*, VFH и некоторых других, упомянутых выше) и подходы в основном применяются для планирования пути при известном и частично известном окружении, то есть предназначены для глобального планирования. Подходы данного типа не предназначены для использования в чистом виде при движении в неизвестном окружении: это либо невозможно, либо требует

много вычислительных затрат. Для решения задачи планирования в условиях неизвестного окружения необходимо или модифицировать рассмотренные алгоритмы, или использовать подходы специального назначения. Такие алгоритмы, в конечном итоге, представляют из себя различные стратегии выбора решения при той или иной конфигурации пространства и используют некоторые идеи описанных выше алгоритмов (эвристические оценки, графы видимостей). Наиболее распространенная группа таких подходов – это Bug-алгоритмы [45, 46]. Ниже рассмотрим несколько их модификаций, а также метод поиска промежутков (Follow the Gap Method, FGM) [47].

1.2.1 Семейство Bug – алгоритмов

Алгоритм поворота (Bug0). Робот движется по прямой линии до цели. Обнаружив препятствие, робот запоминает своё положение и продолжает движение, но уже влево вдоль препятствия с целью поиска «прохода». Когда «проход» найден робот продолжает свой путь либо возвращаясь на ранее спланированную траекторию, либо генерируя новую, исходя из текущего положения. Если робот вернулся в точку выбора, то он начинает движение вправо. Алгоритм может быть модифицирован таким образом, чтобы выбор стороны поворота осуществлялся на основе реальной ситуации – например, с помощью некоторой оценки возможной длины препятствия по обе стороны от объекта управления. Подход не универсален, так как возможны случаи, когда агент не находит путь до цели.

Алгоритм Bug1. Робот осуществляет движение по направлению к цели до встречи с препятствием. После этого запоминает эту позицию, поворачивает налево и продолжает путь по контуру препятствия, непрерывно измеряя свое расстояние от текущего положения до цели, до возврата в позицию встречи с препятствием. Совершив полный оборот вокруг препятствия, робот движется в точку, расстояние от которой до цели оказалось минимальным среди возможных, и из этой точки начинает движение к цели по

прямой. Если наиближайшая точка к цели совпадает с точкой встречи с препятствием, то пути к цели не существует. Алгоритм всегда находит путь, если он существует.

Алгоритм Bug2. Развитие Bug1, направленное на устранение необходимости полного перебора точек вокруг препятствия. Отличие состоит в том, что робот двигается вдоль препятствия до пересечения с изначальной траекторией движения (при встрече с препятствием запоминает направление вектора на цель), а не в поисках самой близкой точки к цели, что уменьшает проходимый путь. Однако, предпочтительно использовать не сам алгоритм Bug2 в чистом виде, а его модификации, которые смогут гарантировать верный результат при объезде невыпуклых препятствий, а вместе с тем и безопасное движение в условиях неизвестного окружения.

Алгоритм Tangent Bug. Предыдущие алгоритмы не предусматривали способности робота определять препятствия дистанционно: движение к цели осуществлялось «на ощупь», что значительно понижало оптимальность выбранного пути. Чтобы учесть возможность определения препятствий на расстоянии, был создан алгоритм, строящий локальные графы касательных. Алгоритм оперирует следующими понятиями: точка разрыва (препятствие/граница видимости) и интервал непрерывности (определяется двумя точками разрыва). Точки разрыва находятся на краях препятствий.

Для применения метода Tangent Bug робот оснащается дополнительными датчиками, которые определяют расстояние до препятствий, и предварительная карта местности не требуется. Из своей текущей позиции робот строит касательные к краям препятствий, и далее представляет данные о расстоянии до препятствия и угла поворота к ним в виде локального графа касательных, который используется для выбора оптимального пути в сторону цели и во время обхода препятствий.

Робот движется по направлению к цели по прямой до тех пор, пока не обнаружит препятствие на пути (произойдет пересечение интервала непрерывности прямой). После этого объект управления начинает движение к

той точке разрыва, которая обладает минимальной эвристической оценкой (например, к той, сумма расстояний от которой до робота и до цели минимальна). Процесс движения робота сопровождается получением новых точек разрыва и движением к ним. Если робот достиг локального минимума (т.е. эвристическая оценка перестала уменьшаться), то дальнейшее движение осуществляется вдоль границы препятствия с сохранением направления. Движение вдоль границы прекращается при условии, что препятствие больше не мешает проходу к целевой точке.

Данный подход требует память для хранения графа касательных, но эти требования незначительны, по сравнению с методами Bug1 и Bug2, когда требуется память для хранения всех посещенных точек во время обхода препятствия.

1.2.2 Метод нахождения промежутков

Метод нахождения промежутков ведёт робота через центр максимального проёма между препятствиями. Робот, обнаружив препятствие на пути к цели, рассчитывает расстояние до препятствия и записывает его в массив. В массиве хранятся расстояния до видимых препятствий. Организуется также массив, предназначенный для хранения расстояний между препятствиями, так называемых проёмов, из которых выбираются самые широкие. Если значения совпадают, то выбирается первое из них. На следующем этапе необходимо вычислить угол между самым широким проёмом и направлением движения. Приоритетным считается направление, на котором отсутствуют препятствия, нежели направление, ближайшее к цели. Метод может учитывать кинематические ограничения мобильного робота.

Недостатком описанного подхода является то, что он не применим в ситуации с U-образным препятствием. Для преодоления данного недостатка была разработана модификация, названная интеллектуальным методом нахождения промежутков (Intelligent Follow the Gap Method, IFGM) [48].

Главное отличие от оригинального метода – это то, существует два варианта движения автономного робота. Величина самого широкого проема (max.gap) и рассчитанное расстояние до видимых препятствий (obs.dist) сравниваются со значением (d_{th}), превышающим ширину робота квадратной формы или радиус робота, и со значением (d_{obs}), превосходящим скорость робота, соответственно:

$$obs.dist > d_{obs},$$

$$max.gap > d_{th}.$$

Если условия выполняются, то движение робота осуществляется с помощью алгоритма FGM, в противном случае – Intelligent Bug Algorithm (IBA) [49]. Алгоритм IBA используется в случаях, когда достигается локальный минимум. Интеллектуальный метод нахождения промежутков применим в ситуациях с U и H – образными препятствиями, и не требует априорной информации об окружении.

1.3 Нечеткая логика

Элементы нечеткой логики могут применяться как для комбинации различных типов поведения [50], так и для реализации отдельного типа [51]. Например, используя инструмент нечеткой логики, можно строить маневры подводного аппарата для достижения цели, находящейся на вершине (крутой горки) препятствия.

Применение нечеткой логики строится на использовании нечетких множеств, которые характеризуются функциями принадлежности. Такие функции соответственно отражают степень принадлежности некоторого элемента множеству и могут принимать значения от 1 (“полностью принадлежит”) до 0 (“полностью не принадлежит”).

1.4 Выводы

Алгоритмы планирования пути по карте предназначены преимущественно для использования в условиях статического известного окружения, однако некоторые из них могут быть преобразованы для использования в динамической среде. Такие методы имеют достаточно простую практическую реализацию и направлены, как правило, на поиск кратчайшего пути. Результирующая траектория представляет собой набор ломаных линий, что накладывает ограничения на допустимую скорость движения робота. С увеличением количества препятствий наблюдается рост сложности графа. Спланированный путь, в связи с особенностями некоторых методов группы, может совпадать с краями препятствий. Существующие модификации способны сделать маршрут безопасным, но приводят к увеличению его длины. Таким образом, подходы данной группы дают относительно оптимальные решения.

Методы на основе пространственной декомпозиции используют упрощенное дискретное представление исходной точной модели геометрической сцены, что упрощает процесс определения маршрута следования. Однако, разрешение сетки оказывает прямое влияние на объем вычислений: с уменьшением размера клеток уменьшается также быстродействие алгоритма, что делает его неэффективным для реализаций, функционирующих в условиях реального времени. Как правило, методы данной группы требуют дополнительного этапа, состоящего в поиске пути на графе. Использование для этого эвристических (информативных) стратегий способно повысить эффективность алгоритма.

Методы поиска пути на основе потенциальных полей подходят как для двумерного, так и для трехмерного случаев. Позволяют строить гладкие траектории, тем не менее они могут привести к резкому изменению направления движения, которое не может быть точно отработано СУ робота. Подходы такого рода не позволяют управлять траекторией робота в процессе

его движения. При использовании данного метода сталкиваются с проблемой попадания в локальный минимум.

Указанные выше алгоритмы не предназначены для использования в чистом виде при движении в неизвестном окружении, поэтому используют подходы специального назначения, решающие задачу локального планирования. Рассмотренные подходы Bug1 и Bug2 не предусматривают способности робота определять препятствия дистанционно и требуют памяти для хранения всех посещенных точек во время обхода препятствия. Алгоритм Tangent Bug предполагает оснащение роботами датчиками. Ему необходима память для хранения графа касательных, но эти требования незначительны. Алгоритм поиска промежутков также предполагает наличие бортовых датчиков. Недостатком данного метода является то, что он не применим в ситуации с U и H – образными препятствиями. Нечеткая логика может выступать в качестве связующего звена между различными методами или использоваться обособленно для реализации отдельного поведения.

2 Описание алгоритма

2.1 Постановка задачи

При разработке метода планирования маршрута мобильных роботов, будем полагать, что траектории их движения формируются заблаговременно. Эти траектории проходят через набор путевых точек $W = \{W_1, W_2, \dots, W_n\}$, где $W_i = (x_{wi}, y_{wi})^T$, и описываются сплайнами Безье третьего порядка. Ввиду того, что рельеф поверхности заранее неизвестен, исходную траекторию движения мобильного робота описывают в горизонтальной плоскости. Таким образом, координата z каждой точки заранее спланированной траектории должна формироваться непосредственно в процессе движения с тем, чтобы обеспечить движение робота с заданной высотой над поверхностью.

Для сбора информации об окружении мобильный робот оснащают набором бортовых датчиков, основываясь на показаниях которых составляется представление об удаленности препятствий от объекта управления в направлении соответствующего датчика. В качестве бортовых датчиков могут выступать 3D-сонары, доплеровские лаги (дли подводных роботов), всевозможные датчики приближения (ультразвуковые или лазерные). В контексте рассматриваемого метода все датчики, интегрированные в робота, делятся на две группы, а именно: сенсоры, которые измеряют расстояние до объектов, расположенных вокруг мобильного робота, и те, что измеряют расстояние до поверхности. Основываясь на показаниях датчиков, непрерывно формируются векторы $D = (d_1, d_2, \dots, d_i, \dots, d_m)$ и $H = (h_1, h_2, \dots, h_j, \dots, h_n)$, где m и n – это количество соответствующих датчиков. В дальнейшей работе будем полагать, что объект управления оснащен четырьмя датчиками, предназначенными для измерения расстояния до поверхности. Эти сенсоры направлены вперед, назад, направо и налево по отношению к мобильному роботу (см. рис. 2.1.1).

Предположим, что данные, поступающие от бортовых датчиков, удовлетворяют следующим условиям:

а) Нулевой выход i-го датчика информирует о том, что препятствие в соответствующем направлении не обнаружено;

б) Дальность обнаружения препятствий датчиками ограничена значением d_{max}, которое может быть различным для отдельных групп датчиков, но принимается идентичным для простоты;

в) Время заполнения векторов D и H показаниями независимых датчиков не превышает период дискретизации системы управления роботом;

г) Ориентация каждого бортового датчика относительно системы координат, закрепленной в объекте управления, полагается известной и неизменной в процессе движения робота. В данной работе предполагается, что датчики каждой группы расположены симметрично относительно оси бортовой качки (характеризуется углом крена) мобильного робота.

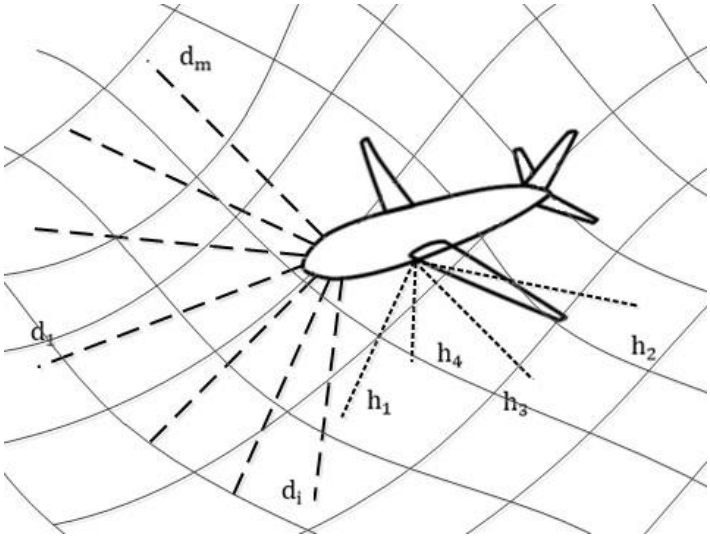

Рисунок 2.1.1 - Сигналы бортовых датчиков дальности

Исходя из того, что перемещение осуществляется в окружении, априорная информация о котором отсутствует или является недостаточной, путь будем формировать заранее, используя для этого набор базовых точек, координаты которых будут корректироваться в процессе движения

мобильного робота по мере поступления информации об окружении от бортовых датчиков. Тем самым к процессу формирования пространственных траекторий предъявляется ряд требований:

а) Коррекция указанных траекторий в процессе движения робота не должна влечь за собой резкие изменения направления или скорости перемещения;

б) Сгенерированный путь должен обеспечивать движение робота через все путевые точки в заданном порядке;

в) Сформированный маршрут должен быть проложен на безопасном расстоянии от обнаруженных препятствий, которое определяется динамическими характеристиками объекта управления. Это расстояние характеризуется величиной D_{min};

г) Необходимо, чтобы сгенерированная траектория на всем своем протяжении располагалась на заданной высоте H_d от поверхности, над которой осуществляется перемещение;

д) Движение вдоль проложенного маршрута не должно приводить к превышению максимально разрешенного значения угла тангажа φ_{max}. Данное ограничение продиктовано особенностями конструкции и двигательных (моторных) систем различных типов роботов.

В качестве основного метода формирования траектории использован метод, описанный в [52]. На основе указанного метода формируется и подстраивается в процессе движения гладкая траектория движения мобильного робота, описываемая в горизонтальной плоскости сплайнами Безье третьего порядка, в соответствии с показаниями датчиков. Такой подход обеспечивает безопасное движение робота в заранее неизвестном окружении, содержащем препятствия. Однако, для того, чтобы указанный метод возможно было применять в условиях пространственного перемещения, необходимо обеспечить возможность обхода препятствий в вертикальной плоскости и

движения на заданной высоте от поверхности, рельеф которой заранее неизвестен.

Таким образом, в работе решается указанная проблема. Зададим исходную гладкую траекторию движения мобильного робота, проходящую через набор опорных точек W, в двумерном пространстве. Объект управления совершает движение в заранее неизвестном окружении и оснащен датчиками, формирующими векторы D расстояний до обнаруженных препятствий и H расстояний до поверхности, над которой осуществляется перемещение. Необходимо, чтобы при коррекции исходной траектории сохранялся набор точек, через которые будет проложен результирующий путь (новый путь должен проходить через каждую точку указанной последовательности), высота H_d над поверхностью движения оставалась неизменной, а расстояние до обнаруженных препятствий – безопасным (не менее D_{min}).

Задача коррекции исходной траектории решается в несколько этапов:

а) Анализ окружения на основе информации, поступающей от бортовых датчиков;

б) Анализ расположения препятствий по отношению к мобильному роботу и выбор стратегии обхода препятствия;

в) Расчет параметров новой траектории на основе алгоритма, описанного в [52], если было принято решение о необходимости осуществления коррекции.

В первую очередь следует оценить рельеф поверхности, и рассчитать координаты опорной точки X_d, которая задает движение робота. Координаты данной точки вычисляются на основе текущей траектории в горизонтальной плоскости и установленной скорости движения мобильного робота, вдоль сформированного пути. Далее, основываясь на показаниях бортовых датчиков, необходимо оценить величину угла тангажа, достаточную для обхода препятствия в вертикальной плоскости. При условии, что рассчитанное значение окажется менее максимально разрешенного φ_{max}, можно сделать

вывод о возможности обхода препятствия в вертикальной плоскости, не выполняя при этом коррекцию траектории в горизонтальной плоскости. В противном случае будет принято решение о необходимости коррекции траектории в горизонтальной плоскости. В таком случае робот обходит препятствие сбоку, сохраняя неизменной высоту движения над поверхностью. Далее происходит процесс формирования новой траектории, если коррекция необходима. Если при выполнении обхода препятствия сбоку будет обнаружено, что робот отдаляется от целевой точки, то расчет координат новой траектории будет производиться таким образом, чтобы одновременно с обходом препятствий сбоку робот также начинал движение «в гору».

Далее каждый из шагов будет описан более подробно.

2.2 Алгоритм формирования пространственных траекторий

Прежде чем начнется работа непосредственно алгоритма формирования пространственных траекторий, информационная управляющая система выполняет расчет координат точек, обнаруженных бортовыми датчиками и лежащих на препятствиях и на поверхности движения. Указанные координаты с учетом пространственной ориентации мобильного робота вычисляются следующим образом:

а) Для датчиков d_i:

$$X_{di} = X + R_G(\theta, \varphi, \psi) R_{MR}(a_{di}, b_{di}, c_{di}) \begin{bmatrix} 0 \\ 0 \\ d_i \end{bmatrix}, i = (\overline{1, m}),$$

где $X = (x, y, z)$ – координаты мобильного робота в абсолютной системе координат; θ, φ, ψ – углы крена, тангажа и курса робота соответственно; a_{di}, b_{di}, c_{di} – углы ориентации i-го сенсора относительно осей x, y и z системы координат, закрепленной в объекте управления; $R_G(.), R_{MR}(.)$ – матрицы

вращения вдоль осей x, y и z абсолютной и связанной с мобильным роботом систем координат соответственно;

б) Для датчиков h_j :

$$X_{hj} = X + R_G(\theta, \varphi, \psi) R_{MR}(a_{hj}, b_{hj}, c_{hj}) \begin{bmatrix} 0 \\ 0 \\ h_j \end{bmatrix}, j = (\overline{1,4}),$$

где a_{hj}, b_{hj}, c_{hj} – углы ориентации j-го датчика, измеряющего расстояние до поверхности, в системе координат, закрепленной в мобильном роботе. Для определенности будем считать, что точка X_{h1} располагается перед роботом, точка X_{h2} – позади, а точки X_{h3} и X_{h4} по левую и правую стороны соответственно.

Когда координаты точек X_{hj} рассчитаны, можно переходить к вычислению координат опорной точки X_d, которая задает желаемое движение робота по траектории. Этот расчет осуществляется в два этапа. На первом этапе выполняется поиск желаемого значения угла φ_d, который обеспечит перемещение по спланированному пути на высоте над поверхностью, равной заданному значению H_d. На втором шаге выполняется расчет координат точки X_d с учетом заданной траектории в горизонтальной плоскости, значения угла φ_d и установленной скорости движения мобильного робота ν_d.

Точки X_{h1} и X_{h2} участвуют в расчете величины угла φ_d, при вычислении которого учитывается, что профиль рельефа может быть аппроксимирован прямой линией, проходящей через эти точки (см. рис. 2.2.1). Прежде всего необходимо рассчитать координаты точки X_z. Эта точка находится на высоте H_d от X_{h1} и расположена на плоскости, проходящей через X, X_{h1} и X_{h2} (см. рис. 2.2.1).

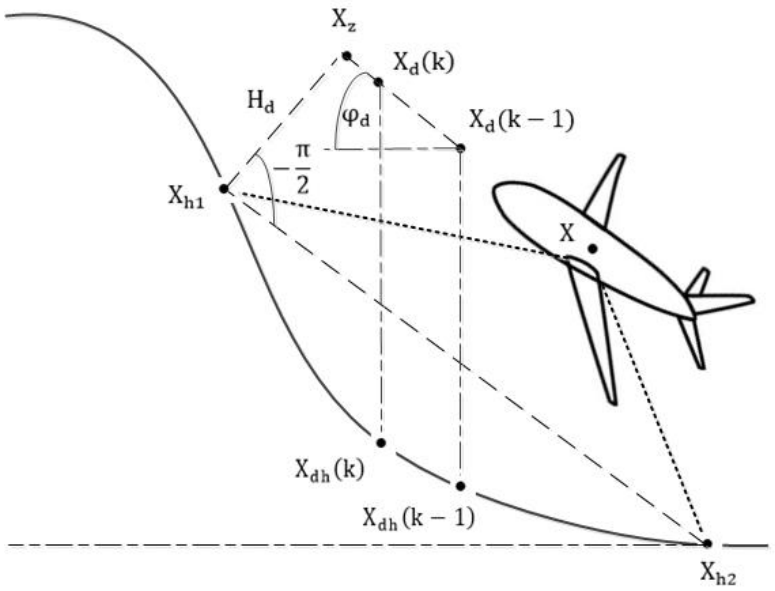

Рисунок 2.2.1 – Расчет координат опорной точки X_d

Координаты точки X_z можно вычислить следующим образом:

$$X_z = R_{\psi\theta\psi}\frac{X_{h1} - X_{h2}}{\|X_{h1} - X_{h2}\|}H_d + X_{h1},$$

где

$$R_{\psi\theta\psi} = R_G(0,0,\psi)R_G\left(0,-\frac{\pi}{2},0\right)R_G(0,0,-\psi),$$

$$R_{\psi\theta\psi} = \begin{bmatrix} \sin^2\psi & -\cos\psi\sin\psi & -\cos\psi \\ -\cos\psi\sin\psi & \cos^2\psi & -\sin\psi \\ \cos\psi & \sin\psi & 0 \end{bmatrix}.$$

Матрица $R_{\psi\theta\psi}$ описывает поворот вектора $(X_{h1} - X_{h2})$ на угол $\pi/2$ вокруг оси, которая перпендикулярна этому вектору и лежит в горизонтальной плоскости. Когда координаты точки X_z становятся известны, можно перейти к расчету угла φ_d, руководствуясь следующим выражением:

$$\varphi_d = \arctan\left(\frac{z_z - z_d(k-1)}{\|[x_z, y_z] - [x_d(k-1), y_d(k-1)]\|}\right),$$

где (x_z, y_z, z_z) и $\big(x_d(k-1), y_d(k-1), z_d(k-1)\big)$ – координаты точек X_z и $X_d(k-1)$ соответственно. Индекс (k-1) говорит о том, что необходимо использовать координаты точки X_d, рассчитанные на предыдущем шаге дискретизации системы управления роботом.

Далее с учетом величины угла φ_d производится расчет координат точки $X_d(k) = \big(x_d(k), y_d(k), z_d(k)\big)$:

$$X_{dh}(k) = (1-\delta(k))^3 W_1 + 3\delta(k)\big(1-\delta(k)\big)^2 P_1 + 3\delta(k)^2\big(1-\delta(k)\big)P_2 + \delta(k)^3 W_2,$$
$$X_{dh}(k) = f_x\big(\delta(k)\big), \qquad (2.2.1)$$
$$z_d(k) = z_d(k-1) + \tan(\varphi_d)\,\|X_{dh}(k) - X_{dh}(k-1)\|,$$

где $W_1 = (x_{W1}, y_{W1})$, $W_2 = (x_{W2}, y_{W2})$, $P_1 = (x_{P1}, y_{P1})$, $P_2 = (x_{P2}, y_{P2})$ – это координаты начальной, конечной и управляющих точек сплайна, описывающего текущий участок траектории движения мобильного робота в горизонтальной плоскости; $X_{dh} = (x_d, y_d)$ – проекция точки X_d на горизонтальную плоскость; $\delta = [1,0]$ – параметр, задающий движение X_{dh} вдоль сплайна от начальной точки W_1 в конечную точку W_2. Индексы k и $(k-1)$ означают, что для вычислений используются значения переменных, полученные на текущем и предыдущем шагах дискретизации системы управления мобильным роботом.

Из выражения (2.2.1) видно, что координаты точки X_d определяются параметром $\delta(k)$. Следовательно, для вычисления координат X_d необходимо найти такое значение $\delta(k)$, которое обеспечило бы перемещение точки X_d с заданной скоростью υ_d. Указанная задача решается с помощью итерационного алгоритма, процесс работы которого может быть разделен на три основных шага:

a) $S_d = \nu_d \Delta\tau, p = 0, \delta_t(p) = \delta(k-1), S(p) = 0, X_{dt}(p) = X_{dh}(k-1),$

где S_d – путь, пройденный точкой X_d с заданной скоростью υ_d, к моменту времени $\Delta\tau$; $\Delta\tau$ – интервал дискретизации системы управления роботом; p – индекс текущей итерации алгоритма.

б) Основной шаг расчета:

$$p = p + 1,$$
$$\delta_t(p) = \delta_t(p-1) + \Delta,$$
$$X_{dt}(p) = f_x\left(\delta_t(p)\right),$$
$$S_x = \|X_{dt}(p) - X_{dt}(p-1)\|,$$
$$S_z = \|\tan(\varphi_d)\|S_x,$$
$$S(p) = S(p-1) + (S_x^2 + S_z^2)^{1/2},$$

где Δ - инкремент δ_t.

в) Проверка выполнения условий завершения расчета.

Если $S(p) > S_d$, то принимаем $\delta(k) = \delta_t(p)$ и завершаем работу алгоритмы, в противном случае возвращаемся ко второму шагу.

Координаты точки X_d на текущем шаге дискретизации вычисляются согласно выражению (1) после определения $\delta(k)$. Для увеличения динамической точности, движение с заданной скоростью υ_d, можно обеспечивать, руководствуясь подходами, описанными в работах [53, 54].

На пути мобильного робота во время выполнения миссии могут встречаться препятствия, и алгоритм планирования пути должен обеспечивать движение объекта на безопасном расстоянии от них. Для этого на основе информации о точках X_{di}, $i = (\overline{1,m})$ и X_{hj}, $j = (\overline{1,n})$ необходимо определять наличие препятствий вблизи мобильного робота и выбирать один из доступных методов обхода: сверху, в вертикальной плоскости без коррекции траектории в горизонтальной, либо сбоку, с коррекцией траектории в горизонтальной плоскости согласно алгоритму, описанному в [52]. Выбор

производится путем оценки величины угла тангажа φ^*, необходимого для обхода препятствия в вертикальной плоскости.

Для расчета величины φ^*, весь набор датчиков d_i, которыми оснащен мобильный робот, объединяют в 3 группы (см. рис. 2.2.2): датчики, находящиеся по левую сторону мобильного робота $L = (d_1, \dots, d_l)$; датчики, расположенные спереди робота $F = (d_{l+1}, \dots, d_{l+f})$ и датчики, помещенные с правой стороны объекта управления $R = (d_{l+f+1}, \dots, d_{l+f+r})$, где $l + f + r = m$. Каждую из выделенных секций связывают с датчиками h_3, h_1 и h_4 соответственно (см. рис. 2.2.2).

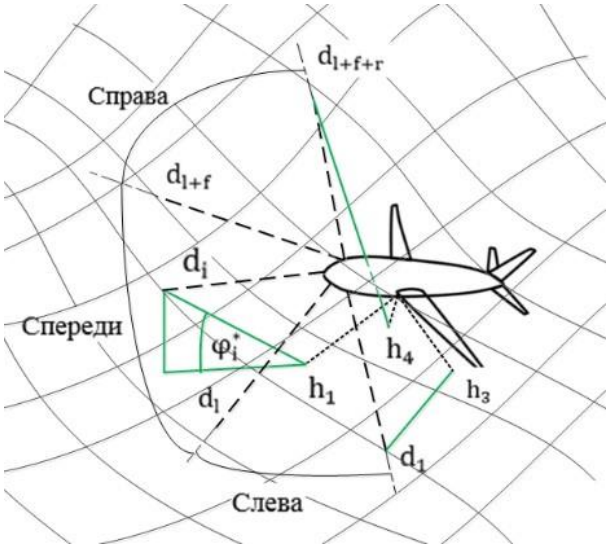

Рисунок 2.2.2 - Выбор стратегии обхода препятствия

Оценка величины угла тангажа φ^* рассчитывается для каждой пары датчиков d_i и h_j согласно следующим выражениям:

$$\varphi_i^* = \begin{cases} \arctan\left(\dfrac{z_{di} - z_{hj}}{\left\|[x_{di}, y_{di}] - [x_{hj}, y_{hj}]\right\|}\right), \text{if } d_i \neq 0 \\ 0, \text{if } d_i = 0 \end{cases},$$

$$i = \begin{cases} \overline{(1, l)}, \text{if } j = 3 \\ \overline{(l+1, l+f)}, \text{if } j = 1 \\ \overline{(l+f+1, l+f+r)}, \text{if } j = 4 \end{cases},$$

где (x_{di}, y_{di}, z_{di}) – координаты точки X_{di}, а (x_{hj}, y_{hj}, z_{hj}) соответственно точки X_{hj}.

Окончательно значение φ^* определяется как $\varphi^* = \max(\varphi_i^*)$, $i = (\overline{1, m})$. Если выполняется неравенство $\varphi^* \leq \varphi_{max}^*$, то обход препятствия возможен в вертикальной плоскости, и коррекция траектории в горизонтальной плоскости не требуется. Если же $\varphi^* > \varphi_{max}^*$, то обход препятствия необходимо осуществить в горизонтальной плоскости с коррекцией исходной траектории. При этом значение φ_{max}^* определено следующим выражением:

$$\varphi_{max}^* = \begin{cases} \min\left(\arctan\left(\frac{H_d}{D_{min}}\right), \varphi_{max}\right), \text{if } d_i \in L \text{ or } d_i \in R \\ \varphi_{max}, \text{if } d_i \in F \end{cases}, i = (\overline{1, m}). \quad (2.2.2)$$

Из (2.2.2) видно, что если препятствие находится слева или справа от мобильного робота, то величина φ_{max}^* зависит от отношения значений H_d и D_{min} и может быть меньше максимально разрешенного угла тангажа φ_{max}. Рисунок 2.2.3 поясняет этот факт.

На рисунке 2.2.3 (а) показано движение мобильного робота вдоль препятствия, которое имеет плавный наклон. В этом случае мобильный робот одновременно поддерживает неизменной высоту H_d движения над поверхностью и безопасным расстояние до препятствия (не менее D_{min}). Однако, если движение происходит вдоль препятствия, наклон которого изменяется резко (см. рис. 2.2.3 (б)), то становится невозможно осуществлять одновременно контроль заданной высоты над поверхностью и минимально допустимого расстояния до препятствия.

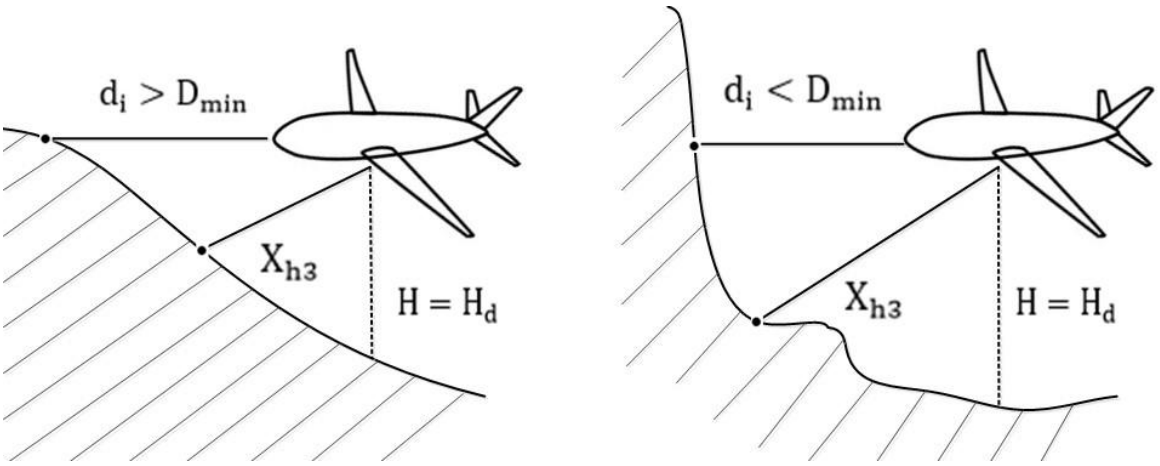

Рисунок 2.2.3 – Особенности движения вдоль различных препятствий

Когда мобильный робот начинает отодвигаться от препятствия с тем, чтобы боковое расстояние стало безопасным, происходит уменьшение высоты H_d, и робот опускается. При этом боковое расстояние снова уменьшается. Следовательно, для одновременного поддержания заданной высоты движения над поверхностью и безопасного бокового расстояния, при расчете φ^*_{max} должно учитываться отношение значений H_d и D_{min}.

Если принято решение об обходе препятствия сбоку, то коррекция траектории, которая в горизонтальной плоскости описывается сплайнами Безье третьего порядка, осуществляется на основе алгоритма, описанного в [52]. При этом в зависимости от положения препятствия относительно мобильного робота существуют две стратегии обхода: уклонение от препятствия (слева или справа) либо смещение траектории на безопасное расстояние, если это расстояние менее D_{min}. После коррекции текущий участок заранее спланированного пути описывается сплайном с новыми параметрами, которые используются в выражении (2.2.1) для вычисления координат точки X_d.

Метод формирования гладких траекторий в неизвестном окружении, предложенный в [15], не учитывает случая, когда цель находится на вершине препятствия с таким уровнем наклона, что обход сверху не представляется возможным, ввиду превышения максимально возможного угла тангажа φ_{max}.

В процессе выполнения дипломного проекта данная задача была решена с помощью применения элементов нечеткой логики.

В момент, когда на пути робота возникает такое препятствие, что $\varphi^* > \varphi_{\max}$, начинает функционировать нечеткий планировщик, и происходит расчет некоторых вспомогательных величин: текущего вектора положения мобильного робота, который пересчитывается, в дальнейшем, на каждом шаге дискретизации системы управления роботом и угла β, образованного начальным, рассчитанным в момент встречи с препятствием, и текущим векторами (см. рис. 2.2.4).

Нечеткий планировщик совмещает существующие стратегии обхода и позволяет роботу одновременно уклоняться от препятствия сбоку, поднимаясь на него. При этом для определения угла тангажа φ_d применяется комплементарный фильтр, который опирается на значения φ^* и $\varphi_{\max}$:

$$\varphi_d = \alpha\varphi^* + (1-\alpha)\varphi_{\max},$$

где $\alpha = [0,1]$ – коэффициент комплементарного фильтра, который является выходом нечеткого планировщика.

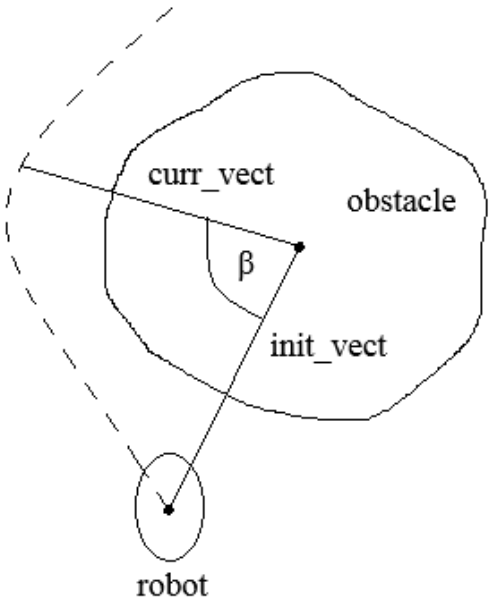

Рисунок 2.2.4 – Расчет угла β между векторами положения робота

Входными значениями нечеткого планировщика являются расстояние от робота до целевой точки (distance) и текущая величина контрольного угла β, на основании которых принимается решение о величине коэффициента α. Лингвистическая переменная distance описывается терм-множеством {N – близко, M-средне, F-далеко}, терм-множества лингвистических переменных β и α являются аналогичными и имеют вид: {S – маленький, M-средний, B-большой}. Вид функций принадлежности приведен на рисунках 2.2.5, 2.2.6 и 2.2.7.

Нечеткий планировщик оперирует набором простейших правил:

а) Если distance – N или β – B, то α – B;

б) Если distance – M или β – M, то α – M;

в) Если distance – F или β – S, то α – S.

На рисунке 2.2.8 приведена графическая интерпретация нечетких правил.

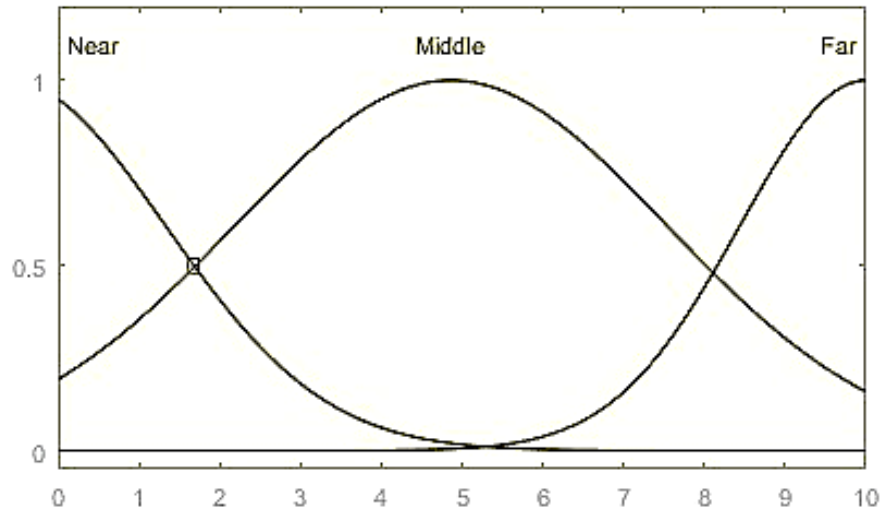

Рисунок 2.2.5 – Входная переменная «расстояние»

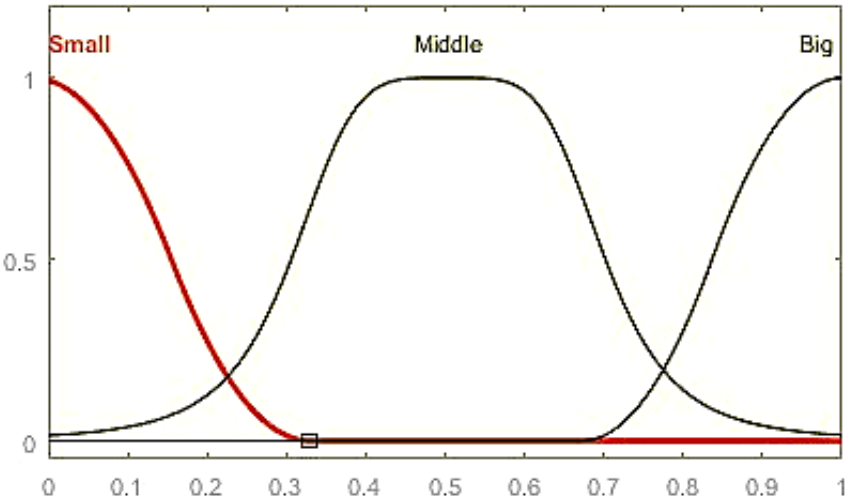

Рисунок 2.2.6 – Входная переменная β

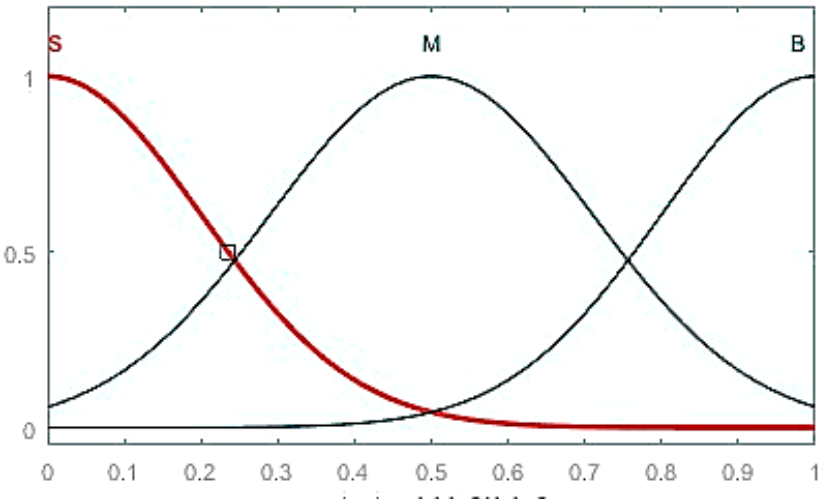

Рисунок 2.2.7 – Выходная переменная «альфа»

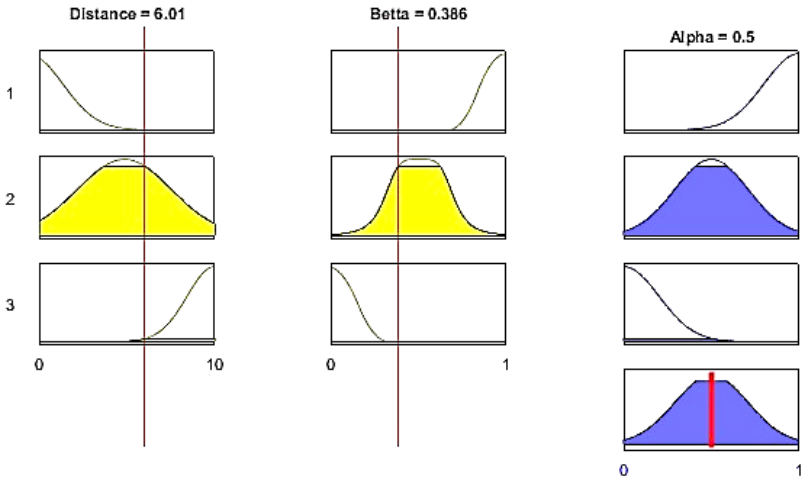

Рисунок 2.2.8 – Графическая интерпретация нечетких правил

Рассматриваемый метод применим в трехмерном пространстве, функционирует в условиях неизвестного окружения и учитывает динамические ограничения робота. Результирующие траектории являются гладкими в горизонтальной плоскости и могут быть точно отработаны системой управления роботом. Данный алгоритм позволяет прокладывать путь заранее на основе показаний бортовых датчиков, которые непрерывно анализируются в процессе движения. Он позволяет планирование прохождения узких областей, и не приводит к колебательному движению в областях, наподобие коридоров, как это случается в случае использования для планирования потенциальных полей. С помощью нечеткой логики обеспечивается комбинация стратегий обхода препятствий в непосредственной близости цели. Алгоритм характеризуется малой вычислительной сложностью и не требует большого объема памяти: она необходима, в основном, для хранения массивов показаний датчиков и препятствий, что является незначительным требованием.

3 Результаты моделирования

Для исследования описанного выше метода была произведена интеграция среды Matlab с робототехническим симулятором V-REP. В ходе исследования V-REP был использован для построения виртуальной окружающей среды, содержащей различные препятствия, а Matlab – в качестве платформы для реализации рассматриваемого метода планирования пути. Блок-схема алгоритма представлена на рисунках 3.1 – 3.3.

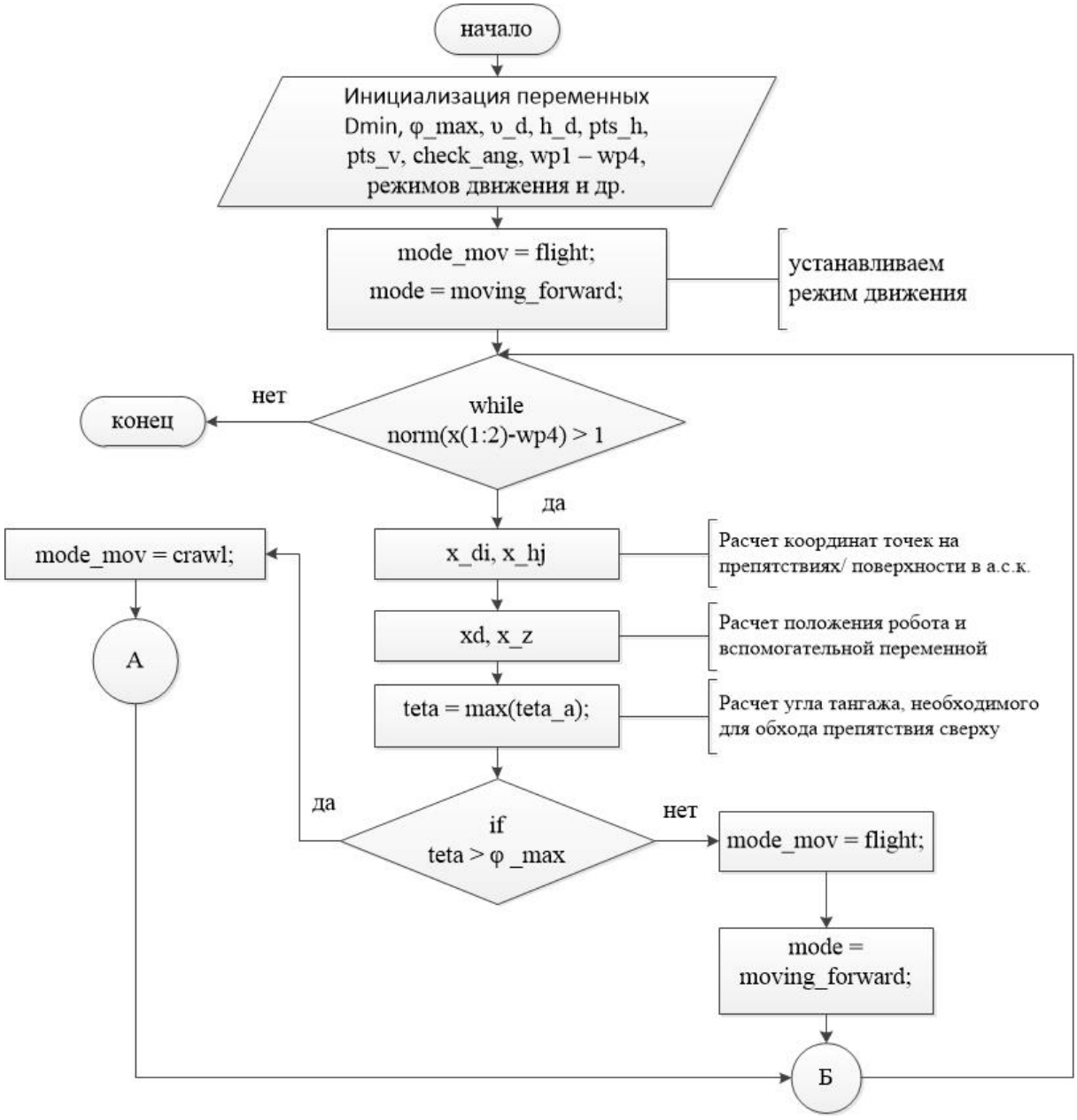

Рисунок 3.1 – Блок-схема алгоритма (часть 1)

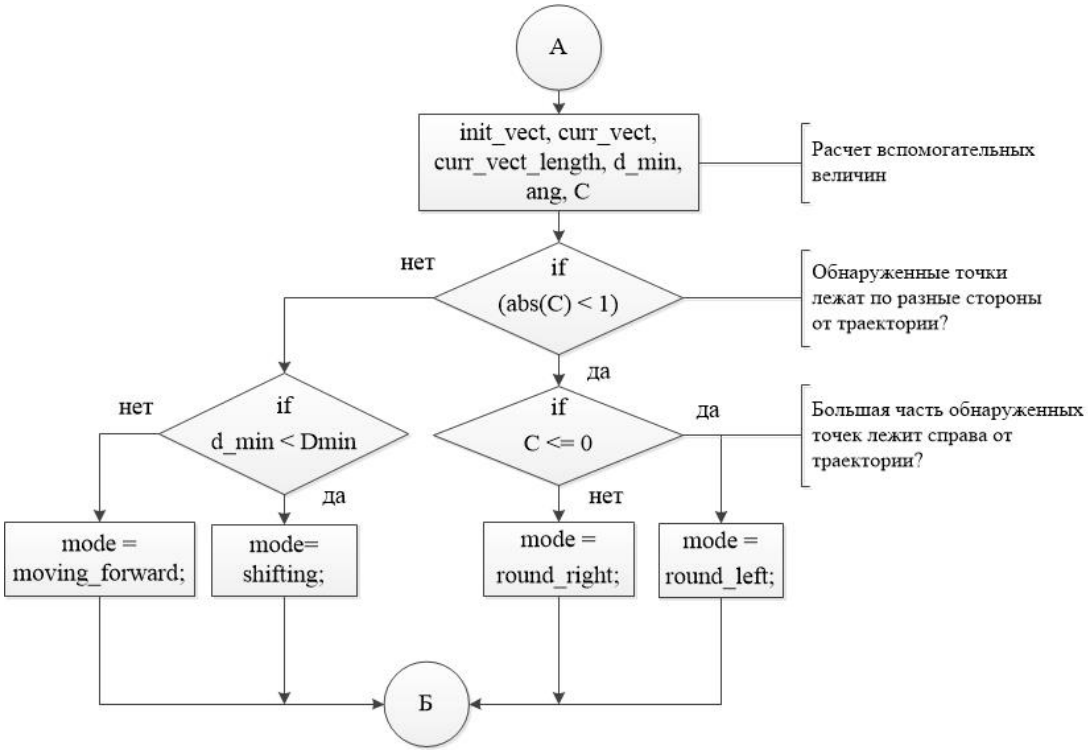

Рисунок 3.2 – Блок-схема алгоритма (часть 2)

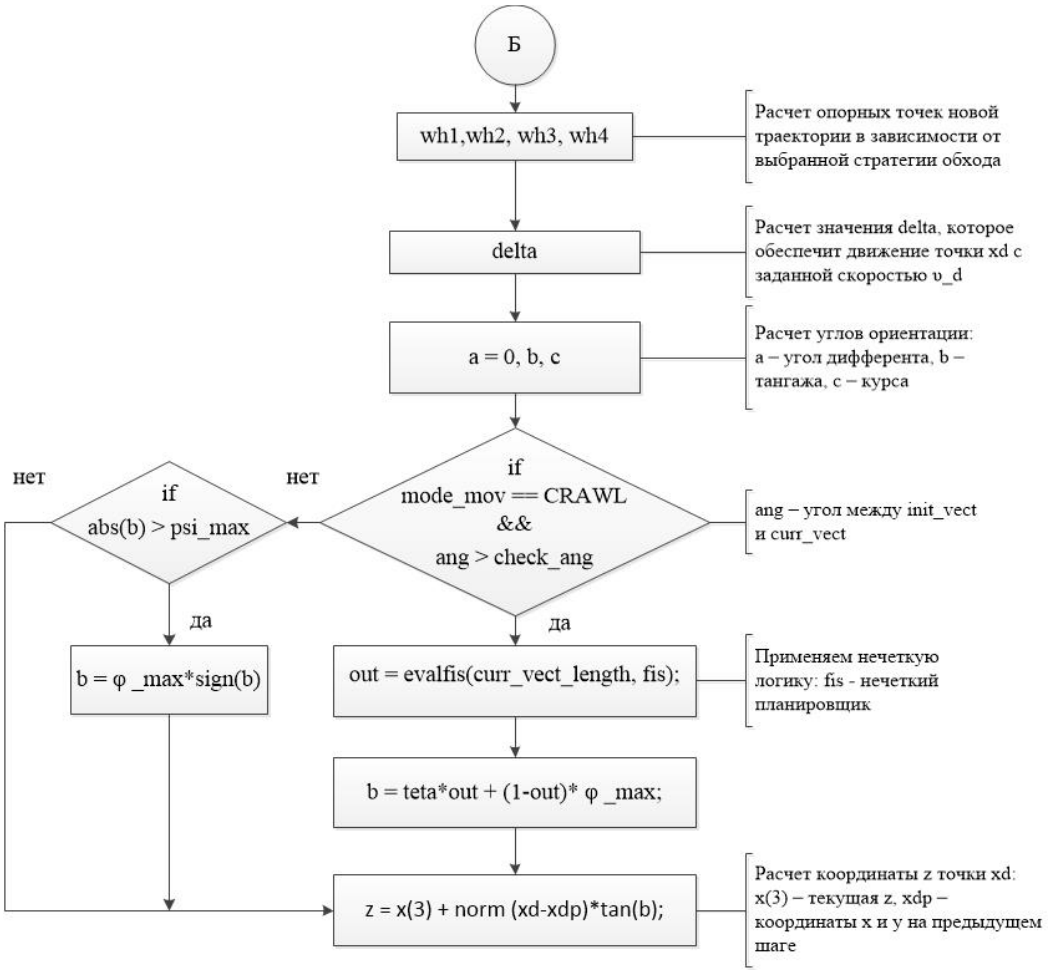

Рисунок 3.3 – Блок-схема алгоритма (часть 3)

Листинг программы приведен в Приложении А.

В процессе выполнения математического моделирования предполагалось, что мобильный робот оснащен семью датчиками, которые измеряют расстояние до препятствий (горизонтальные) и четырьмя – до поверхности (вертикальные). Вертикальные датчики расположены спереди, сзади и по обе стороны мобильного робота и отклонены от оси курса на угол $\pi/4$ в соответствующих направлениях. Эталонное значение высоты движения над поверхностью H_d составляло 2 метра, безопасное расстояние до препятствий $D_{min} = 2$ метра, максимально разрешенное значение угла тангажа $\varphi_{max} = 0.6$ радиан и заданная скорость движения мобильного робота была равна $\upsilon_d = 1$ м/с. Начальная W_1 и конечная W_4 точки заранее спланированной траектории имели координаты (44, 44) и (-38, 20) в абсолютной системе координат соответственно, а угол подхода к цели составлял $\pi/2$ радиан и обеспечивался особым выбором управляющих точек исходного сплайна. Высота z мобильного робота над поверхностью в начале движения имела значение 4 метра, далее на каждом шаге дискретизации системы управления роботом рассчитывалась таким образом, чтобы высота движения робота над поверхностью оставалась постоянной. При моделировании предполагается, что система управления мобильным роботом точно отрабатывает спланированную траекторию, а углы курса и тангажа задаются направлением касательных к этой траектории в вертикальной и горизонтальной плоскостях.

На рисунке 3.4 пунктирной линией изображена исходная траектория, заданная в горизонтальной плоскости, а сплошной – проекция результирующей на горизонтальную плоскость. На рисунке 3.5 проиллюстрировано, как изменялась координата z мобильного робота на протяжении всего пути (см. рис. 3.4). На рисунке 3.6 изображена виртуальная сцена в V-REP и путь, пройденный роботом в трехмерном пространстве (синяя линия).

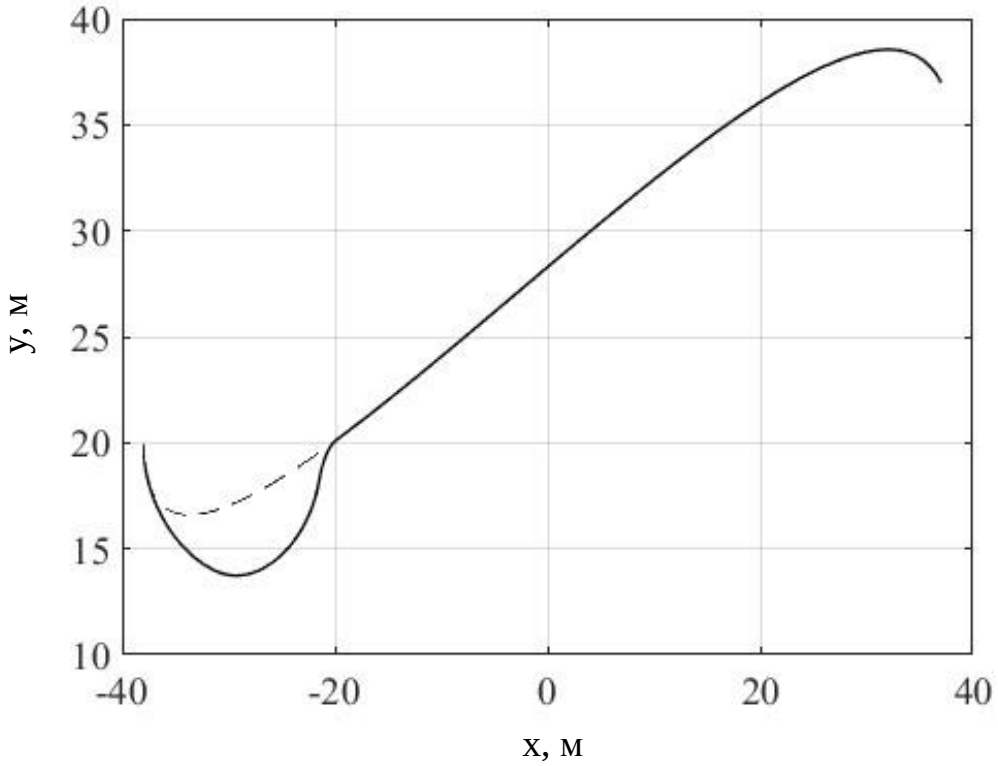

Рисунок 3.4 – Исходная и результирующая траектории

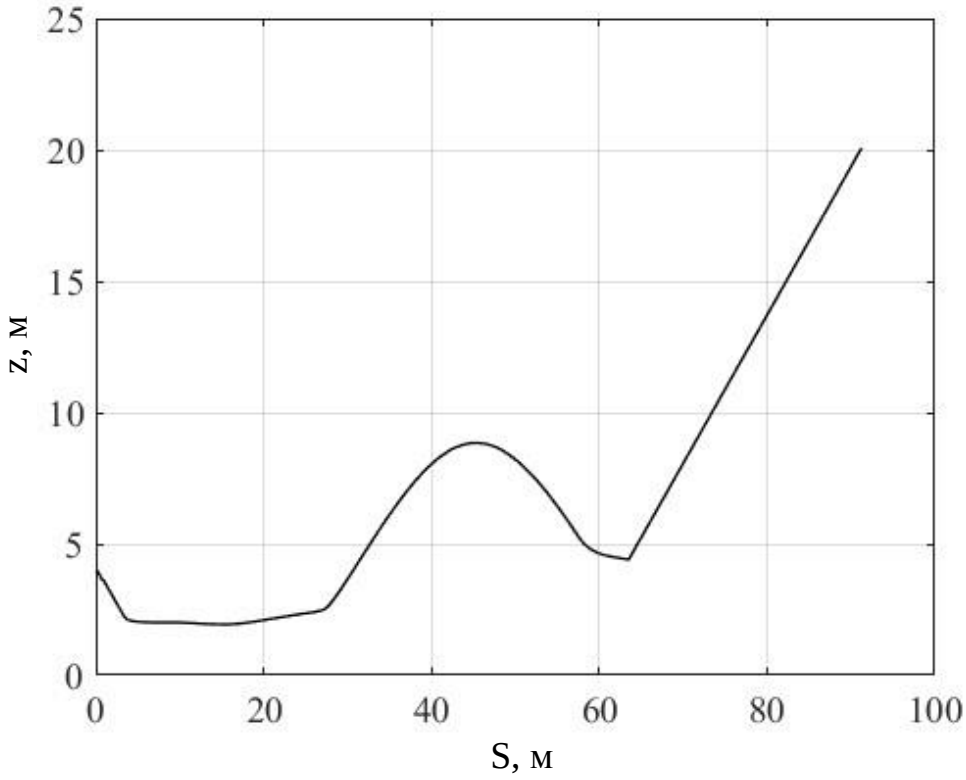

Рисунок 3.5 – Изменение координаты z робота

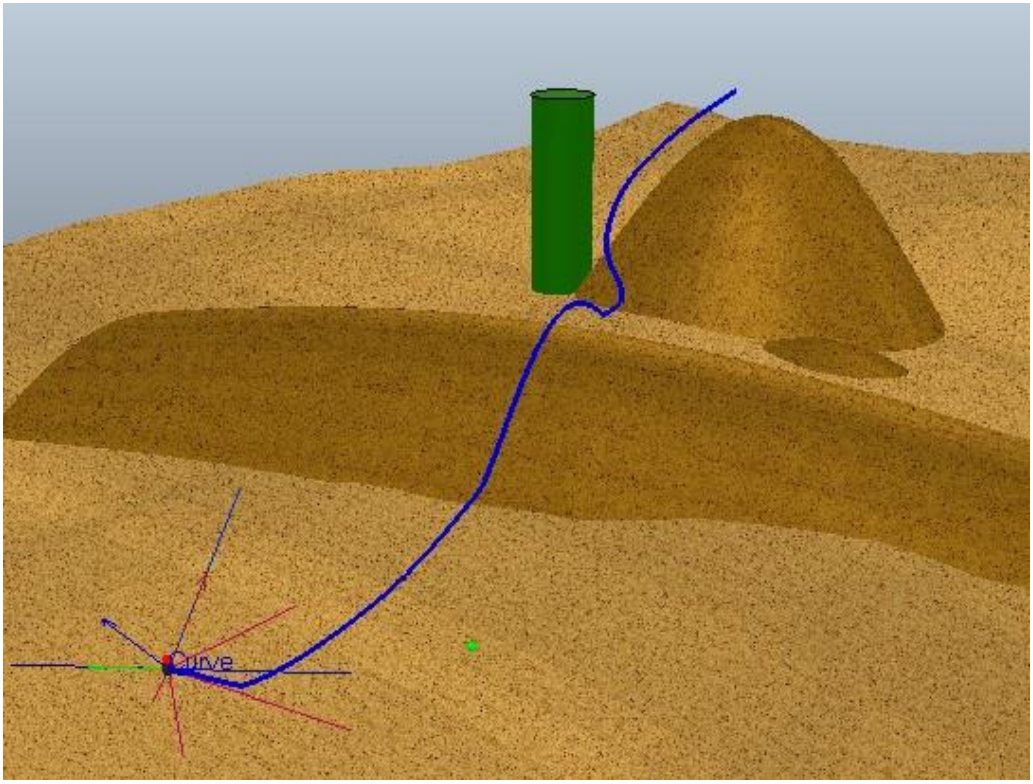

Рисунок 3.6 – Виртуальная сцена

Из рисунков 3.4-3.6 видно, что в процессе движения на пути мобильного робота встречаются препятствия разных видов, в том числе возвышенности с крутым и плавным уклоном. Высота первого препятствия позволяет роботу обойти его сверху: в этом случае результирующая и исходная траектории (см. рис. 3.4) совпадают. В случае со следующим препятствием применяется стратегия коррекции заранее спланированного пути в горизонтальной плоскости, при этом после превышения значения контрольного угла β_{max}, стратегии обхода в вертикальной и горизонтальной плоскостях совмещаются. Значение угла тангажа рассчитывается с помощью нечеткой логики. Робот начинает восхождение на препятствие.

На рисунке 3.7 показано изменение значения φ^* в процессе движения робота. Из рисунка видно, что начало движения робота сопровождается изменением значения φ^*. Данное явление связано с тем, что изначально робот находится на высоте 4 метра над поверхностью, но желаемая высота составляет 2 метра: в процессе снижения высоты в зону видимости

горизонтальных датчиков попадает поверхность движения, и расчетное значение φ^* изменяется. Когда осуществляется обход первого препятствия, $\varphi^* < \varphi^*_{\max}$ (см. рис. 3.7), т.е. первое препятствие преодолевается в вертикальной плоскости. Во время обхода второго препятствия $\varphi^* > \varphi^*_{\max}$: в данном случае выполняется коррекция исходной траектории и осуществляется обход сбоку. Приблизительно на 80 сек. виден скачок значения φ^*, который происходит в момент принятия решения о восхождении на возвышенность для достижения целевой точки. Данный скачок объясняется тем, что до начала восхождения вертикальные датчики «видят» поверхность движения, но далее в их зону видимости попадает непосредственно возвышенность, и наблюдается уменьшение значения φ^* (отталкиваемся от особенности расчета φ^*).

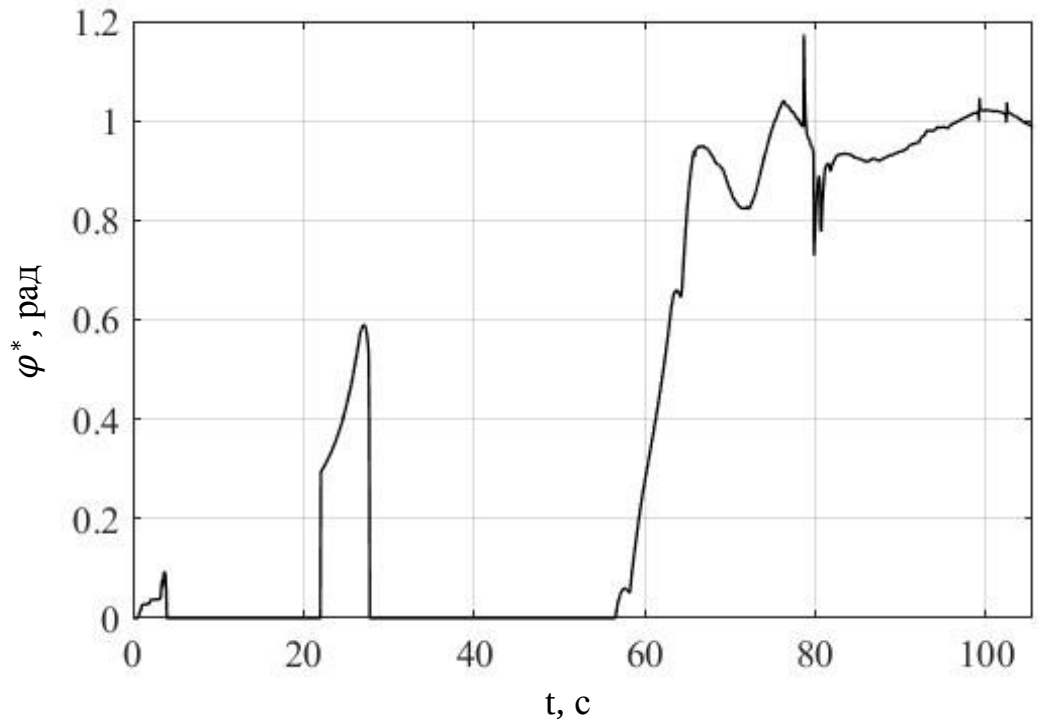

Рисунок 3.7 – Изменение значения φ^*

Во время движения значение угла тангажа мобильного робота φ_d не превышает максимально разрешенного значения $\varphi^*_{\max} = 0.6$ радиан (см. рис. 3.8), а отклонение высоты движения над поверхностью H от заданной высоты

H_d (см. рис. 3.9) до начала восхождения на вторую возвышенность не превышает 0.18 м., при этом наибольшее отклонение наблюдается в момент подъема на первую возвышенность и спуска с нее. Восхождение на вторую возвышенность сопровождается увеличением высоты H.

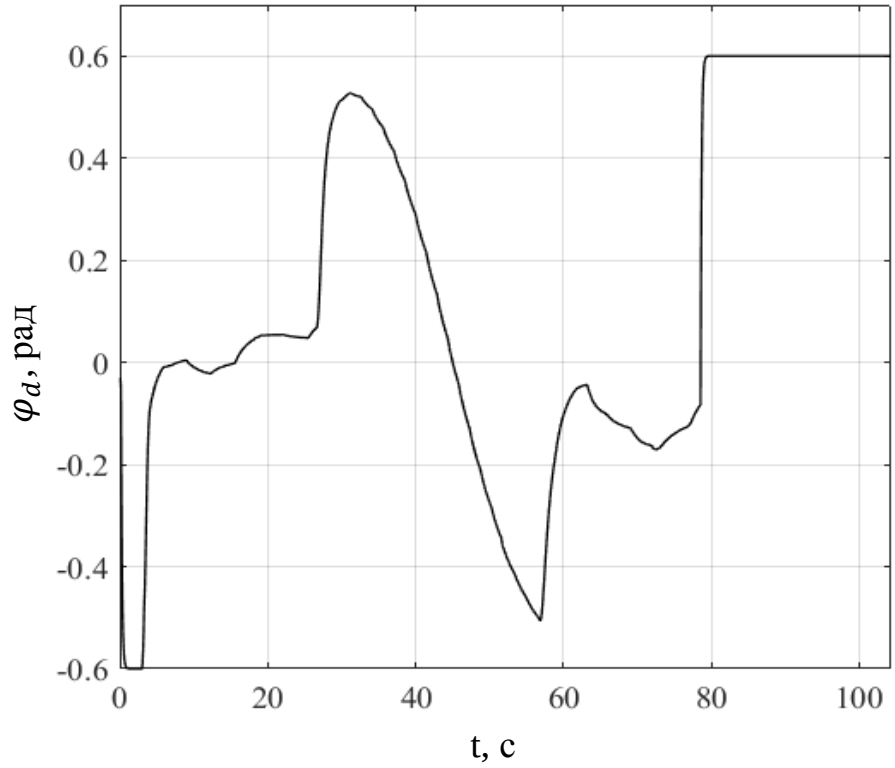

Рисунок 3.8 – Изменение значения угла тангажа φ_d

На рисунках 3.10 и 3.11 показан вид результирующей траектории робота при использовании метода, описанного в [15], без дополнения его элементами нечеткой логики.

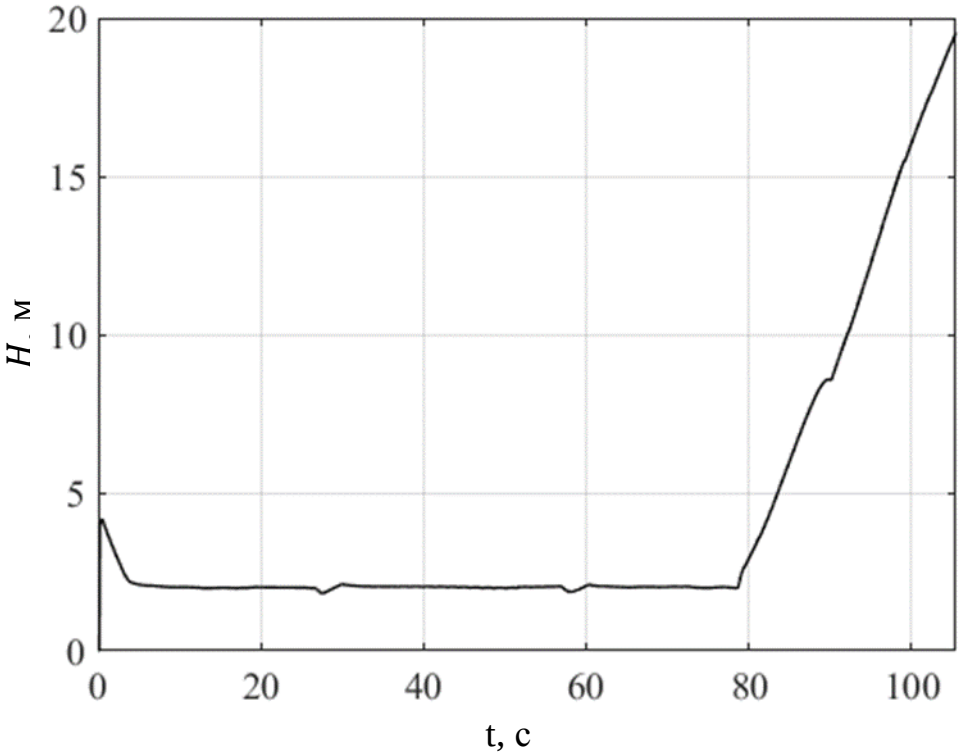

Рисунок 3.9 – Изменение значения *H*

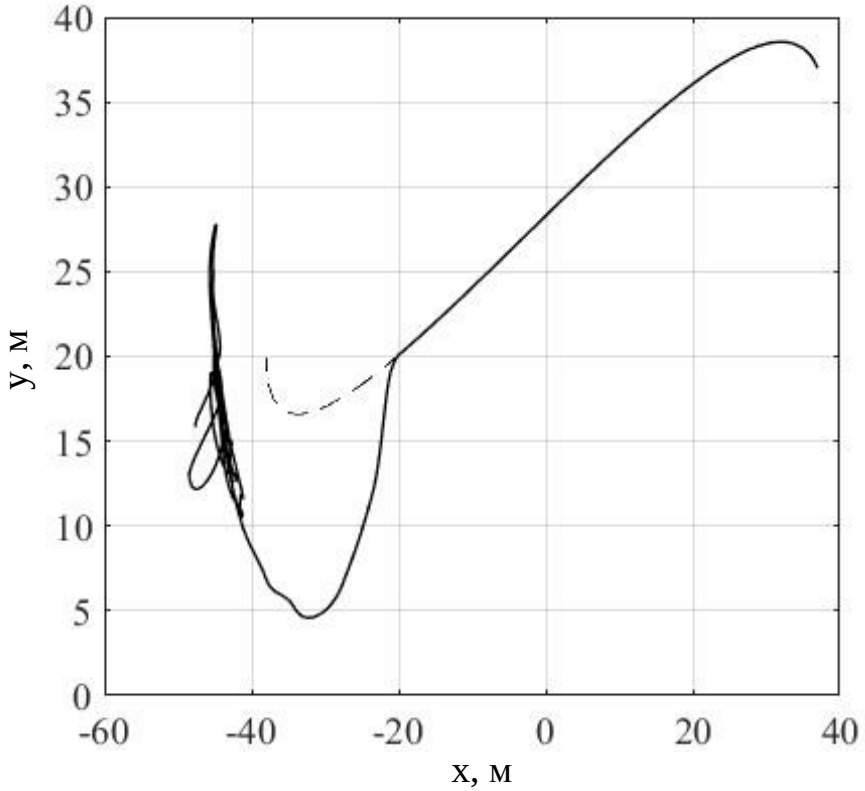

Рисунок 3.10 – Исходная и результирующая траектория

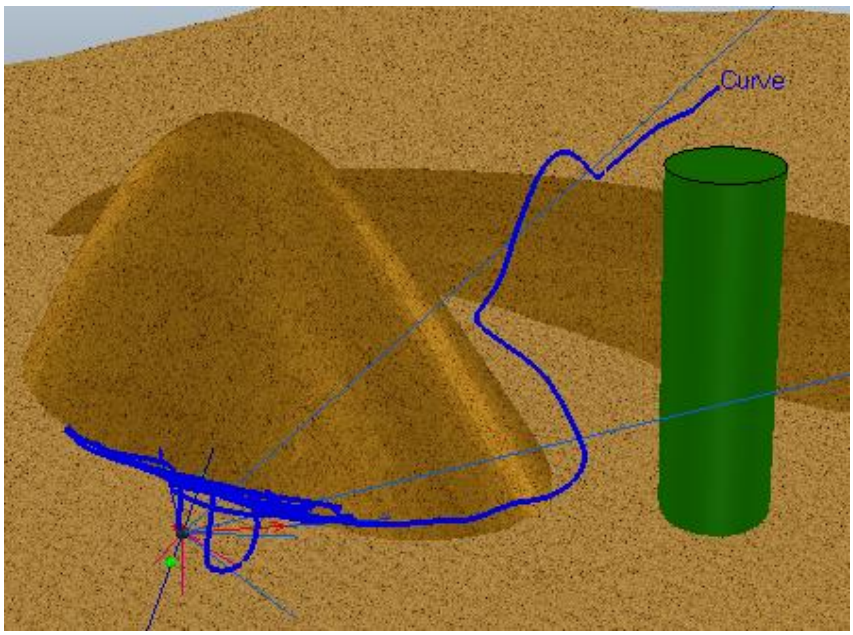

Рисунок 3.11 – Виртуальная сцена

Из рисунков 3.10 и 3.11 видно, что исходный метод позволяет роботу успешно обойти препятствия, встречающиеся на его пути. Однако, не подразумевает возможность восхождения на возвышенность с резким уклоном. Таким, что для обхода только в вертикальной плоскости пришлось бы превысить ограничение по углу тангажа φ_d.

Результаты моделирования говорят о том, что рассматриваемый алгоритм позволяет планировать пространственные траектории движения мобильных роботов в неизвестном окружении, при этом высота до поверхности движения поддерживается постоянной, а расстояние до препятствий безопасным. Алгоритм позволяет осуществлять автоматический выбор стратегии обхода препятствия, а за счет применения элементов нечеткой логики становится доступным промежуточный режим обхода, когда препятствие преодолевается в вертикальной плоскости с одновременной коррекцией заранее спланированного в горизонтальной плоскости пути, если это необходимо для достижения цели.

Заключение

В процессе выполнения данной работы был реализован и дополнен элементами нечеткой логики метод формирования пространственных траекторий движения мобильного робота в неизвестном окружении на основе данных, поступающих с бортовых датчиков приближения, предложенный в работе [15]. Данный алгоритм основан на методе формирования гладких траекторий, описанном в работе [52]. Нечеткая логика применяется в том случае, когда это необходимо для достижения целевой точки, при этом осуществляется некоторый промежуточный режим движения.

Разработанный алгоритм характеризуется механизмом выбора стратегии обхода препятствия: в вертикальной плоскости без коррекции заранее спланированного пути, сбоку с его коррекцией или, если целевая точка находится на вершине возвышенности с резким уклоном и не может быть достигнута путем обхода препятствия сверху, в вертикальной плоскости с одновременной коррекцией исходной траектории – и принципом формирования гладких траекторий, который заключается в использовании сплайнов Безье третьего порядка.

Данный метод отличается малой вычислительной стоимостью и за счет своих особенностей позволяет увеличить скорость движения мобильного робота, что является его преимуществом.

В дальнейшем метод может быть модифицирован путем применения сплайнов Безье к траектории движения робота в пространственных координатах. Работа может вестись в направлении создания режимов, позволяющих роботу обходить препятствие снизу или заезжать в туннели, а также выбирать направление обхода препятствия на основе уже пройденного пути.

Список использованных источников

1. Lim Chee Wang. Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment / Lim Chee Wang, Lim Ser Yong, M.H. Ang. // Proceedings of the IEEE International Symposium on Intelligent Control (ISIC). – Vancouver, Canada: – 2002. – С. 821-826.

2. Wu, Zh. Obstacle prediction–based dynamic path planning for a mobile robot / Zh. Wu, L. Feng. // International Journal of Advancements in Computing Technology, 2012. – Vol. 4. – № 3. – С. 118-124.

3. Jianping, T. Genetic algorithm based path planning for a mobile robot / T. Jianping, S. Yang. // Proceedings of the IEEE International Conference on Robotics and Automation. – Taipei, Taiwan: – 2003. – Vol. 1. – С. 1221-1226.

4. Shiltagh, N. Path Planning of Intelligent Mobile Robot Using Modified Genetic Algorithm / N. Shiltagh, L. Jalal. // International Journal of Soft Computing and Engineering (IJSCE), 2013. – Vol. 3. – № 2. – С. 31-36.

5. Sedighi, K.H. Autonomous Local Path-Planning for a Mobile Robot Using a Genetic Algorithm / K.H. Sedighi, K. Ashenayi, T.W. Manikas, R.L. Wainwright and other. // Proceedings of the Congress on Evolutionary Computation. – Portland, USA: – 2004. – Vol. 2. – С. 1338-1345.

6. Shiltagh, N. Optimal path planning for intelligent mobile robot navigation using modified particle swarm optimization / N. Shiltagh, L. Jalal. // International Journal of Engineering and Advanced Technology, 2013. – Vol. 2. – № 4. – С. 260-267.

7. Garrido, S. Path planning for mobile robot navigation using Voronoi diagram and fast marching / S. Garrido, L. Moreno, D. Blanco, P. Jurewicz. // International Journal of Robotics and Automation, 2011. – Vol. 2. – № 1. – С. 42-64.

8. Mohanraj, T. Mobile robot path planning using ant colony optimization / T. Mohanraj, S. Arunkumar, M. Raghunath, M. Anand. // International Journal of

Research in Engineering and Technology (IJRET), 2013. – Vol. 213. – № 11. – C. 1-6.

9. Saha, O. Real-time robot path planning around complex obstacle patterns through learning and transferring options / O. Saha, P. Dasgupta. // Proceedings of the IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). – Coimbra, Portugal: – 2017. – C. 278-283.

10. Jiang, M. Mobile robot path planning based on dynamic movement primitives / M. Jiang, Y. Chen, W. Zheng, H. Wu and other. // Proceedings of the IEEE International Conference on Information and Automation. – Ningbo, China: – 2017. – C. 980-985.

11. Ge, S. S. Dynamic motion planning for mobile robots using potential field method / S. S. Ge, Y.J. Cui. // Autonomous Robots, 2002. – Vol. 13. – C. 207-222.

12. Zhou, J.-H. A self-localization and path planning technique for mobile robot navigation / J.-H. Zhou, H.-Y. Lin. // Proceedings of the 8th World Congress on Intelligent Control and Automation (WCICA). – Taipei, Taiwan: – 2013. – C. 694-699.

13. Tan, J. The 3D Path Planning Based on A^* Algorithm and Artificial Potential Field for the Rotary-Wing Flying Robot / J. Tan, L. Zhao, Y. Wang, Y. Zhang and other. // Proceedings of the 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC). – Hangzhou, China: – 2016. – Vol. 2. – C. 551-556.

14. Агеев, М.Д. Автономные подводные роботы: системы и технологии: учебник / М.Д. Агеев. – М.: Наука, 2005 – 398 с.

15. Yukhimets, D.A. Method of spatial path planning for mobile robot in unknown environment / D.A. Yukhimets, A.V. Zuev, A.S. Gubankov. // Proceedings of the 28th DAAAM International Symposium. – Vienna, Austria: – 2017. – C. 258-267.

16. Leena, N. A Survey on Path Planning Techniques for Autonomous Mobile robots / N. Leena, K. K. Saju. // . IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE), 2014. – C. 76-79.

17. Zeyad, A. A. A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games / A. A. Zeyad, S. Shahrizal, K. Hoshang. // International Journal of Computer Games Technology, 2015. – Article ID 736138. – 11 c.

18. Huang, H.-P. Dynamic visibility graph for path planning / H.-P. Huang, S.-Y. Chung. // Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). – Sendai, Japan: – 2004. – Vol. 3. – C. 2813-2818.

19. Janet, J.A. The essential visibility graph: An approach to global motion planning for autonomous mobile robots / J.A. Janet, R.C. Luo, M.G. Kay // Proceedings of the IEEE International Conference on Robotics and Automation. – Nagoya, Japan: – 1995. – Vol. 2. – C. 1958-1963.

20. Habib, M.K. Efficient method to generate collision free paths for an autonomous mobile robot based on new free space structuring approach / M.K. Habib, H. Asama // Proceedings of the IEEE/RSJ International Workshop on intelligent robots and systems. – Osaka, Japan: – 1991. – Vol. 2. – C. 563-567.

21. Peter, E. Hart. A formal basis for the heuristic determination of minimum cost paths / Peter E. Hart, Nils J. Nilsson, Bertram Raphael. // IEEE Transactions on Systems Science and Cybernetics, 1968. – Vol. 4. – C. 100-107.

22. Lingelbach, F. Path planning using probabilistic cell decomposition / F. Lingelbach. // Proceedings of the IEEE International Conference on Robotics and Automation. – New Orleans, LA, USA: – 2004. – Vol. 1. – C. 467-472.

23. Sleumer, N.H. Exact cell decomposition of arrangements used for path planning in robotics / N.H. Sleumer, N. Tschichold-Gurman. // Technical Report, 1999.

24. Elfes, A. Using occupancy grids for mobile robot perception and navigation / A. Elfes. // Computer, 1989. – Vol. 22. – № 6. – C. 46-57.

25. Yahja, A. Framed-quadtree path planning for mobile robots operating in sparse environments / A. Yahja, A. Stentz, S. Singh, B.L. Brumitt. // Proceedings of the IEEE International Conference on Robotics and Automation. – Leuven, Belgium: – 1998. – Vol. 1. – C. 650-655.

26. Kitamura, Y. 3-D path planning in a dynamic environment using an octree and an artificial potential field / Y. Kitamura, T. Tanaka, F. Kishino, M. Yachida. // Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. – Pittsburgh, PA, USA: – 1995. – Vol. 2. – C. 474-481.

27. Kazakov, K.A. Topological Mapping Complex 3D Environments Using Occupancy Octrees / K.A. Kazakov, V.A. Semenov, V.A. Zolotov. // Proceedings of the 21st International Conference on Computer Graphics and Vision. – Moscow, Russia: – 2011. – C. 111-114.

28. Chazelle, B. Approximation and Decomposition of Shapes / B. Chazelle. // Advances in Robotics: Algorithmic and Geometric Aspects of Robotics, 1987. – Vol. 1. – C. 145-185.

29. De Berg, M. Computational Geometry: Algorithms and Applications: textbook / De Berg M., Cheong O., Van Kreveld M., Overmars M. – 3rd edition. – S.: Springer-Verlag Telos, 2008. – 386 c.

30. Canny, J.F. An opportunistic global path planner / J.F. Canny, M.C. Lin. // Algorithmica, 1993. – Vol. 10. – № 2-4. – C. 102-120.

31. Stuart, J. R. Artificial Intelligence: A Modern Approach: textbook / Stuart J. Russell, Peter Norvig – 1st edition – Englewood Cliffs, NJ: Prentice-Hall Inc. A Simon and Schuster Company, 1995. – 75 c.

32. Stentz, A. Optimal and Efficient Path Planning for Partially-Known Environments / A. Stentz. // Proceedings of the IEEE International Conference on Robotics and Automation. – San Diego, California: – 1994. – Vol. 4. – C. 3310-3317.

33. Nash, A. Theta*: Any-Angle Path Planning on Grids / A. Nash, K. Daniel, S. Koenig. // Journal of Artificial Intelligence Research, 2010. – Vol. 39. – C. 533-579.

34. Koenig, S. Lifelong planning A* / S. Koenig, M. Likhachev, D. Furcy. // Artificial Intelligence, 2004. – Vol. 155. – № 1-2. – C. 93-146.

35. Koenig, S. D* lite / S. Koenig, M. Likhachev. // Proceedings of the 18th National Conference on Artificial Intelligence. – Edmonton, Alberta, Canada: – 2002. – С. 476-483.

36. Ge, S.S. New potential functions for mobile robot path planning / S.S. Ge, Y.J. Cui. // IEEE Transactions on Robotics and Automation, 2000. – Vol. 16. – № 5. – С. 615-620.

37. Jing, R. Modified Newton's method applied to potential fieldbased navigation for mobile robots / R. Jing, K.A. McIsaac, R.V. Patel. // IEEE Transactions on Robotics, 2006. – Vol. 22. – № 2. – С. 384-391.

38. Ferrara, A. Second-order sliding-mode control of a mobile robot based on a harmonic potential field / A. Ferrara, M. Rubagotti. // IET Control Theory and Applications, 2008. – Vol. 2. – № 9. – С. 807-818.

39. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots / O. Khatib. // International Journal of Robotics Research, 1986. – Vol. 5. – № 1. – С. 90-98.

40. Wallgrun, J. O. Voronoi graph matching for robot localization and mapping / J. O. Wallgrun. // Transactions on computational science IX. – B.: Springer, 2010 – С. 76-108.

41. Fan, X.-P. Dynamic obstacle-avoiding path plan for robots based on a new artificial potential field function / X.-p. Fan, S.-y. Li, T.-f. Chen. // Control Theory and Applications, 2005. – Vol. 22. – № 5. – С. 703-707.

42. Borenstein, J. The vector field histogram-fast obstacle avoidance for mobile robots / J. Borenstein, Y. Koren. // IEEE Transactions on Robotics and Automation, 1991. – Vol. 7. – № 3. – С. 278-288.

43. Ulrich, I. VFH+: Reliable obstacle avoidance for fast mobile robots / I. Ulrich, J. Borenstein. // Proceedings of the IEEE International Conference on Robotics and Automation. – Leuven, Belgium: – 1998. – Vol. 2. – С. 1572-1577.

44. Ulrich, I. VFH*: Local obstacle avoidance with look-ahead verification / I. Ulrich, J. Borenstein. // Proceedings of the IEEE International Conference on

Robotics and Automation. – San Francisco, CA, USA: – 2000. – Vol. 3. – C. 2505-2511.

45. Zhu, Y. A New Bug-type Navigation Algorithm Considering Practical Implementation Issues for Mobile Robots / Yi Zhu, Tao Zhang, Jingyan Song, Xiaqin Li. // Proceedings of the IEEE International Conference on Robotics and Biomimetics. – Tianjin, China: – 2010. – C. 531-536.

46. Emam Fathy Mohamed. An improved Tangent Bug method integrated with artificial potential field for multi-robot path planning / Emam Fathy Mohamed, Khaled El-Metwally, A. R. Hanafy. // Proceedings of the International Symposium on Innovations in Intelligent Systems and Applications. – Istanbul, Turkey: – 2011. – C. 555-559.

47. Sezer, V. A Novel Obstacle Avoidance Algorithm: Follow the Gap Method / V. Sezer, M. Gokasan. // Robotics and Autonomous Systems, 2012. –Vol. 60. – № 9. – C. 1123-1134.

48. Zohaib, M. An Improved Algorithm for Collision Avoidance in Environments Having U and H Shaped Obstacles / M. Zohaib, M. Pasha, N. Javaid, A. Salaam and other. // Studies in Informatics and Control Journal, 2014. –Vol. 23. – № 1. – C. 97-106.

49. Zohaib, M. Intelligent Bug Algorithm (IBA): A Novel Strategy to Navigate Mobile Robots Autonomously / M. Zohaib, M. Pasha, J. Nadeem, I. Jamshed. // Proceedings of the Third International Multi-topic Conference. – Jamshoro, Pakistan: – 2012. – C. 291-299.

50. Motlagh, R. E. Development of a new minimum avoidance system for a behavior-based mobile robot / R. E. Motlagh, T. S. Hong, N. Ismail. // Fuzzy Sets and Systems, 2009. –Vol. 160. – № 13. – C. 1929-1946.

51. Бекасов, Д.Е. Применение аппарата нечеткой логики при решении задачи поиска пути в неизвестном окружении / Д.Е. Бекасов. // Молодежный научно-технический вестник МГТУ им. Н.Э. Баумана, 2012. – № 5. – С. 40.

52. Filaretov, V.F. Planning smooth paths for mobile robots in an unknown environment / V.F. Filaretov, D.A. Yukhimets. // International Journal of Computer and Systems Sciences, 2017. –Vol. 56. – № 4. – С. 738-748.

53. Filaretov, V.F. The formation of motion laws for mechatronics objects along the paths with the desired speed / V.F. Filaretov, A.S. Gubankov, I.V. Gornostaev. // Proceedings of the International Conference on Computer, Control, Informatics and Its Applications. – Jakarta, Indonesia: – 2016. – С. 93-96.

54. Filaretov, V. F. Adaptive system forming extremely high speed of multilink manipulator gripper / V. F. Filaretov, A. S. Gubankov. // Proceedings of the 23rd International DAAAM Symposium. – Vienna, Austria: – 2016. – С. 473-476.

Содержание

Приложение А. Программная реализация системы планирования траекторий

Аннотация

В данном приложении приведен текст программы «SmoothPath», предназначенной для планирования гладких пространственных траекторий движения мобильного робота на основе нечеткой логики. Текст программы реализован в виде символической записи в формате m-code во встроенной среде разработки Matlab.

Основной задачей программы является интеграция среды Matlab с робототехническим симулятором V-REP для проведения математического моделирования с целью исследования описанного метода. Для этого используется интерфейс программирования удаленного приложения Remote API для Matlab.

Программа «SmoothPath» реализует следующие функции:

а) Сбор и обработка данных с бортовых датчиков виртуального робота;

б) Пересчет показаний в абсолютную систему координат;

в) Расчет траектории робота;

г) Выбор стратегии движения;

д) Расчет дополнительных опорных точек траектории в соответствии с выбранной стратегией;

е) Расчет координат положения и углов ориентации мобильного робота.

Исходный код

```
% function simpleTest()
    disp('Program started');
    vrep=remApi('remoteApi');
    vrep.simxFinish(-1);% закрыть все существующие соединения (на всякий
случай)
    clientID=vrep.simxStart('127.0.0.1',19997,true,true,5000,5);
```

```
    if (clientID>-1)
        disp('Connected to remote API server');

        % enable the synchronous mode on the client:
        vrep.simxSynchronous(clientID,true);

        % получаем хэндлы устройств

        ps  = zeros(4,1); % формируем вектор-строку (4x1), заполненную
нулями, которую далее заполним хэндлами вертикальных датчиков (4 штуки)
        psd = zeros(7,1); % вектор-строка (7x1), предназначенная для
последующего заполнения хэндлами горизонтальных датчиков (7 штук)

        % заполняем вектор-строку ps
        % вертикальные датчики располагаются на Cuboid
        [res, ps(1)] = vrep.simxGetObjectHandle(clientID,
'Proximity_sensor3', vrep.simx_opmode_oneshot_wait);% 1-ый элемент
соответствует 3-му датчику, расположенному слева-вниз
        [res, ps(2)] = vrep.simxGetObjectHandle(clientID,
'Proximity_sensor2', vrep.simx_opmode_oneshot_wait);% 2-ый элемент
соответствует 2-му датчику, расположенному спереди-вниз
        [res, ps(3)] = vrep.simxGetObjectHandle(clientID,
'Proximity_sensor1', vrep.simx_opmode_oneshot_wait);% 3-ый элемент
соответствует 1-му датчику, расположенному справа-вниз
        [res, ps(4)] = vrep.simxGetObjectHandle(clientID,
'Proximity_sensor4', vrep.simx_opmode_oneshot_wait);% 4-ый элемент
соответствует 4-му датчику, расположенному сзади-вниз

        % Cuboid выступает в роли объекта управления (robot)
        [res, robot] = vrep.simxGetObjectHandle(clientID, 'Cuboid'
, vrep.simx_opmode_oneshot_wait);

        % заполняем вектор-строку psd
        % горизонтальные датчики располагаются на Sphere
        % угол между соседними датчиками 30 градусов
        [res, psd(4)] = vrep.simxGetObjectHandle(clientID, 'PSD1'
, vrep.simx_opmode_oneshot_wait);% 4-й элемент соответствует 1-му датчику
        [res, psd(1)] = vrep.simxGetObjectHandle(clientID, 'PSD2'
, vrep.simx_opmode_oneshot_wait);% 1-й элемент соответствует 2-му датчику
        [res, psd(2)] = vrep.simxGetObjectHandle(clientID, 'PSD3'
, vrep.simx_opmode_oneshot_wait);% 2-й элемент соответствует 3-му датчику
        [res, psd(3)] = vrep.simxGetObjectHandle(clientID, 'PSD4'
, vrep.simx_opmode_oneshot_wait);% 3-й элемент соответствует 4-му датчику
        [res, psd(5)] = vrep.simxGetObjectHandle(clientID, 'PSD5'
, vrep.simx_opmode_oneshot_wait);% 5-й элемент соответствует 5-му датчику
        [res, psd(6)] = vrep.simxGetObjectHandle(clientID, 'PSD6'
, vrep.simx_opmode_oneshot_wait);% 6-й элемент соответствует 6-му датчику
        [res, psd(7)] = vrep.simxGetObjectHandle(clientID, 'PSD7'
, vrep.simx_opmode_oneshot_wait);% 7-й элемент соответствует 7-му датчику


        [res, point_as] = vrep.simxGetObjectHandle(clientID, 'Sphere'
, vrep.simx_opmode_oneshot_wait);

        % недетектируемые сферы для проверки
        [res, pts1_vis] = vrep.simxGetObjectHandle(clientID, 'P1'
, vrep.simx_opmode_oneshot_wait); % красная сфера
        [res, pts2_vis] = vrep.simxGetObjectHandle(clientID, 'P2'
, vrep.simx_opmode_oneshot_wait); % зеленая сфера
        [res, pts3_vis] = vrep.simxGetObjectHandle(clientID, 'P3'
, vrep.simx_opmode_oneshot_wait); % синяя сфера
```

```
        [res, pts4_vis] = vrep.simxGetObjectHandle(clientID, 'P4'
, vrep.simx_opmode_oneshot_wait); % сиреневая сфера
        [res, pts5_vis] = vrep.simxGetObjectHandle(clientID, 'P5'
, vrep.simx_opmode_oneshot_wait); % зеленая сфера (конечная точка)

        [res, time_vrep] = vrep.simxGetFloatSignal(clientID,
'TIME',vrep.simx_opmode_streaming);

        % получаем позицию (3 координаты) и ориентацию робота (3 угла) в
абсолютной системе координат
        [res, x] = vrep.simxGetObjectPosition(clientID, robot, -1,
vrep.simx_opmode_streaming);
        [res, r] = vrep.simxGetObjectOrientation(clientID, robot, -1,
vrep.simx_opmode_streaming);

        % получаем данные с вертикальных датчиков
        % создаем матрицы необходимых размерностей, заполненные нулями
        dt_v   = zeros(4,1); % вектор-строка, подлежащая заполнению статусами
обнаружения препятствия вертикальными датчиками (0 или 1)
        pts_v  = zeros(4,3); % матрица (4x3), в которую будут помещены
координаты (X, Y, Z)обнаруженных датчиками точек на поверхности, над которой
осуществляется движение
        obj_v  = zeros(4,1); % вектор-строка, в которую будут помещены хэндлы
обнаруженных вертикальными датчиками объектов
        norm_v = zeros(4,3); % матрица (4x3), в которую будут помещены
координаты векторов нормали до детектируемых поверхностей

        % заполняем нулевые матрицы dt_v, pts_v, obj_v, norm_v
соответствующими данными
        % данные будут получены относительно систем координат, закрепленных в
датчиках
        for i=1:4
            [res, dt_v(i), pts_v(i,:), obj_v(i), norm_v(i,:)] =
vrep.simxReadProximitySensor(clientID, ps(i), vrep.simx_opmode_streaming);
        end

        % получаем данные с горизонтальных датчиков
        % создаем матрицы необходимых размерностей, заполненные нулями
        d      = zeros(7,1);
        dt_h   = zeros(7,1); % вектор-строка, подлежащая заполнению статусами
обнаружения препятствия горизонтальными датчиками (0 или 1)
        pts_h  = zeros(7,3); % матрица (4x3), в которую будут помещены
координаты (X, Y, Z)обнаруженных датчиками точек на препятствиях
        obj_h  = zeros(7,1); % вектор-строка, в которую будут помещены хэндлы
обнаруженных горизонтальными датчиками объектов
        norm_h = zeros(7,3); % матрица (4x3), в которую будут помещены
координаты векторов нормали до детектируемых поверхностей

        % заполняем нулевые матрицы dt_h, pts_h, obj_h, norm_h
соответствующими данными
        % данные будут поличены относительно систем координат, закрепленных в
датчиках
        for i=1:7
            [res, dt_h(i), pts_h(i, :), obj_h(i), norm_h(i, :)] =
vrep.simxReadProximitySensor(clientID, psd(i), vrep.simx_opmode_streaming);
        end

        hh_r = zeros(7,3); % нулевая матрица (7x3), в которую будут помещены
углы ориентации каждого из горизонтальных датчиков в абсолютной системе
координат
        % hh_rp = zeros(7,3);
```

```
        xh    = zeros(7,3); % нулевая матрица (7x3), в которую будут помещены
координаты ( уже пересчитанные в а.с.к.) точек, обнаруженных горизонтальными
датчиками

    %--- получаем данные для определения координат точек ------------------
---
    % заполняем матрицу hh_r
    for i = 1:7
        [res, hh_r(i,:)] = vrep.simxGetObjectOrientation(clientID, psd(i), -
1, vrep.simx_opmode_streaming);% ориентация горизонтальных датчиков
    end

    % заполняем вектор-строки (3x1) углами ориентации вертикальных датчиков
    [res, fd_r] = vrep.simxGetObjectOrientation(clientID, ps(2),
vrep.sim_handle_parent, vrep.simx_opmode_streaming); % ориентация 2-го
датчика, который расположен спереди-вниз
    [res, bd_r] = vrep.simxGetObjectOrientation(clientID, ps(4),
vrep.sim_handle_parent, vrep.simx_opmode_streaming); % 4-й датчик (сзади-
вниз)
    [res, ld_r] = vrep.simxGetObjectOrientation(clientID, ps(1),
vrep.sim_handle_parent, vrep.simx_opmode_streaming); % 3-й датчик (слева-
вниз)
    [res, rd_r] = vrep.simxGetObjectOrientation(clientID, ps(3),
vrep.sim_handle_parent, vrep.simx_opmode_streaming); % 1-й датчик (справа-
вниз)

    %------ исходные параметры для моделирования ----------------------------

        % режимы движения относительно поверхности, над которой это движение
осуществляется
        FLIGHT = 1; % код режима "лететь"
        CRAWL  = 2; % код режима "ползти"

        % типы режимов движения
        MOVING_FORWARD = 0; % поведение "двигаться прямо"
        ROUND_LEFT     = 1; % поведение
        ROUND_RIGHT    = 2; % поведение
        SHIFTING       = 3; % поведение

        mode_mov = FLIGHT;
        prev_mode = mode_mov;
        mode = MOVING_FORWARD;
        corr_tr = 0;
        Dmin = 2.0; % минимально разрешенное расстояние до препятствий
        Hd = 2.0; % расстояние от точки Xh1 до точки Xz (после расчета Xz
рассчитываем желаемый угол тангажа psi_d)
        ARRIVAL_ANGLE = pi/2; % угол ориентации робота в конечной точке
траектории
        psi_max = 0.6; % максимально разрешенное значение угла тангажа psi_d
        psi_ind = 0.6;
        v_d = 1.0; % желаемая скорость движения робота
        step_t = 0.07; % время dt в функции calc_position_xd для расчета
эталонного пути S точки X_d
        tr = 0;

        Start = 1;

        rx = [];
        st = [];
        dm = [];
        delta = 0;
```

```
        x = [0 0 0];
        xs = 0;
        xdp = 0;
        xsr = [];
        sim_time = [];
        xsp = 0;
        zp = 0;
        b = 0;
        idx_pt = 4;

        xfd = [0 0 0]; % вектор-строка, в которую будут помещены координаты
точки, обнаруженной 2-м вертикальным датчиком в а.с.к. (расположен forward-
down)
        xbd = [0 0 0]; % вектор-строка, в которую будут помещены координаты
точки, обнаруженной 4-м вертикальным датчиком в а.с.к. (расположен backward-
down)
        xff = [0 0 0]; % вектор-строка, в которую будут помещены координаты,
обнаруженной 1-м горизонтальным датчиком точки в а.с.к.
        xff_p = xff;

        xbdr = [];
        xfdr = [];
        mm = [];
        dist = [];
        dist_r = [];
        z = 4.0;

        Mb = [-1  3 -3 1;
               3 -6  3 0;
              -3  3  0 0;
               1  0  0 0];

        flag = 0;
        curr_vect_length = 0;
        ang = 0;
        check_ang = 0.8; % 0.8 рад = 45 градусов
        counter = 1;

    % ------- Путевые точки ----------------------------------------
    % путевые точки в горизонтальной плоскости
    % Whx = [x, y]
    Wp1 = [37, 37]; % координаты начальной точки траектории
    Wp4 = [-38,20]; % координаты конечной точки траектории
    target_point = [Wp4, z];
    Wp2 = Wp1 + [-10, 10];
    Wp3 = Wp4 - [cos(ARRIVAL_ANGLE), sin(ARRIVAL_ANGLE)]*norm(Wp4 - Wp1)/5; %
задается таким образом, чтобы прийти в конечную точку с нужной ориентацией

    Wh1 = Wp1;
    Wh2 = Wp2;
    Wh3 = Wp3;
    Wh4 = Wp4;

        % устанавливаем ОУ (robot, он же Cuboid) в начальную точку Wh1
(высота над поверхностью 4 метра)
        res = vrep.simxSetObjectPosition(clientID, robot, -1, [Wh1 4],
vrep.simx_opmode_oneshot);

        % заполняем вектор-строки x и r координатами положения и углами
ориентации ОУ в а.с.к. соответственно
```

```
        [res, x] = vrep.simxGetObjectPosition(clientID, robot, -1,
vrep.simx_opmode_buffer);
        [res, r] = vrep.simxGetObjectOrientation(clientID, robot, -1,
vrep.simx_opmode_buffer);
        pause(1.5);
        res = vrep.simxStartSimulation(clientID,vrep.simx_opmode_oneshot); %
начинаем выполнение моделирования в V-REP

        [res, time_vrep] = vrep.simxGetFloatSignal(clientID,
'TIME',vrep.simx_opmode_buffer);

 % ******************* Основной цикл расчета
**************************************************************************

        % пока норма вектора (x(1:2) - Wp4) > 1.0, где x(1:2) - это
координаты X и Y положения робота в а.с.к., Wp4 - это координаты конечной
точки траектории
        while norm(x(1:2) - Wp4) > 1.0

            % собираем данные с вертикальных датчиков приближения
            for i=1:4
             [res, dt_v(i), pts_v(i, :), obj_v(i), norm_v(i, :)] =
vrep.simxReadProximitySensor(clientID, ps(i), vrep.simx_opmode_buffer);
             % vrep.simxSynchronousTrigger(clientID);
            end

            dtv_r = [dtv_r; dt_v'];

            % собираем данные с горизонтальных датчиков приближения
            for i=1:7
             [res, dt_h(i), pts_h(i, :), obj_h(i), norm_h(i, :)] =
vrep.simxReadProximitySensor(clientID, psd(i), vrep.simx_opmode_streaming);
             %vrep.simxSynchronousTrigger(clientID);
            end

            % получаем координаты положения и углы ориентации ОУ в а.с.к.
соответственно
            [res, x] = vrep.simxGetObjectPosition(clientID, robot, -1,
vrep.simx_opmode_buffer);
            rx = [rx; x];
            [res, r] = vrep.simxGetObjectOrientation(clientID, robot, -1,
vrep.simx_opmode_buffer);

            % ------ расчитываем координаты точек на обнаруженных препятствиях

            % запрашиваем ориентацию горизонтальных датчиков в а.с.к.
            for i = 1:7
                [res, hh_r(i,:)] = vrep.simxGetObjectOrientation(clientID,
psd(i), -1, vrep.simx_opmode_buffer);
                %vrep.simxSynchronousTrigger(clientID);
            end

            % запрашиваем ориентацию вертикальных датчиков в с.к.,закрепленной
в роботе (Cuboid)
            [res, fd_r] = vrep.simxGetObjectOrientation(clientID, ps(2),
vrep.sim_handle_parent, vrep.simx_opmode_buffer); % 2-й датчик (спереди-вниз)
            [res, bd_r] = vrep.simxGetObjectOrientation(clientID, ps(4),
vrep.sim_handle_parent, vrep.simx_opmode_buffer); % 4-й датчик (сзади-вниз)
            [res, ld_r] = vrep.simxGetObjectOrientation(clientID, ps(1),
vrep.sim_handle_parent, vrep.simx_opmode_buffer); % 3-й датчик (слева-вниз)
```

```
            [res, rd_r] = vrep.simxGetObjectOrientation(clientID, ps(3),
vrep.sim_handle_parent, vrep.simx_opmode_buffer); % 1 -й датчик (справа-вниз)

             Rr = rotator(r(1), r(2), r(3)); % доворачиваем с.к., закрепленную
в роботе (Cuboid)до совпадения с а.с.к.

%------- пересчитываем координаты обнаруженных на поверхности точек в а.с.к.
-----------------------------------------------------
            if dt_v(2) % если второй вертикальный датчик обнаружил препятствие
                Rfd = rotator(fd_r(1), fd_r(2), fd_r(3)); % доворачиваем с.к.,
закрепленную во 2-м датчике до совпадения с с.к., закрепленной в роботе
(Cuboid)
                % пересчитываем координаты обнаруженной датчиком точки в
а.с.к., затем прибавляем координаты положения робота в а.с.к.
                xfd = x + (Rr*Rfd*(pts_v(2,:)'))'; % непосредственно пересчет
координат (x_fd, так как второй вертикальный датчик расположен forward-down
(спереди-вниз))
            end

             if dt_v(4) % если 4-й вертикальный датчик обнаружил препятствие
                Rbd = rotator(bd_r(1), bd_r(2), bd_r(3));
                xbd = x + (Rr*Rbd*(pts_v(4,:)'))';
             end

             if dt_v(1) % если 3-й вертикальный датчик обнаружил препятствие
                Rld = rotator(ld_r(1), ld_r(2), ld_r(3));
                xld = x + (Rr*Rld*(pts_v(1,:)'))';
             end

             if dt_v(3) % если 1-й вертикальный датчик обнаружил препятствие
                Rrd = rotator(rd_r(1), rd_r(2), rd_r(3));
                xrd = x + (Rr*Rrd*(pts_v(3,:)'))';
             end

             xbdr = [xbdr; xbd];% в коде не используются
             xfdr = [xfdr; xfd];% в коде не используются

%------- пересчитываем координаты обнаруженных на препятствиях точек в а.с.к.
-----------------------------------------------------
             xh = zeros(7,3);
             xhp = zeros(7,3); % в коде не используется

             for i = 1:7
                 if dt_h(i)
                    Rh = rotator(hh_r(i, 1), hh_r(i, 2), hh_r(i,3));
                    xh(i,:) = x + (Rh*(pts_h(i,:)'))';
                 end
                 %vrep.simxSynchronousTrigger(clientID);
             end

             xff = xh(4,:); % вектор-строка из координат, обнаруженной 1-м
горизонтальным датчиком точки в а.с.к. далее в коде не используется

%------- расчитываем новое положение робота---------------------------------
-----------------------------------------------------

            xd = [delta^3 delta^2 delta 1]*Mb*[Wh1; Wh2; Wh3; Wh4]; %
уравнение сплайна Безье в горизонтальной плоскости
            vd = [3*delta^2 2*delta 1 0]*Mb*[Wh1; Wh2; Wh3; Wh4]; % первая
производная уравнения сплайна Безье в горизонтальной плоскости
```

```
            % calculation dz

            % направление профиля рельефа поверхности
            av     = direction_vector(xfd - xbd); % нормированный вектор X_h1
(2-й вертикальный датчик)- X_h2 (4-й вертикальный датчик)
            % atan2(av(2), av(1)) - арктангенс в диапазоне от -pi до +pi
            % угол курса (рысканья) ОУ
            phi    = atan2(av(2), av(1));
            xfdt   = rotator(0, 0, -phi)*av';                % rotation of
line sensors fd-bd along x-axis
            Z_dltt = rotator(0, -pi/2, 0)*xfdt*Hd;           % calculation of
ve
            Z_dlt  = (xfd' + rotator(0, 0, phi)*Z_dltt)';


            % cross(av, (x - xfd)) - это векторное произведение, где av и (x -
xfd)- трёхмерные векторы
            H = norm(cross(av, (x - xfd)))/(norm(av) + 0.001);
            Hr = [Hr;H];

%---------- определяем режим движения ---------------------------------------
-------------------------------------------------
  teta_a = zeros(7,1);

  % рассчитываем углы тангажа, необходимые для обхода препятствия в
вертикальной плоскости
  for i = 1:7
      if dt_h(i) % если препятствие обнаружено i-ым горизонтальным датчиком
         switch i
             case 1 % первый горизонтальный датчик обнаружил препятствие
                 if dt_v(2)% если 2-й вертикальный датчик обнаружил
препятствие (расположен спереди)
                     teta_a(1) = atan((xh(1,3)-xrd(3))/(norm(xh(1,1:2) -
xrd(1:2))));
                 end
             case 2
                 if  dt_v(3)% если 1-й вертикальный датчик обнаружил
препятствие (расположен справа)
                     teta_a(2) = atan((xh(2,3)-xrd(3))/(norm(xh(2,1:2) -
xrd(1:2))));
                 end
             case 3
                 if dt_v(3)
                     teta_a(3) = atan((xh(3,3)-xfd(3))/(norm(xh(3,1:2) -
xfd(1:2))));
                 end
             case 4
                 if dt_v(3)
                     teta_a(4) = atan((xh(4,3)-xfd(3))/(norm(xh(4,1:2) -
xfd(1:2))));
                 end
             case 5
                 if dt_v(1) % если 3-й вертикальный датчик обнаружил
препятствие (расположен слева)
                     teta_a(5) = atan((xh(5,3)-xfd(3))/(norm(xh(5,1:2) -
xfd(1:2))));
                 end
             case 6
                 if dt_v(1)
                     teta_a(6) = atan((xh(6,3)-xld(3))/(norm(xh(6,1:2) -
xld(1:2))));
```

```
                end
            case 7
                if dt_v(1)
                    teta_a(7) = atan((xh(7,3)-xld(3))/(norm(xh(7,1:2) -
xld(1:2))));
                end
        end
    end
  end
  teta = max(teta_a); % ищем максимальный из всех рассчитанных углов тангажа
  teta_r = [teta_r; teta];

  if teta > psi_ind || flag % и если этот угол больше максимально возможного,
то сверху обойти препятствие не можем и будем выполнять обход в
горизонтальной плоскости
     mode_mov = CRAWL; % устанавливаем режим движения "ползти"
     vrep.simxAddStatusbarMessage(clientID,'CRAWL because teta >
teta_max',vrep.simx_opmode_oneshot);
     if prev_mode ~= CRAWL
         prev_mode = CRAWL;
         vrep.simxAddStatusbarMessage(clientID,'CRAWL because teta >
teta_max',vrep.simx_opmode_oneshot);
     end

  else
     mode_mov = FLIGHT; % устанавливаем режим движения "лететь"
     vrep.simxAddStatusbarMessage(clientID,'FLIGHT
1',vrep.simx_opmode_oneshot);
  end

  % ------------------------------------------------------------------------
-----------
  if (mode_mov == CRAWL) && (flag == 0)
        init_vect = [x(1)-target_point(1), x(2)-target_point(2), x(3)-
target_point(3)] % расчет координат вектора из начального положения в
желаемую точку в момент включения режима CRAWL
        flag = 1;
        vrep.simxAddStatusbarMessage(clientID,'Calculate the
init_vector',vrep.simx_opmode_oneshot);
  end
  if mode_mov == FLIGHT
      flag=0;
  end
  % ------------------------------------------------------------------------
-----------

  dist_c = abs(pts_h(:,3).*dt_h);
  dist = [];
  for i = 1:7
     if dist_c(i)
         dist = [dist; dist_c(i)];
     end
    % vrep.simxSynchronousTrigger(clientID);
  end
  if size(dist,1) == 0
         dist = [dist; 0];
  end
  dist_r = [dist_r; min(dist)];


% ****************** Алгоритм обхода препятствий **********************
% --- ищем ближайшие точки траектории----------------------------------
```

```
    xc = zeros(7,2);
    d  = zeros(7, 1);

    for i = 1:7
        if dt_h(i)
            [xc(i,:) d(i)] = closest_point(xh(i,1:2), Wh1, Wh2, Wh3, Wh4,
delta);
        end
    end

%-------------------------------------------------------------------------
O =zeros(4,7); % массив препятствий
current_obcstacle =0 ;%номер текущего препятствия которое формируем
total_obstacles =0;
obstacle_formation_flag =0;% флаг, по которому видим что идёт процесс
формирования препятствия
%формируем массив препятствий
for i = 1:7
        if dt_h(i)% если i-й датчик онаружил препятствие
           if ~obstacle_formation_flag %если процесс формирования препятствия
ещё не начат
               obstacle_formation_flag=1;%выставляем флаг формирования
препятствия
               current_obcstacle=current_obcstacle+1;%увеличиваем счётчик
формируемого препятствия
           end
           O (current_obcstacle,i)=1;% в строку препятствия записываем 1 в
место соответствующее номеру датчика, обнаружившему препятствие
           if i<7 % если массив датчиков ещё не перебран полностью
               if ~dt_h(i+1)%если следующий датчик не обнаружил препятствие
                  obstacle_formation_flag=0;%снимаем флаг формирования
препятствия, чтобы на следующем шаге началось формироваться новое
               end
           end
        end
end
total_obstacles =current_obcstacle;

%конец блока формирования массива препятстви
%анализ массива препятствий на предмет выяснения близко расположенных и
%слияния их в одно
if total_obstacles>1% если препятствий более чем 1 то выполняем данный
анализ, если нет то пропускаем этот шаг
    for i = 1:total_obstacles % перебор препятствий
       if i==total_obstacles break; end % если перебрали все препятствия то
прерываем цикл, чтобы последний цикл был пустым и мы не превысили индекс
массива
       if norm(xh(left_point(O,i))-xh(right_point(O,i+1))) < 2*Dmin % если
расстояние между правой точкой текущего препятствия и левой точкой следующего
меньше 2Dmin- сливаем препятствия
          for m=left_point(O,i+1):-1:left_point(O,i)
              O(i+1,m)=1;% формируем общее препятствие, помещаем его в
второе, первое обнуляем
          end
          for m=1:7 O(i,m)=0;end
       end
    end
end
%конец блока анализа массива препятствий и слияния в одно
%близкорасположенных препятствий
```

```
%начало блока поиск близкорасположенного препятствия относительно траектории
движения робота
%соответвтующая подмена оригинальных матриц на скорректированные
if total_obstacles %если препятствия обнаружены, то выполняем этот шаг
  for i=1:4%перебираем возможные препятствия
      my_d=d;%создаём копию матрицы расстояний
      close_obstacle_flag=0;
      if sum(O(i,1:7))%проверяем массив перпятствий, если есть ненулевой
элемент в строке- значит препятствие есть
          for m=1:7 %перезаписываем в копию матрицы расстояний только
расстояния от точек данного препятствия, остальные ставим 0
              my_d(m)=my_d(m)*O(i,m);
              if my_d(m)<Dmin % если какое-либо растояние от точки менее чем
Dmin, считаем что данное препятствие нужно обходить и на другие уже не
смотрим
                                % т.к. ситуация с двумя препятствиями
                                % находящимися на расстоянии менее Dmin уже
                                % разрулена ранее, когда мы сливали близко
                                % расположенные препятствия в одно
                  close_obstacle=i;%ставим флаг что обнаружено
близкорасположенное к траектории препятствие во флаг пишем номер данного
препятствия в массиве (пригодится далее)
              end
          end
      end
      if close_obstacle
         break;%рвём текущий цикл for i=1:4, т.е. на остальные препятствия
уже не смотрим
      end
  end
  d=my_d;% перезаписываем оригинальную матрицу расстояний своими, с
удалёнными элементами, не относящимися к нашему препятствию
  for i=1:7
      dt_h(i)=dt_h(i)*O(close_obstacle,i);%перезаписываем оригинальнуб
матрицу датчиков, убирая показания не онтосящиеся к нашему препятствию
  end
end
%конец блока поиск близкорасположенного препятствия относительно траектории
движения робота

%- далее из найденного выбираем ближайшую к траектории точку на препятствии

    da = abs(d);% убираем знак (он здесь не нужен), если 0, значит датчик
ничего не запеленговал
    d_min = 1000;
    idx_cp = 0;

    for i = 1:7
        if dt_h(i)
            if d_min > da(i)
                d_min = da(i);
                idx_cp = i;
            end
        end
    end
%-------------------------------------------------------------------------
%--выбираем режим движения ---------------------------------------------------

 S = sum(sign(d)); % положение точек по отношению к траектории
 CA = sum(dt_h);   % количество точек

 if CA && mode_mov == CRAWL
```

```
    C = S/CA;
    curr_vect = [x(1)-target_point(1), x(2)-target_point(2), x(3)-
target_point(3)]; % координаты текущего вектора положения, считаем при работе
режима CRAWL
    curr_vect_length = abs(sqrt
((curr_vect(1)^2)+(curr_vect(2)^2)+(curr_vect(3)^2)))
    ang = calc_angle_between_vectors (init_vect, curr_vect);
     if (abs(C) < 1)
         if C <= 0 % то справа от траектории больше точек на препятствии
             mode = ROUND_LEFT;
         else
             disp('C > 0')
             mode = ROUND_RIGHT; % в случае если больше обнаруженных на
препятствии точек находится слева
         end
      else
         if d_min < Dmin % проверяем, если mod (C) = 1
            mode = SHIFTING; % делаем так, чтобы стало d_min > Dmin
         else
           mode = MOVING_FORWARD; % делаем вывод, что препятствие находится
на достаточном расстоянии и корректировка не выполняется
         end
      end
 else
       mode = MOVING_FORWARD;
 end

%-- find point of round ------------------------------------------------

    switch mode
        case ROUND_LEFT
            idx_pt = 7; % leftes point
            while ~dt_h(idx_pt)
                idx_pt = idx_pt -1;
            end
        case ROUND_RIGHT
            idx_pt = 1; % rightest point
            while ~dt_h(idx_pt)
                idx_pt = idx_pt +1;
            end
        case SHIFTING
            idx_pt = idx_cp; % closest point
    end

%----- расчет опорных точек новой траектории ----------------------------

    switch mode
        case ROUND_LEFT

vrep.simxAddStatusbarMessage(clientID,'LEFT',vrep.simx_opmode_oneshot);

            In = direction_vector(xh(idx_pt, 1:2) - xh(idx_pt -1, 1:2))';
            Jn = [0 -1; 1 0]*In;
            Nn = [In'; Jn']*(x(1:2) - xh(idx_pt - 1, 1:2))';
            Wt = xh(idx_pt, 1:2) - ([0 1; -1 0]* direction_vector((xh(idx_pt,
1:2) - xh(idx_pt - 1, 1:2))'))'*1.5*Dmin*sign(Nn(2));
            Wt1 = x(1:2);
            Wt2 = x(1:2) + [cos(r(3)) sin(r(3))]*norm(Wt - Wt1)/3;
            Wt3 = Wt - direction_vector(xh(idx_pt, 1:2) - xh(idx_pt -1,
1:2))*norm(Wt - Wt1)/3;
            corr_tr = 1;
            delta = 0.0;
```

```
            Wh1 = Wt1;
            Wh2 = Wt2;
            Wh3 = Wt3;
            Wh4 = Wt;

        case ROUND_RIGHT

vrep.simxAddStatusbarMessage(clientID,'RIGHT',vrep.simx_opmode_oneshot);

            In = direction_vector(xh(idx_pt, 1:2) - xh(idx_pt + 1, 1:2))';
            Jn = [0 -1; 1 0]*In;
            Nn = [In'; Jn']*(x(1:2) - xh(idx_pt + 1, 1:2))';
            Wt = xh(idx_pt, 1:2) - ([0 1; -1 0]* direction_vector((xh(idx_pt,
1:2) - xh(idx_pt + 1, 1:2))'))'*1.5*Dmin*sign(Nn(2));
            Wt1 = x(1:2);
            Wt2 = x(1:2) + [cos(r(3)) sin(r(3))]*norm(Wt - Wt1)/3;
            Wt3 = Wt - direction_vector(xh(idx_pt, 1:2) - xh(idx_pt + 1,
1:2))*norm(Wt - Wt1)/3;
            corr_tr = 1;
            delta = 0.0;
            Wh1 = Wt1;
            Wh2 = Wt2;
            Wh3 = Wt3;
            Wh4 = Wt;

        case SHIFTING

vrep.simxAddStatusbarMessage(clientID,'SHIFT',vrep.simx_opmode_oneshot);

           Wt = direction_vector(xc(idx_pt,:) - xh(idx_pt, 1:2))*1.5*Dmin +
xh(idx_pt, 1:2);
            Wt1 = x(1:2);
            Wt2 = x(1:2) + [cos(r(3)) sin(r(3))]*norm(Wt - Wt1)/3;
            dir = [0 -1; 1 0]*direction_vector(xc(idx_pt,:) - xh(idx_pt,
1:2))';
            Wt3 = Wt - dir'*sign(C)*norm(Wt - Wt1)/3;
           corr_tr = 1;
            delta = 0.0;
            Wh1 = Wt1;
            Wh2 = Wt2;
            Wh3 = Wt3;
            Wh4 = Wt;

        case MOVING_FORWARD

vrep.simxAddStatusbarMessage(clientID,'MOVING',vrep.simx_opmode_oneshot);
            if norm(x(1:2) - Wh4) < 0.5 && corr_tr
                Wf1 = x(1:2);
                Wf4 = Wp4;
                Wf2 = Wf1 + [cos(r(3)) sin(r(3))]*norm(Wf1 - Wf4)/3;
                Wf3 = Wf4 - [cos(ARRIVAL_ANGLE) sin(ARRIVAL_ANGLE)]*norm(Wf4
- Wf1)/3;

                % checking for back semisphere

               corr_tr = 0;
                delta = 0.0;
                Wh1 = Wf1;
                Wh2 = Wf2;
                Wh3 = Wf3;
                Wh4 = Wp4;
```

```
            end

        case NOTHING
            vrep.simxAddStatusbarMessage(clientID,'ONLY
FORWARD',vrep.simx_opmode_oneshot);
    end

%**************************************************************************

            if ~delta && Start
                xdp = xd;
                zp = z;
                xsp = 0;
                Start = 0;
            end

            if (delta + 0.001) <= 1.0
                % ищем значение tau (delta), которое обеспечит движение точки
Xd с заданной скоростью vd
                delta = calc_position_xd(Wh1, Wh2, Wh3, Wh4, delta, v_d,
step_t, r(2));
                xs = xs + norm(xd - xdp);

                xsr = [xsr; xs];
            end

            %----- расчитываем желаемую ориентацию робота ------------------

            a = 0; % угол крена
            bd = atan((Z_dlt(3) - x(3))/(norm(Z_dlt(1:2) - x(1:2)) + 0.001));
% угол тангажа (phi_d обеспечивает движение ОУ по траектории с заданной
высотой над поверхностью)


            % abs возвращает абсолютное значение каждого элемента массива
            if abs(bd) > psi_max % && ang < check_ang
                bd = psi_max*sign(bd);
            end

            fismat = readfis('DistAlpha');
            if mode_mov == CRAWL
                out = evalfis([double(curr_vect_length);double(ang)], fismat);
                bd = teta*out + (1-out)* psi_ind;
                if abs(bd) > psi_max % && ang < check_ang
                bd = psi_max;
                end
            end

                b = b + (bd - b)*0.4;

            %--------------------------------------------------------------

            c = atan2(vd(2), vd(1)+0.001);

            br = [br; b];

            %--------------------------------------------------------------

             z = x(3) + norm(xd - xdp)*tan(bd); % расчет координаты z точки Xd
```

```
            %-------------------------------------------------------------


            %------- устанавливаем новые положение и ориентацию робота -------

            vv = [vv; norm([xd z] - [xdp zp])/step_t];

            res = vrep.simxSetObjectPosition(clientID, robot, -1, [xd, z],
vrep.simx_opmode_oneshot);
            % res = vrep.simxSetObjectOrientation(clientID, robot, -1, [a b
c], vrep.simx_opmode_oneshot);
            res = vrep.simxSetObjectQuaternion(clientID, robot, -1, rot_q (-
b,c), vrep.simx_opmode_oneshot);

            res = vrep.simxSetObjectPosition(clientID, pts1_vis, -1, [Wh1 z],
vrep.simx_opmode_oneshot);
            res = vrep.simxSetObjectPosition(clientID, pts2_vis, -1, [Wh2 z],
vrep.simx_opmode_oneshot);
            res = vrep.simxSetObjectPosition(clientID, pts3_vis, -1, [Wh3 z],
vrep.simx_opmode_oneshot);
            res = vrep.simxSetObjectPosition(clientID, pts4_vis, -1, [Wh4 z],
vrep.simx_opmode_oneshot);
            res = vrep.simxSetObjectPosition(clientID, pts5_vis, -1, [Wp4 z],
vrep.simx_opmode_oneshot);


vrep.simxAddStatusbarMessage(clientID,'************************',vrep.simx_op
mode_oneshot);
            %-------------------------------------------------------------

            [res, time_vrep] = vrep.simxGetFloatSignal(clientID,
'TIME',vrep.simx_opmode_buffer);
            sim_time = [sim_time; time_vrep];
            xdp = xd;
            zp = z;
           xsp = xs;
           pause(step_t);
            tr = tr + step_t;
           time_real = [time_real; tr];
            st = [st; time_vrep];

            remember_xd (1,counter)= xd(1);
            remember_xd (2,counter)= xd(2);
            counter=counter+1;
 vrep.simxSynchronousTrigger(clientID);
      end
%
*****************************************************************************
*************************************************************

        res = vrep.simxStopSimulation(clientID,vrep.simx_opmode_oneshot); %
останавливаем процесс моделирования

        vrep.simxGetPingTime(clientID);

        % Закрываем соединение с V-REP
        vrep.simxFinish(clientID);

    else
```

```
        disp('Failed connecting to remote API server');
    end
    vrep.delete(); % вызываем деструктор

    for t=0.1:.001:1
        init_traject = [t^3 t^2 t 1]*Mb*[Wp1; Wp2; Wp3; Wp4];
         remember_init_traject (1,counter)= init_traject(1);
         remember_init_traject (2,counter)= init_traject(2);
         counter=counter+1;
    end

    plot (remember_init_traject(1,:),remember_init_traject(2,:),'-- k',
remember_xd (1,:),remember_xd(2,:),'- k' )

    disp('Program ended');
% окончание програм
```

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный университет»

Инженерная школа
Кафедра автоматизации и управления

ОТЗЫВ РУКОВОДИТЕЛЯ

На выпускную квалификационную работу студента(ки) Кармановой Светланы Вячеславовны
(фамилия, имя, отчество)

Направление подготовки Мехатроника и робототехника

группа Б3421

Руководитель ВКР д.т.н. (ученая степень, ученое звание) Юхимец Д.А. (ФИО)

На тему Разработка системы планирования гладких пространственных траекторий движения мобильного робота на основе нечеткой логики

Дата защиты ВКР «04» июля 2018 г.

Выпускная квалификационная работа Кармановой С.В. посвящена разработке метода планирования гладких пространственных траекторий для мобильных роботов с использованием нечеткой логики. Актуальность представленной в работе задачи обусловлена расширением области использования мобильных роботов для автономного выполнения различных осмотровых операций в неизвестном окружении. При этом выполнение таких операций с помощью подводных аппаратов и беспилотных летательных аппаратов требует их перемещения по пространственным траекториям, которые должны формироваться в процессе движения этих роботов. Основная проблема, которая возникает в процессе формирования этих траекторий это необходимость обеспечения движения робота одновременно на заданном расстоянии от поверхности и безопасной дистанции от препятствий.

Основной задачей, решаемой в работе, является модификация алгоритма формирования пространственной траектории так, чтобы робот, имеющий ограничения на величину угла дифферента, мог выйти в любую заданную точку независимо от ее расположения на рельефе.

В работе были рассмотрены различные методы формирования траекторий мобильных роботов, разработана модификация алгоритма формирования пространственной траектории мобильного робота, позволяющая ему достичь любой заданной точки,

указанная модификация алгоритма была реализована на языке программирования Matlab и апробирована в среде моделирования V-REP.

В результате, Кармановой С.В. был самостоятельно разработан алгоритм выбора режима движения мобильного робота при выходе в целевую точку, расположенную на элементах рельефа с большой крутизной склона, была разработана модификация этого алгоритма на основе нечеткой логики, выполнена реализация разработанного алгоритма формирования пространственных траекторий на языке программирования Matlab, проведено моделирование с использованием среды моделирования V-REP.
В целом ВКР Кармановой С.В. выполнена на достаточно высоком уровне. Результаты, полученные в ходе выполнения работы, развивают практические приложения в области разработки систем управления автономных робототехнических систем.
Оригинальность текста составляет 89% (по данным системы «SafeAssign», bb.dvfu.ru).
Считаю, что выпускная квалификационная работа заслуживает оценки «отлично», а студентка Карманова С.В. – присвоения квалификации «бакалавр».

Руководитель ВКР _____ Юхимец Д.А.
(подпись) (ФИО)

«21» июня 2018 г.