САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ – ПРОЦЕССОВ УПРАВЛЕНИЯ

КАФЕДРА МЕХАНИКИ УПРАВЛЯЕМОГО ДВИЖЕНИЯ

Федоров Виктор Михайлович

Управление группой подвижных объектов в среде с препятствиями

Магистерская диссертация

Направление 010400

Прикладная математика и информатика

Математическое моделирование в задачах естествознания

Научный руководитель:

кандидат физ.-мат. наук,

доцент

Алферов Г. В.

Санкт-Петербург

2019

Saint-Petersburg State University

FACULTY OF APPLIED MATHEMATICS AND CONTROL PROCESSES

Department of Mechanics of Controlled Motion

Fedorov Viktor

Control of a group of mobile objects in an environment with obstacles

Master's Thesis

01.04.02 – Applied mathematics and informatics

Mathematical Modelling in Problems of Natural Science

Scientific supervisor:

Candidate of Physico-Mathematical Sciences,

Associate Professor

Alferov G.

Saint-Petersburg

2019

Оглавление

Введение

В современном мире развитие робототехники является важным направлением развития технологий. Роботы с различной функциональностью уже обширно используются во многих сферах жизни. Одной из наиболее актуальных задач по этому направлению является построение системы навигации для группы роботов, выполняющих общую цель. Такая система может использоваться, например, для исследовательских и транспортных задач.

Важной частью данной задачи является построение навигационной системы, которая позволит роботу автономно передвигаться из пункта «А» в пункт «В», для каждого отдельного робота из системы. Решением данной задачи занимаются многие ученые и крупные корпорации. Решение данной задачи является необходимым шагом для создания полностью автономного транспорта, что делает это направление в исследовании очень востребованным.

Кроме того, решение данной задачи позволяет изучать труднодоступные для человека места. Одним из наиболее важных направлений для изучения является космос, для изучения которого просто необходима группа подвижных объектов, оснащенных навигационной системой, позволяющей им исследовать космическое пространство без участия человека, отправляя ему только итоговую информацию.

Задачу построения навигационной системы для группы подвижных объектов можно разделить на следующие подзадачи:

1) Выбор и реализация алгоритма распределения целей в группе роботов.
2) Построение мультиагентной робототехнической системы (МРТС).

3) Реализация навигационной системы, позволяющей роботу передвигаться из одной позиции в другую, избегая столкновения с препятствиями, для каждого робота. Данная задача состоит из следующих подзадач:

   a) Построение навигационной системы для движения в детерминированной среде;

   b) Построение навигационной системы для движения в недетерминированной среде со стационарными препятствиями;

   c) Построение навигационной системы для движения в недетерминированной среде с динамическими препятствиями;

В настоящее время самыми часто используемыми автономными подвижными объектами по причине удобства их использования являются мобильные роботы. В связи с этим рассматривается задача построения навигационной системы именно для данного типа роботов.

Постановка задачи

Разработать систему навигации, которая позволит мобильным роботам автономно выполнять поставленные перед ними задачи по транспортировке и исследованию территории.

Цели работы:

- Анализ известных методов построения мультиагентной робототехнической системы
- Анализ известных алгоритмов группового управления
- Разработка навигационной системы, позволяющей мобильным роботам совершать автономное движение в различных средах;
- Реализация некоторых вариантов навигационной системы на мобильных роботах для решения реальных практических задач.

Обзор литературы

На сегодняшний день существует множество методов, применяемых для управления роботом в среде с препятствиями. К ним можно отнести метод структурного синтеза, использование нечеткой логики, построение навигационной системы, основанной на алгоритме SLAM.

Метод структурного синтеза, рассмотренный в [1, 2] является синергетическим методом. Главной особенностью данного метода является задание цели управления, а управление рассматривается в виде аттракторов. В работах [3, 4] рассматривается закон управления, позволяющий строить асимптотически устойчивое движение.

Мобильные роботы, основанные на использовании нечеткой логики, применяются для решения различных задач планирования движения автономных подвижных объектов. В работе [5] предложен метод, позволяющий управлять мобильным роботом в недетерминированной среде со статическими препятствиями. В статье [6] представлены методы, основанные на использовании нечеткой логики и позволяющие подвижным объектам перемещаться в среде с динамическими препятствиями.

В статье [7] представлен метод управления роботом в трехмерной среде с точечными препятствиями. Данный метод создан на базе позиционно траекторного закона управления. Рассматривается движение объекта вдоль траекторий, состоящих из прямых линий.

В работах [8, 9] рассматривается проблема генерации стратегии в сложных задачах в условиях отсутствия полной информации о внешней среде. Проведен ряд экспериментов, доказывающих эффективность предложенного метода, заключающегося в способности системы самостоятельно генерировать шаги в среде с неполной информацией.

Один из самых эффективных методов управления подвижными объектами в среде с препятствиями предложил Герасимов В. Н. в своей диссертации [10], в которой представил разработанную им систему, основанную на использовании алгоритма SLAM для локализации мобильного робота и построения динамической карты окружающей среды. Кроме того данная система позволяет строить траекторию, используя различные известные алгоритмы построения траектории в среде с препятствиями. На данный момент известно множество алгоритмов, позволяющих строить траекторию обхода препятствий в различных средах, в следствии чего сфера применения предложенной системы очень велика.

Алгоритм SLAM, подробно разобранный в работе [11] на сегодняшний день является очень популярным и эффективным. Задача навигации мобильного робота в недетерминированной среде с ограниченной областью действия сенсорных датчиков была решена с использованием данного алгоритма в работе [12].

Наиболее популярными алгоритмами построения траектории обхода препятствий являются алгоритм Дейкстры, алгоритм A* и алгоритм D*.

Алгоритм Дейкстры является базовым алгоритмом, разработанным Эдсгером Дейкстрой в 1959 году.Несмотря на то, что на данный момент этот алгоритм в большинстве случаев является слишком медленным, в следствии чего редко применяется на практике, ему до сих пор можно найти применение в некоторых сферах. В работе [13] с помощью данного алгоритма была решена проблема использования модели сети дорог с параметрами для прокладки кратчайшего пути.

Алгоритм A* является усовершенствованной модификацией алгоритма Дейкстры, разработанной Нильсом Нильсоном, Бертрамом Рафаэлем и Питером Хартом. Его основным отличием являет использование эвристики, благодаря которой значительно повышается скорость работы алгоритма с

сохранением оптимальности траектории по длине пути. Задача построения траектории обхода препятствий с использованием данного алгоритма и последующего применения к полученной траектории метода эффективного пути рассмотрена в [14, 15].

Алгоритм D*, разработанный Свеном Кёнигом и Максимом Лихачевым в 2002 году, позволяет строить траекторию обхода препятствий в недетерминированной среде. Задача построения траектории обхода препятствий с помощью данного алгоритма была рассмотрена в [16].

В работах [17, 18] рассмотрены основные и наиболее актуальные алгоритмы и модели коллективного управления группой роботов. Данные работы несут в себе большую информативность и позволяют освоить данную тематику.

Тема применения принципов группового управления для решения транспортной задачи подробно описана в [19, 20]. Это направление является одним из ключевых в развитии робототехники, что делает материал, изложенный в этих статьях, действительно значимым.

Принципы построения мультиагентных робототехнических систем достаточно подробно изложены в [21, 22]. Так же в этих работах представлено множество алгоритмов распределения целей между подвижными объектами для каждого типа группового управления. Так же принципы группового управления подробно изложены в книге [23].

Способность насекомых, таких как пчелы или муравьи, сообща выполнять различные сложные задачи заставила многих исследователей внимательно изучать их поведение и переносить его на неодушевленных роботов. В [24] приведена аналогия между управлением группой мобильных роботов и роем насекомых.

Глава 1. Обзор мультиагентной робототехнической системы

Для описания структуры и принципов работы мультиагентной робототехнической системы необходимо ввести следующие понятия:

1) Мультиагентные системы - это системы, состоящие из множества взаимодействующих агентов, коллективно решающих общую задачу. Понятие «агент» включает любые физические или виртуальные единицы, обладающие следующими свойствами:
   a) коммуникабельность - способность обмениваться информацией с другими агентами;
   b) целенаправленность - направленность на достижение какой-либо цели;
   c) наличие ряда выполняемых функций;
   d) автономность, предполагающая наличие собственных ресурсов (энергетических, вычислительных, информационно-измерительных и т. п.);
   e) реактивность - способность к восприятию окружающей среды и способность строить частичные представления об этой среде на основе воспринимаемой информации;
   f) способность к самоорганизации.
2) Состояние робота в момент времени t описывается функцией $X_i(t) = \{x_{1i}(t), x_{2i}(t), \dots, x_{mi}(t)\}$, где $x_{ji}(t), i = \overline{1,N}, j = \overline{1,m}$ – переменные состояния $i-$го робота ($N-$число роботов в системе).
3) $G = \{g_1, g_2, \dots, g_M\}$ – множество целей, характеризуемое вектором состояния $X_G = \{X_{G1}, X_{G2}, \dots, X_{GM}\}$.
4) Каждый робот системы обладает определенными наборами параметров, функций, а так же сенсорной системой для изучения окружающей среды.

1.1 Структура МРТС

Для реализации навигационной системы для группы мобильных роботов была разработана МРТС, позволяющая решать различные исследовательские и транспортные задачи (рис. 1).

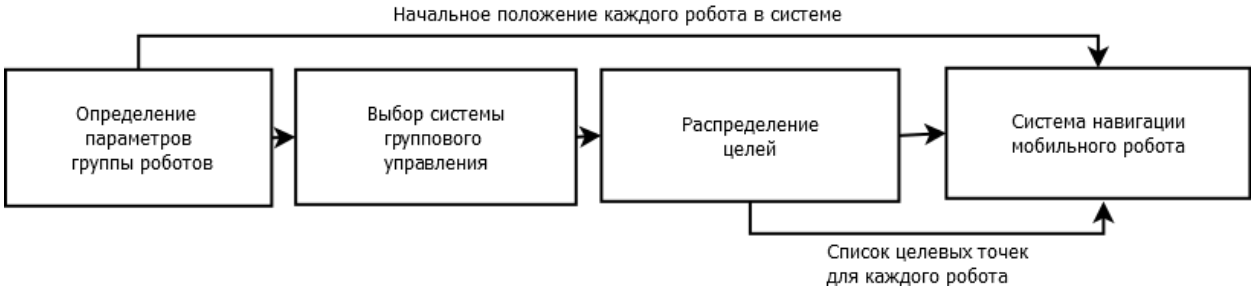

Рисунок 1. Структура МРТС

1) Определение ключевых параметров группы роботов:
   a) Количество роботов N;
   b) Начальное положение каждого робота в системе;
   c) Положение целевых точек.
2) Выбор одной из трех систем управления:
   a) Централизованная система;
   b) Децентрализованная система;
   c) Гибридная система.

   Критерии выбора системы и их ключевые особенности описаны в главе 1.2.

3) Распределение целей между роботами из группы. Описано в главе 2.
4) Реализация системы навигации для каждого робота в зависимости от его параметров. Различные варианты этой системы представлены в главе 3.

1.2 Виды систем группового управления

Все системы группового управления делятся на 3 основных типа:

1) Централизованные системы – системы, состоящие из группы мобильных роботов и центра управления, который распределяет цели между мобильными роботами, а также проводит локальное планирование для каждого из них. Таким образом в данной системе отсутствует необходимость установки мощных вычислительных средств для каждого робота, но нужен один компьютер, способный проводить все необходимые операции для каждого робота за короткий промежуток времени. Такие системы являются ненадежными, так как поломка одного компьютера приведет к свою всей системы. Кроме того, при увеличении числа подвижных объектов в группе, сложность задачи возрастает экспоненциально, что делает централизованную систему не эффективной для большого числа роботов

2) Децентрализованные системы - системы, состоящие из группы мобильных роботов, каждый из которых имеет на борту достаточную вычислительную мощность для реализации системы локального планирования. Данные системы являются более надежными, поскольку поломка одного робота не приведет к сбою всей системы, так как задачи этого робота могут быть быстро перераспределены среди остальных участников этой группы. Кроме того, при добавлении нового робота в группу, сложность задачи возрастает линейно, что делает данную системы более эффективной для большого числа роботов. Однако, отсюда следует один из главных недостатков этой системы: высокая стоимость каждого конкретного робота при большом их количестве.

3) Гибридные (смешанные) – системы, состоящие из подгрупп роботов, к каждой из которых привязан свой центр управления. Эта система используется при большом наборе роботов в группе и совмещает в себе преимущества и недостатки централизованных и децентрализованных систем.

Глава 2. Глобальное планирование

Одной из главных подзадач построения системы управления группой роботов является задача распределения целей между подвижными объектами этой группы. В этой главе приведены различные алгоритмы, позволяющие решить данную задачу.

2.1. «Ценовой» алгоритм распределения задач в коллективе роботов

Данный алгоритм используется для распределения задач при централизованной системе управления [21]. В нем для каждой j-й задачи, выполняемой i-м роботом, $i = \overline{1,n}$; $j = \overline{1,m}$, ставится в соответствие величина C_{ij} - «стоимость» выполнения этим роботом данной задачи. В качестве «стоимости» может рассматриваться время, необходимое роботу для решения задачи, либо длина пути при транспортных и навигационных задачах, рассматриваемых в этой работе. Кроме того, может быть использован параметр расхода топлива. Суть алгоритма заключается в минимизации общего расхода (т. е. минимизации «стоимости»).

Изначально распределяющий центр ставит каждому мобильному роботу в соответствие «стоимость» решения этим роботом каждой задачи. Эти данные центр получает с помощью методов локального планирования (глава 3), используя всю известную на данный момент информацию о расположении препятствий. В результате для каждого робота образуется массив «стоимостей» для каждой цели (таблица 1)

Таблица 1. Пример возможностей роботов в группе

№ робота \ № задачи	**1**	**2**	**…**	**m**
1	C_{11}	-	**…**	C_{1m}
2	C_{21}	C_{22}	**…**	-
…	**…**	**…**	**…**	**…**
n	-	-	**…**	C_{nm}

Если число роботов меньше количества задач $n < m$, то управляющий центр для каждого робота выбирает минимальную по «стоимости» задачу.

Остальные задачи распределяются между освободившимися после выполнения первоначальных задач роботами.

В случае, если $\mathrm{n} \geq \mathrm{m}$, управляющий центр выбирает m роботов, для которых затраты, необходимые для выполнения задач, минимальны.

Таким образом, если $n < m$:

Шаг 1. $i = 1$.

Шаг 2. Для i-го робота выбирается такая j-ая задача, для которой «стоимость» C_{ij} минимальная (т. е. выбирается минимальное значение в каждой строке таблицы 1). При наличии в строе одинаковых значения для разных задач, выбираем ту ячейку, для которой соответствующий номер задачи минимален.

Шаг 3. Из таблицы удаляются i-я строка и j-й столбец, $i = i + 1$.

Шаг 4. Возвращаемся к шагу 2. Повторяем алгоритм, пока $i < n + 1$.

Если $n \geq m$:

Шаг 1. $j = 1$.

Шаг 2. Для j -й задачи выбираем i-ого робота, для которого соответствующая «стоимость» C_{ij} минимальна (т. е. минимум в каждом столбце таблицы 1).

Шаг 3. Из таблицы удаляются i-я строка и j-й столбец, $j = j + 1$.

Шаг 4. Возвращаемся к шагу 2. Повторяем алгоритм, пока $j < m + 1$.

2.2. Мультиагентное распределение задач для группы мобильных роботов

Группа состоит из n роботов. Каждый из этих роботов может выполнять одну или несколько задач из списка типов задач l. Основная

задача разбивается на m простых задач. Для решения j-й задачи необходимо выделить группу из n_j мобильных роботов, где $j = \overline{1,m}$. C_{ij} - «стоимость» выполнения i-м роботом j-й задачи, $i = \overline{1,n}$..

Мультиагентное распределение построена на базе «Аукциона» и использует обмен информацией между роботами. Роботы устраивают «торги» между собой за право взять на себя ту или иную задачу. Возможности «покупки» задач роботами ограничена, а целесообразность покупки оценивается степенью полезности этой задачи, которая вычисляется как разность между «доходом» от задачи и затратами на его покупку. Один робот из группы является лидером (аукционером).

Алгоритм:

Шаг 1. Всем роботам сообщается количество и тип задач, а также число роботов, необходимое для их решения.

Шаг 2. Каждый робот рассчитывает массив «стоимостей» для каждой задачи, в который записывает «стоимость» выполнения отдельных задач. После получения этой информации от каждого робота формируется таблица (таблица 1).

Шаг 3. Каждый робот сортирует массив «стоимостей» в порядке возрастания.

Шаг 4. Определение роли лидера, которым становится незанятый робот с наименьшим порядковым номером.

Шаг 5. Лидер выбирает наименее затратную из своих задач и узнает, для кого из низ данная задача также является лучшей.

Шаг 6. Лидер создает структуру данных, содержащую номера ответивших роботов и соответствующие им «стоимости», а так же свои данные.

Шаг 7. В зависимости от числа элементов массива, сформированного на предыдущем шаге, возможны три ситуации, описанные в таблице 2 (n_k — число роботов, необходимое для решения k-той задачи).

Таблица 2. Возможные ситуации

Условие\ Действия			**Выбор всех роботов массива**	**Выбор роботов, предложив-ших наименьшую цену**	**Оповеще-ние роботов о выполнении задачи k**	**Исключение задачи k из ценовых массивов роботов**
Кол-во элементов массива	=	n_k	+	—	+	+
	<		+	—	+	—
	>		—	+	+	+

Шаг 8. Возвращаемся к шагу 4. Повторяем алгоритм, пока ценовые массивы роботов содержат хотя бы один элемент.

Глава 3. Локальное планирование

Для реализации системы группового управления каждый робот должен уметь совершать движение от одной точки к другой, избегая столкновения с препятствиями. В этой главе представлены наиболее эффективные для решения данной задачи методы.

3.1. Метод нейронных сетей

Сфера применения нейронных сетей очень широка благодаря возможностям, которые они открывают. Робототехника так же попала под их влияние, но несмотря на все возможности нейронных сетей, используются они не так часто. Такая ситуация сложилась по следующим причинам:

- Достаточно большой коэффициент неверно принятых решений. Это может привести к поломке дорогостоящего оборудования и невыполнению поставленных задач.
- Недостаточная функциональность. Нейронные сети не выполняют всех задач, необходимых для оптимального перемещения из одной точки в другую. В нейронной сети, приведенной ниже в этой главе есть как минимум один очень существенный недостаток, который не позволяет использовать этот алгоритм для практического решения поставленной задачи – перемещение происходит в направлении, не зависимом от положения конечной точки. Эту проблему возможно устранить, но она значительно усложнит полученную нейронную сеть.
- Получение не оптимальных решений. Из-за погрешности в полученных выходных векторах и, в последствии, неверно принятых решений, существует большая вероятность получения не оптимального результата.

Несмотря на все недостатки, нейронные сети имеют свои плюсы, благодаря которым их можно использовать в узком кругу задач, посвященных автономному движению мобильных роботов:

- Минимальные требования к аппаратуре. Робота, запрограммированного с помощью технологии нейронных сетей можно запустить даже на роботе, управляемом с помощью платы “Arduino”.
- Простота. Нейронные сети достаточно просты в разработке благодаря множеству библиотек для различных языков программирования.
- Увеличение функционала нейронной сети за счет добавления элементов во входной и(или) результирующий вектор, а так же с помощью увеличения числа слоев, количества нейронов в них и изменения активационных функций.

Таким образом применять нейронные сети целесообразно для решения некоторых простых задач с использованием недорогого оборудования. Подобную задачу описывает схема (рис. 2)

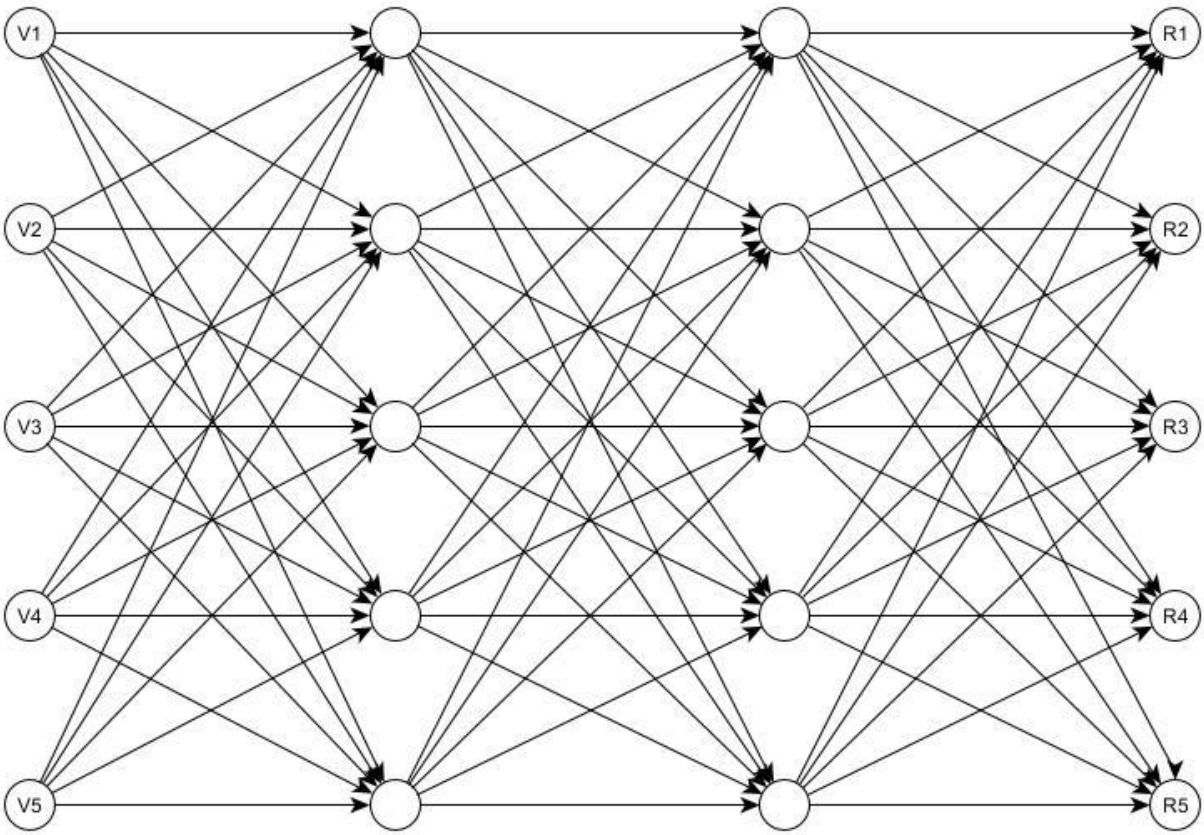

Рисунок 2. Структура нейронной сети

Здесь $V = [V1, V2, V3, V4, V5] -$ входной вектор, $V1 - V3 -$ расстояния, полученные с помощью дальномера, $V4, V5$ – показания с других

датчиков (в данном примере они равны 0, так как другие датчики не используются. Добавлены для возможности быстрого расширения возможностей нейронной сети).

Вектор $R = [R1, R2, R3, R4, R5]$ – результирующий вектор, состоящий из одной единицы и четырех нулей. Элемент равный единице соответствует действию, которое должен совершить робот.

При построении матриц весовых коэффициентов были использованы активационные функции:

1. Relu
2. Relu
3. Sigmoid

Для реализации данной сети был использован язык “Python” с использованием библиотек “Keras”, “NumPy”, “Pandas”. В результате работы данного программного обеспечения были получены следующие матрицы весовых коэффициентов:

1. [[0.20042136, 0.6265539 , -0.13393332, -0.05438387, 0.0460667],
   [0.22865786, -0.69653577, 0.8663023 , -0.57229203, -0.34758186],
   [0.05971239, 0.01293267, 0.11849754, -0.5440213 , 0.11720096],
   [-0.11901385, 0.23858422, 0.10948835, -0.53810453, 0.15719718],
   [-0.17215312, 0.22754352, 0.29704425, -0.24777633, 0.13172989]]
2. [[0.7429964 , 0.05953575, 0.4352131 , -1.0002222 , -0.58549947],
   [0.50720984, -0.22808188, 0.71975374, -0.35620582, 0.30150846],
   [0.74406314, 0.2290597 , -0.29957077, -0.25139022, -0.5864916],
   [0.6937592 , -0.6200428 , 0.05410463, 0.03972965, 0.24807107],
   [-0.89941967, 0.6996742 , 0.5873009 , 0.8305814 , -0.9807337]]
3. [[0.27732217, -3.70134 , -1.0843853 , -1.4565762 , -1.5560904],
   [-0.5293996 , -3.9536407 , -0.5888512 , 0.08424984, -2.0163887],
   [-0.396352 , -4.7015724 , 0.36925882, -0.5129481 , -0.8023801],
   [-1.9868035 , -1.7674725 , -2.3748152 , -0.755013 , -0.36411628],
   [-0.05135759, 0.37764692, -0.11632747, -0.6868904 , 0.6364576]]

Применение данные весовых коэффициентов к описанной сети позволяет получать корректный результат примерно в 997 случаях из 1000.

3.2 Метод потенциальных полей

Метод потенциальных полей – один из наиболее популярных и часто используемых методов в задачах, реализующих автономное движение подвижных объектов в среде с препятствиями.

Основные преимущества использования данного метода:

- Один из наиболее простых, но в тоже время эффективных алгоритмов навигации.
- Не требователен к аппаратуре. Без использования различных модификаций может быть реализован даже на “Arduino”.
- На данный момент известно множество различных модификаций и улучшений данного метода.

Но данный метод не лишен и недостатков:

- Данный метод не всегда эффективен для решения задач, в которых требуется достигать сложных целей.
- Высокая вероятность попадания объекта в локальный минимум.

Метод заключается в том, что множество объектов в пространстве делятся на два типа: аттракторное и репеллерное. Аттракторное множество образуется целевой точкой. Оно придает объекту силу, направленную от объекта к цели и независящую по модулю от расстояния между ними. Репеллерное множество образуется препятствиями. Оно придает объекту силы, значение которых по модулю пропорционально зависит от расстояния между объектом и препятствием, а направление вектора силы противоположно направлению от объекта к препятствию. Возможный результат работы метода представлен на рис. 3.

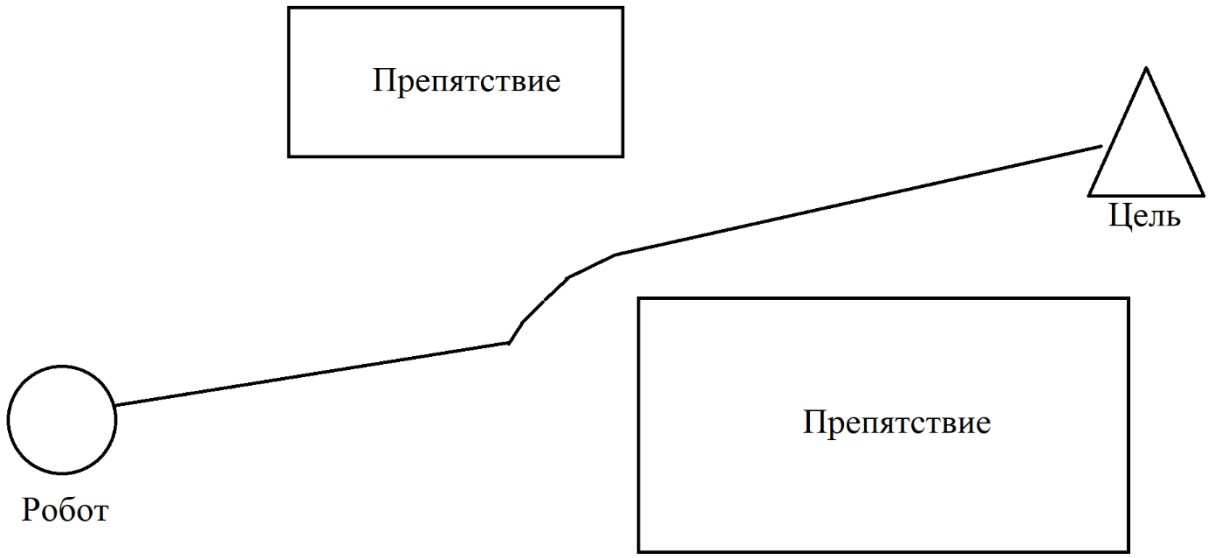

Рисунок 3. Метод потенциальных полей

Данный метод нуждается в применении некоторых модификаций для успешной работы. Наиболее часто используемым улучшением является применение фильтра Калмана, позволяющего минимизировать помехи, образующиеся при прохождении объекта рядом с препятствиями. Данный метод в большинстве случаев эффективнее, чем применение нейронных сетей, но в некоторых более сложных задачах уступает системе навигации, основанной на использовании алгоритма SLAM.

3.3 Система навигации мобильного робота

3.3.1 Описание движения подвижного объекта

Уравнения движения подвижного объекта:

$$\begin{cases} \dot{v} = a \\ \dot{\omega} = \varepsilon \\ \dot{\varphi} = \omega \\ \dot{x} = v * \cos(\varphi) \\ \dot{y} = v * \sin(\varphi) \end{cases},$$

Где x, y – координаты, соответствующие центру окружности, описывающей мобильного робота, φ – курсовой угол, v, a – линейные скорость и ускорение, ω, ε – угловые скорость и ускорение.

3.3.2 Структура системы навигации

Рассмотрим полученную структуру системы навигации автономного мобильного робота (рис. 4).

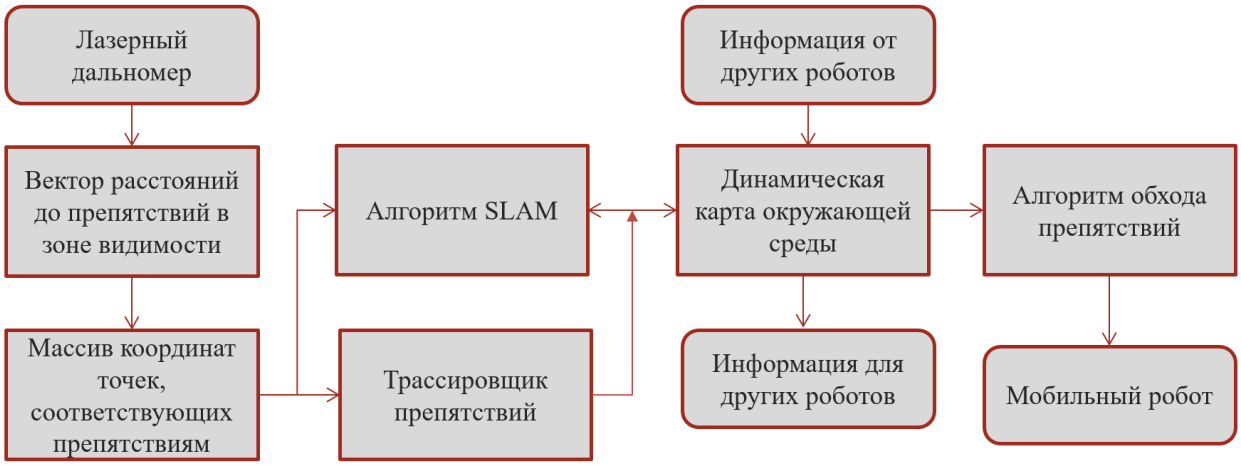

Рисунок 4. Структура системы навигации

Рассмотрим по отдельности некоторые элементы данной системы:

- Лазерный дальномер – устройство, позволяющее сканировать окружающее пространство и получать данные о находящихся в данном пространстве объектах (препятствиях) в виде векторов расстояний;

- Алгоритм SLAM (Simultaneous Localization and Mapping) - алгоритм, разработанный для локализации мобильного робота в пространстве, а так же построения динамической карты окружающей среды. Данный алгоритм более подробно описан в 1.3;

- Трассировщик препятствий на основе имеющихся данных о текущем местоположении робота и текущего скана карты окружающей среды строит список подвижных препятствий и прогнозирует их местоположение в следующие моменты времени. Более подробно данная задача описана в 1.5;

- Алгоритм обхода препятствий – алгоритм, позволяющий построить траекторию обхода препятствий мобильным роботом при имеющихся данных о местоположении робота в данный момент времени и текущем скане

карты окружающей среды. Различные алгоритмы обхода препятствий рассмотрены в 2.2;

- Мобильный робот – автономный робот, движущийся по полученной траектории с помощью некоторого управления.

3.3.3 Алгоритм SLAM

Алгоритм SLAM необходим для создания мобильных роботов, способных автономно перемещаться в недетерминированной среде. Данный алгоритм имеет две основные функции:

- Локализация мобильного робота – получение информации о нахождении робота в текущий момент времени;

- Построение динамической карты окружающей среды.

Алгоритм SLAM для построения решения использует фильтр Калмана.

Фильтр Калмана – эффективный рекурсивный фильтр, который оценивает состояние динамической системы по ряду неточных измерений. Был разработан в 1960 году и назван в честь Рудольфа Калмана. Данный фильтр необходим для устранения погрешности алгоритма SLAM, вызванного одометрией. Фильтр Калмана обрабатывает входные данные, то есть положение робота и особых точек, и возвращает их оценочные значения.

Принцип работы алгоритма SLAM:

1. Робот находится в некотором неизвестном месте. С помощью данных, полученных датчиками, происходит построение видимого участка карты с данной позиции;

2. С помощью полученной на данном шаге траектории выбирается следующая позиция для передвижения;

3. Происходит передвижение на новую позицию и сравнение текущее положение с ожидаемым, полученным на предыдущем шаге;

4. По полученным данным и данным с предыдущей итерации происходит обновление карты. Далее переходим к п. 2.

3.3.4 Построение карты окружающей среды в виде карты-сетки

Карта окружающей среды представлена в форме карты-сетки. Размер ячеек выбирается в зависимости от необходимой точности. При этом, чем меньше размер ячеек, тем дольше будет работать алгоритм обхода препятствий.

Полученные ячейки разделяем на два типа (рис. 5):

- свободные - это те ячейки, через которые робот может совершать беспрепятственное движение);
- ячейки, содержащие в себе препятствия.

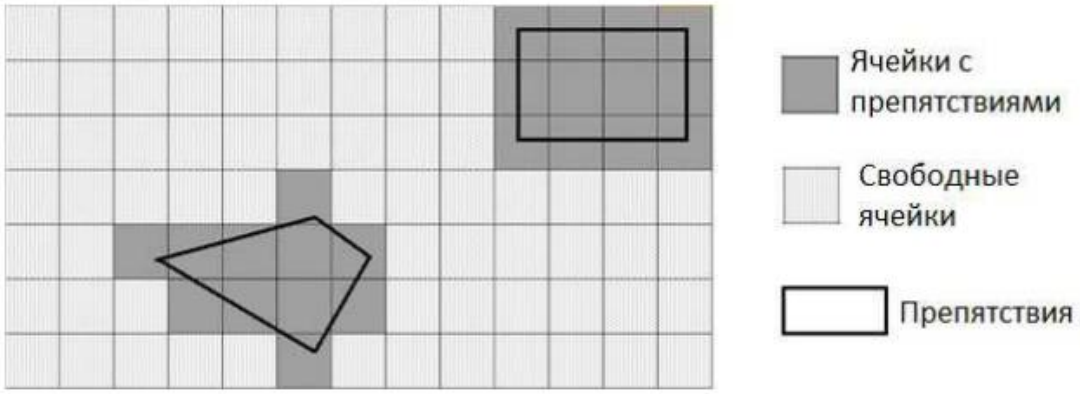

Рисунок 5. Пример карты окружающей среды

Для роботов, способных двигаться во всех направлениях на плоскости, размер ячейки обычно соответствует габаритам робота, то есть контур робота можно вписать в данную ячейку. Для достижения более высокой точности возможно уменьшение размера ячеек, но при этом алгоритм построения

траектории необходимо изменить. Пример карты, при которой робот занимает несколько ячеек, продемонстрирован в 2.2.3

3.3.5 Трассировка препятствий

Трассировка подвижных препятствий позволяет определять местоположение в следующие моменты времени и параметры движения мобильного робота в системе координат, связанной с окружающей средой. Большинство препятствий перемещаются без смены направления. Благодаря этому мы можем с помощью трассировки препятствий спрогнозировать их движение.

Мы можем применить трассировку к препятствию если:

- Препятствие является твердым телом;
- Подчиняется уравнениям движения робота;
- Не может скачкообразно менять скорость;
- Его можно вписать в окружность.

Каждое такое препятствие обладает следующим набором параметров:

$$\begin{cases} \dot{v} = a \\ \dot{\omega} = \varepsilon \\ \dot{\varphi} = \omega \\ \dot{x} = v * \cos(\varphi) \\ \dot{y} = v * \sin(\varphi) \end{cases},$$

Где x, y – координаты, соответствующие центру окружности, описывающей препятствие, φ – курсовой угол, v, a – линейные скорость и ускорение, ω, ε – угловые скорость и ускорение.

Вектор состояния препятствия выглядит следующим образом:

$$s = (x\ y\ \varphi\ v\ \omega\ a\ \varepsilon\ R)^T$$

Принцип работы трассировщика препятствий:

1. Создание списка подвижных препятствий;

2. Определение радиуса окружности R, описывающей препятствия из полученного списка;

3. Составление вектора состояния s для каждого препятствия.

4.

3.4 Планирование пути мобильного робота

3.4.1 Общая структура планирования

Несмотря на то, что различные алгоритмы предназначены для построения траектории в различных ситуациях, можно выделить общую структуру планирования (рис. 6) для построения траектории.

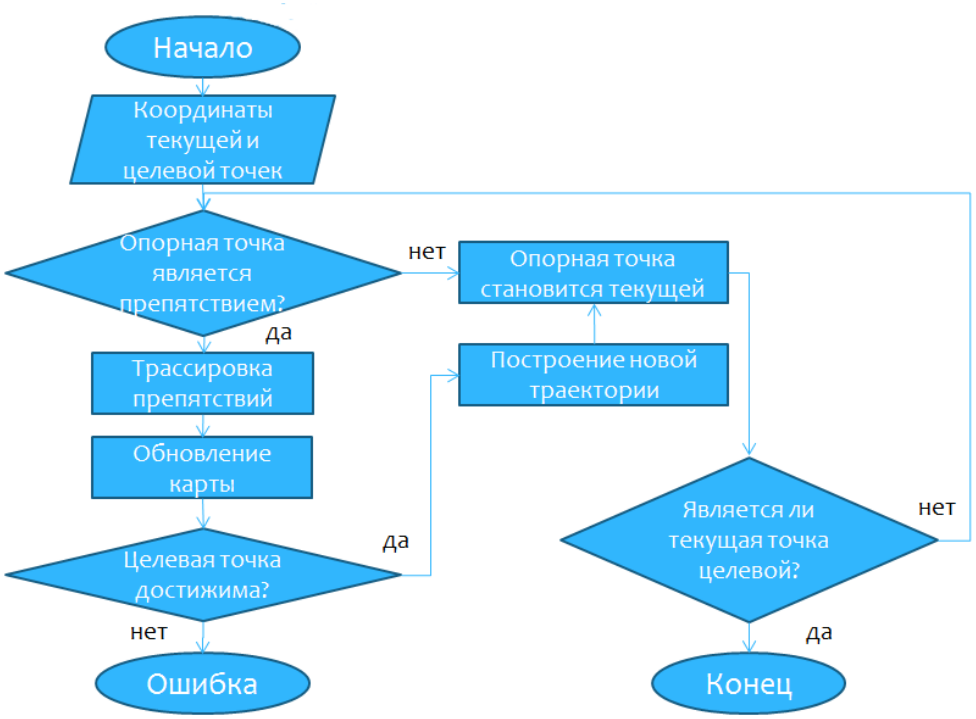

Рисунок 6. Общая структура планирования.

3.4.2 Алгоритмы обхода препятствий

Для построения траекторий в разных ситуациях необходимо применять разные алгоритмы:

- Детерминированная среда со стационарными препятствиями - алгоритм A*;
- Недетерминированная среда – модифицированный алгоритм D*.

Кроме того, алгоритм может изменяться в силу различных динамических свойств рассматриваемого подвижного объекта, например при рассмотрении задач построения траектории для колесного и гусеничного робота.

3.4.3 Алгоритм A* (A-star)

Алгоритм “A*” является улучшенной модификацией алгоритма Дейкстры, разработанной для более быстрого нахождения оптимальной траектории с помощью эвристики. В данной работе рассматривается эвристическая функция $H = \sqrt{(x - x_к)^2 + (y - y_к)^2}$, где x, y – координаты рассматриваемой точки, $x_к, y_к$ – координаты целевой точки.

Принцип работы алгоритма

1. Строим карту окружающей среды, отмечая на ней так же начальное положение робота и целевую точку (рис.5)

2. Создаем 2 списка ячеек – открытый и закрытый. В открытом списке изначально находится только ячейка, соответствующая начальному положению робота, в закрытом – ячейки содержащие препятствия (рис. 6).

3. Вводим функцию $L = H + l$, где H – эвристическая функция, l – расстояние от начального положения робота до текущего, которая отображает минимальное расстояние от начальной точки до целевой при построении траектории через данную точку.

4. Сортируем открытый список по возрастанию функции L. Берем за рассматриваемую ячейку первую из данного списка.

5. Помещаем данную точку в закрытый список. Рассматриваем все соседние точки, не принадлежащие закрытому списку (рис. 7). Добавляем их в открытый список. Если одна из рассмотренных точек является целевой – переходим к пункту 6, иначе – к пункту 4.

6. По полученным данным строим траекторию.

3.4.4 Алгоритм A* (A-star) для колесных роботов

Некоторые виды колесных роботов имеют ограниченными возможности в выборе направления для дальнейшего перемешения (рис. 7) Робот представляем в форме прямоугольника с соотношением сторон 1:2.

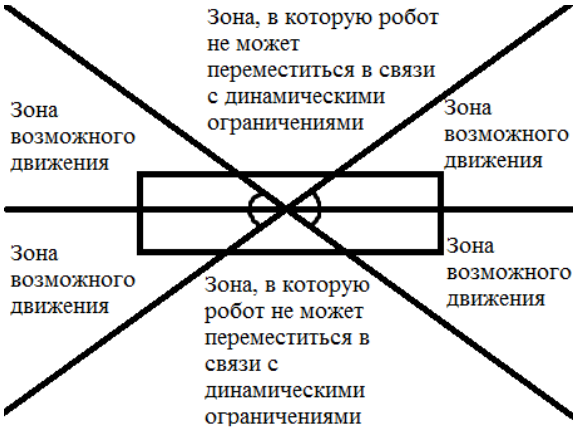

Рисунок 7. Возможные направления движения колесного робота.

Рассматривается движение, при котором сохраняется угол поворота относительно рассматриваемой системы координат после перемещения в любом из возможных направлений.

Из-за ограничений, наложенных на возможность выбора направления движения колесным мобильным роботом, выделяем шесть основных направлений движения (рис. 8). Для определения местоположения робота в текущий момент времени необходимо знать только координату, соответствующую левому верхнему краю робота.

Так как стандартный алгоритм A* исследует восемь направлений возможного движения, а мы рассматриваем только шесть, то его необходимо изменить.

Принцип работы модифицированного алгоритма A*:

1. Строим карту местности, отмечая на ней начальное положение робота и целевую точку (рис. 5). В данной модификации начальному положению робота будет соответствовать левая верхняя ячейка, принадлежащая роботу.

2. Создаем два списка ячеек – открытый и закрытый. В открытом списке изначально находится только ячейка, соответствующая начальному положению робота, в закрытом – ячейки, содержащие препятствия. Первоначальное графическое отображение открытого и закрытого списка представлено на рис. 6.

3. Введем функцию $L = H + l$, где H – эвристическая функция, l – расстояние от начального положения робота до текущего.

4. Сортируем открытый список по возрастанию функции L, соответствующей ячейке.

5. Берем за рассматриваемую ячейку первую из получившегося списка.

6. Рассматриваем шесть соседних ячеек, в которые можно попасть за одну итерацию. Если данные ячейки, а также ячейки, необходимые для совершения движения из предыдущей ячейки в текущую не принадлежат закрытому списку, то добавляем их в открытый, иначе – в закрытый (см. рис. 8).

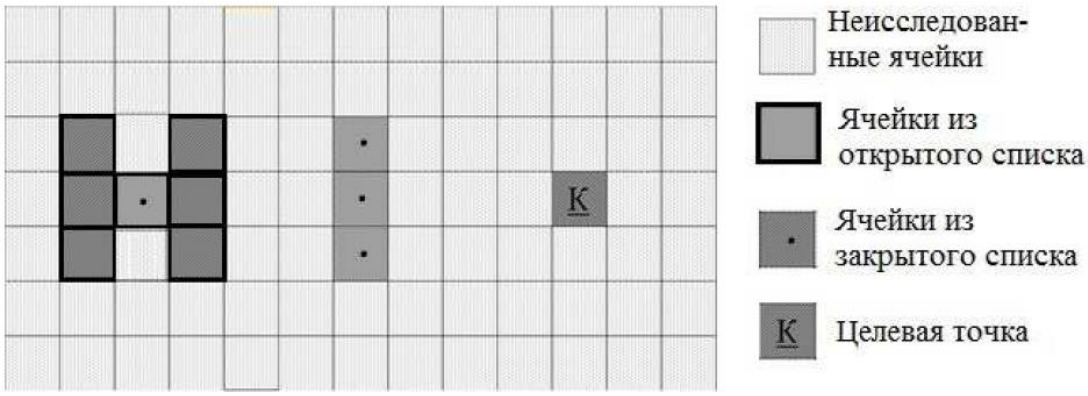

Рисунок 8. Изменение открытого и закрытого списка

7. Если в открытом списке есть ячейка, соответствующая целевой точке, то переходим к следующему пункту, иначе – возвращаемся к пункту 2.

8. По полученным данным строим искомую траекторию.

3.4.5 Алгоритм D* (D-star)

Робот может совершать движение только по заданной сетке, то есть на каждом новом шаге у него имеется четыре различных варианта совершения движения – вверх, вниз, влево, вправо (рис. 9).

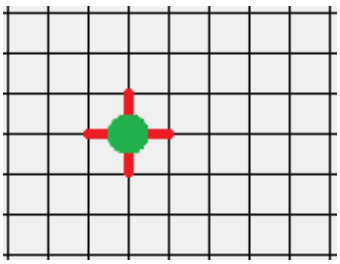

Рисунок 9. Возможные направления движения.

Алгоритм “D*” основан на использовании алгоритма “A*”, позволяющего строить траекторию обхода препятствий, оптимальную по длине пути.

Принцип работы алгоритма:

1. Строим первоначальную карту окружающей среды, отмечая на ней начальное положение робота и целевую точку (рис. 10).

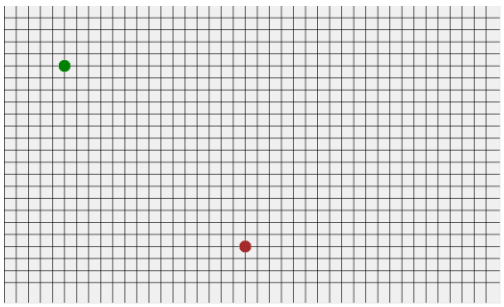

Рисунок 10. Первоначальная карта окружающей среды.

2. Создаем список истории, хранящий информацию о всех ячейках, рассмотренных в ходе каждого построения траектории. С помощью алгоритма “A*” строим траекторию из целевой точки в начальную (рис. 11), добавляя все рассмотренные ячейки с список истории. Все элементы данного списка имеют следующие характеристики:

a. Вес: $\begin{cases} \omega = 1, & \text{если ячейка не содержит препятствие} \\ \omega = 999999, & \text{если ячейка содержит препятствие} \end{cases}$;

b. Предыдущая (k-1) ячейка;

c. длина пути из целевой ячейки: $\rho(k) = \omega(k) + \rho(k-1)$;

d. текущие координаты x и y.

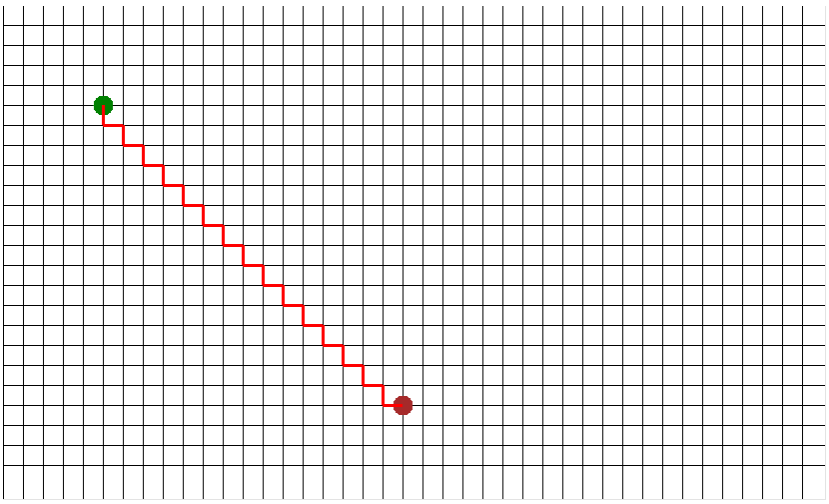

Рисунок 11. Первоначальная карта окружающей среды.

3. Рассматриваем движение по полученной траектории от начальной точки к целевой. На каждом шаге делаем проверку (рис.12): если следующая

ячейка данной траектории не содержит препятствие и не является целевой – двигаемся дальше, если содержит препятствие – прекращаем движение и переходим к пункту 4, если является целевой – завершаем работу алгоритма.

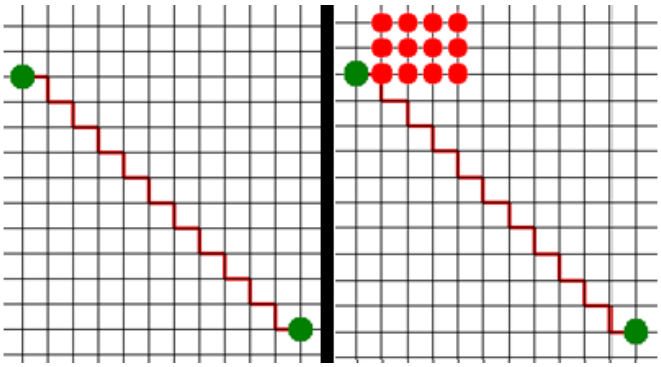

Рисунок 12. Совершение движения и проверка следующей точки.

4. Если хотя бы одна из соседних ячеек (k_{new}) доступна в списке истории и обновленное расстояние от этой ячейки до целевой $\rho(k_{new}) \leq \rho(k) + 3$, то переходим к пункту 5, иначе – делаем текущую ячейку начальным положением робота и переходим к пункту 2.

5. Из соседних ячеек, удовлетворяющих условию из пункта 4, выбираем ту, расстояние от целевой точки до которой наименьшее и переходим к пункту 6.

6. С помощью известной для каждой ячейки информации о предыдущей ячейке, рекурсивным методом строим новую траекторию, проходящую через полученную в пункте 5 ячейку и переходим к пункту 3

3.4.5 Модифицированный алгоритм D* (D-star)

Алгоритм D* имеет ряд недостатков, из-за которых применение его на реальных подвижных устройствах может быть затруднительно. В данной работе представлена модификация этого алгоритма с двумя исправленными недостатками, значительно облегчающими его практическую реализацию:

1. Вместо четырех направлений движения робота рассматривается восемь.

2. Данная модификация алгоритма учитывает, в каком направлении остановился робот в каждой конкретной точке своего движения.

Постановка задачи. Цель — разработать алгоритм, позволяющий мобильному роботу автономно перемещаться в недетерминированной среде из одной позиции в другую, избегая столкновения с препятствиями. Данный алгоритм должен учитывать положение робота в каждой конкретной точке его траектории.

Требования к мобильному роботу. Для работы этого алгоритма требуется хранение достаточно больших объемов информации, поэтому не каждое устройство может работать под его управлением. Необходимые компоненты:

1. Эффекторы (двигатели, колеса/гусеничное шасси).

2. Сенсорная система, состоящая из двух инфракрасных дальномеров, один из которых расположен спереди на серво-приводе, а второй — сзади. Также желательно наличие гироскопа и акселерометра для более высокой точности.

3. Основная плата «Raspberry Pi».

4. Motor shield (плата управления двигателем).

5. Sensor shield (плата для подключения датчиков).

Принцип работы алгоритма:

1. Задаем на карте ячейки, соответствующие начальному положению робота и целевой точке.

2. С помощью алгоритма A* строим начальную траекторию движения робота. Эта траектория ищется от целевой точки до начального положения, что позволяет знать точное расстояние от каждой рассмотренной точки до целевой. Добавляем все рассмотренные точки в список истории, в котором

хранится следующая информация о каждой рассмотренной точке: • вес m, равный сумме веса предыдущей точки и расстояния от нее до текущей. Вес целевой точки равен нулю, а вес каждой ячейки, содержащей препятствие, стремится к бесконечности;

• предыдущая $(k - 1)$ ячейка, т. е. та, которая будет следующей точкой в траектории, полученной с помощью алгоритма A*;

• длина пути из целевой ячейки $p(k) = m(k) + p(k - 1)$;

• координаты x и y.

3. Рассматриваем движение по полученной траектории от начальной точки к целевой. Примем за направление робота в точке координату точки, в которую попадем при движении вперед. Примем за начальное — направление, в котором робот находится в начальной точке при запуске алгоритма (x_1, y_1), где $x, y = \{-1, 0, 1\}$.

4. Для нахождения направления от одной точки до следующей по траектории необходима следующая информация:

• координаты текущей точки;

• направление робота в текущей точке;

• координаты следующей точки на траектории. Для движения роботу нужно знать следующие значения: направление dir ($dir = -1$ — назад, $dir = 1$ — вперед), направление поворота $turn$ ($turn = -1$ — направо, $turn = 1$ — налево, $turn = 0$ — без поворота), угол поворота φ (от 0 до 90 градусов).

Составим два вектора:

• вектор $\vec{a}$ с началом в текущей точке, концом — в координате, соответствующей направлению робота в этой точке;

• вектор $\vec{b}$ с началом в текущей точке, концом — в следующей точке на траектории. Угол между векторами $\vec{a}$ и $\vec{b}$ обозначим за φ_0. Находим по формуле $\cos \varphi_0 = \frac{\vec{a} * \vec{b}}{|\vec{a}| * |\vec{b}|}$, откуда $\varphi_0 = \arccos \varphi_0$. Если $\varphi_0 \leq 90°$, то для данной точки переменные состояния $dir = 1$, а $\varphi = \varphi_0$, иначе, $dir = -1$, а $\varphi = 180 - \varphi_0$. Направление поворота находим с помощью векторного произведения. Для этого положим координату по оси z у векторов $\vec{a}$ и $\vec{b}$ равной нулю и рассмотрим знак компоненты z у результата их векторного произведения: если $x_a * y_b - y_a * x_b < 0$, то $turn = -1$, если $x_a * y_b - y_a * x_b > 0$, то $turn = 1$, иначе, $turn = 0$.

5. На каждом шаге делаем проверку: если следующая ячейка данной траектории не содержит препятствие, то двигаемся дальше, если содержит — прекращаем движение и переходим к пункту 6, если является целевой — завершаем работу алгоритма.

6. Если хотя бы одна из соседних ячеек k_{new} доступна в списке истории и обновленный вес этой ячейки $m(k_{new}) \leq m(k) + \alpha$, где α — коэффициент, определяющий какой из критериев важнее: длина пути или время работы алгоритма (оптимальная по длине траектория при $\alpha = 3$), то переходим к пункту 7, иначе, делаем текущую ячейку начальным положением робота и переходим к пункту 2.

7. Из соседних ячеек, удовлетворяющих условию из пункта 6, выбираем ту, расстояние от целевой точки до которой наименьшее, и переходим к пункту 8.

8. С помощью известной для каждой ячейки информации о предыдущей ячейке совершаем движение от каждой новой рассмотренной точки к предыдущей, пока не попадем в целевую точку.

Глава 4. Результаты программной реализации

На основе теоретических сведений, изложенных в этой работе, были разработаны два мобильных робота:

1) Гусеничный робот на базе “Arduino”, оснащенный ультразвуковым дальномером, закрепленным на серво-приводе. Реализован метод нейронных сетей, позволяющий роботу совершать движение, избегая столкновений с препятствиями.
2) 4-колесный робот на базе “Raspberry Pi 3B”, оснащенный инфракрасным дальномером, закрепленном на сервоприводе и ультразвуковым дальномером. Реализован модифицированный алгоритм D*, позволяющий мобильному роботу совершать движение от одной точки до другой, избегая столкновений с препятствиями.

На языке C# разработано ПО, реализующее рассмотренные алгоритмы построения траектории обхода препятствий. В данной главе приведены примеры работы данного ПО и результаты, полученные в ходе сравнения исследуемых алгоритмов.

4.1 Построение карты препятствий

Пример построения карты-сетки, полученный с помощью созданного ПО (рис. 13):

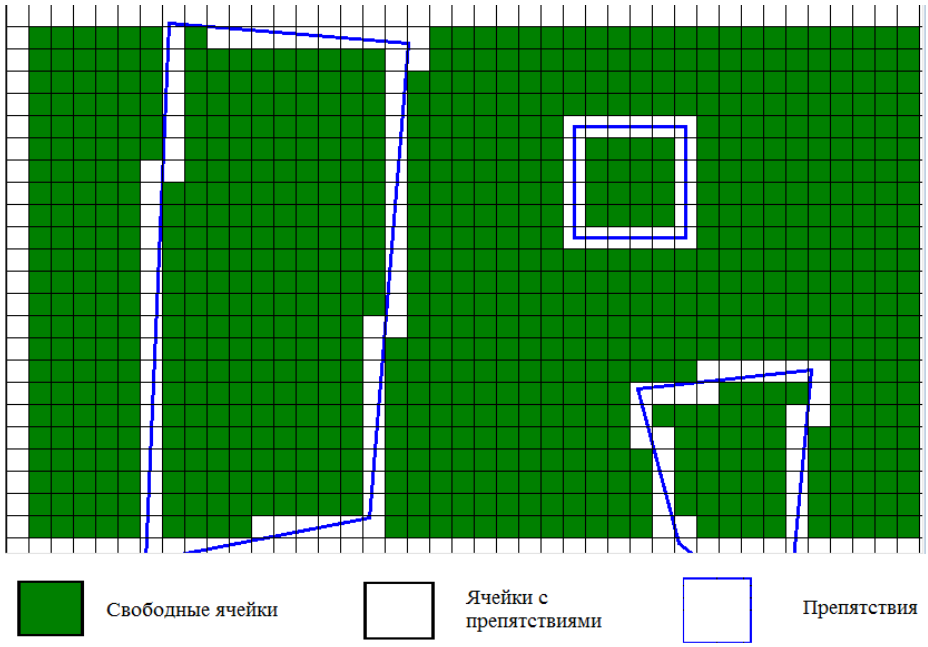

Рисунок 13. Пример карты сетки

4.2 Реализация рассмотренных алгоритмов

В данном разделе представлены ключевые моменты в работе следующих алгоритмов:

- Алгоритм A* (рис. 14);
- Алгоритм D* (рис. 15-19).

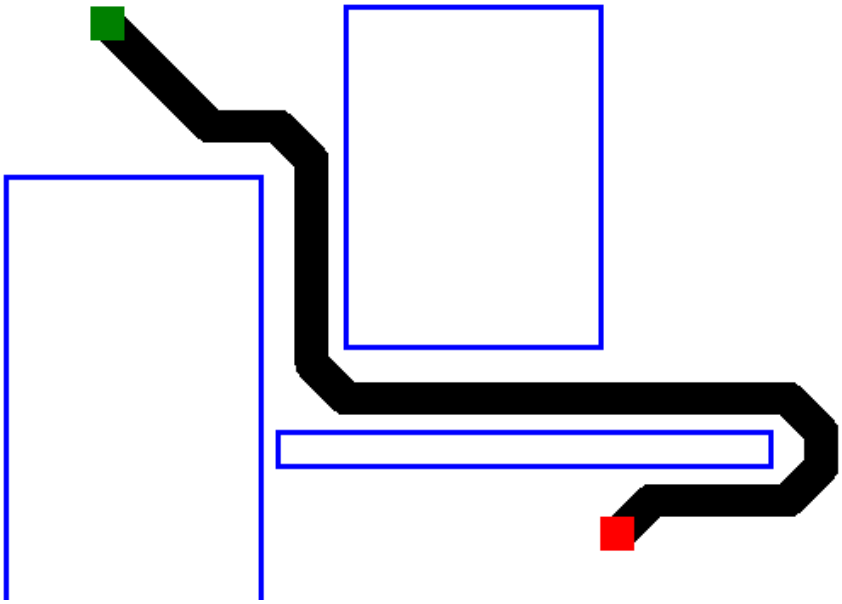

*Рисунок 14. Построение траектории алгоритмом A**

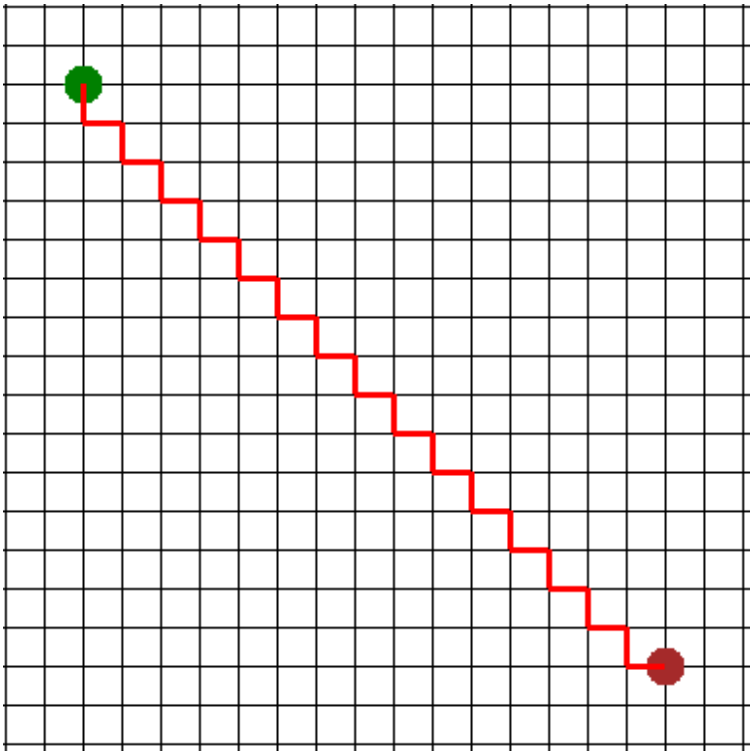

Рисунок 15. Робот строит первоначальную траекторию в недетерминированной среде.

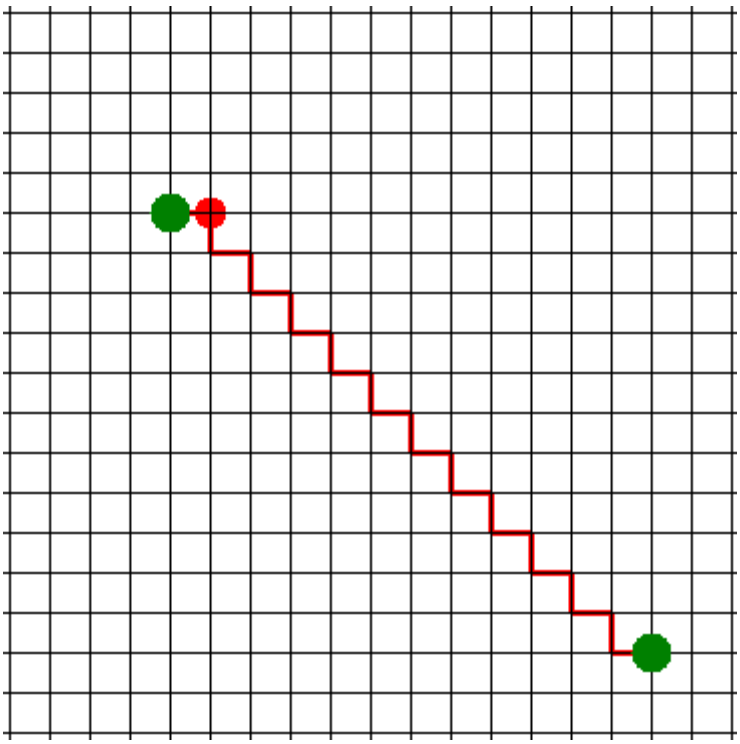

Рисунок 16. Во время движения робот находит на следующем шаге препятствие.

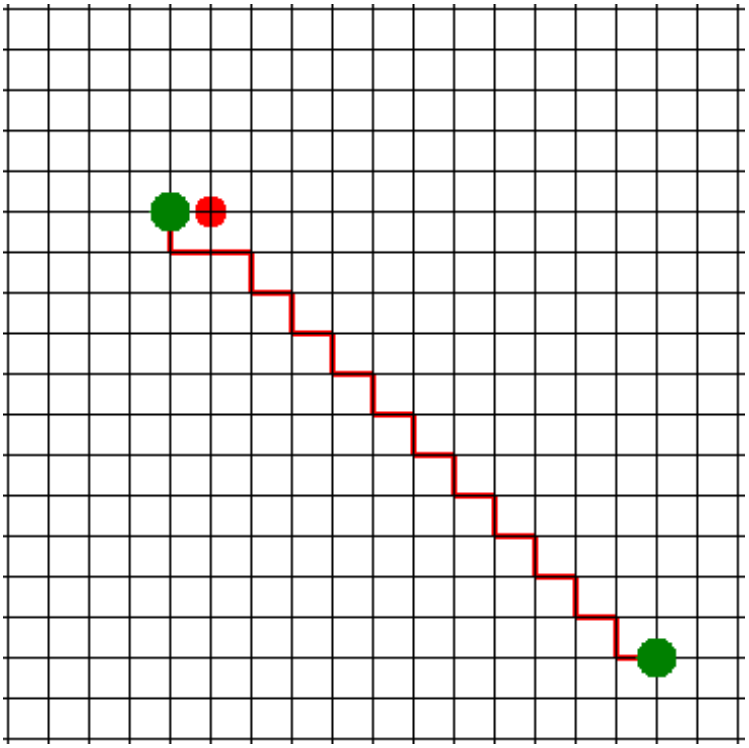

Рисунок 17. Робот обходит препятствие с помощью соседней точки из списка истории

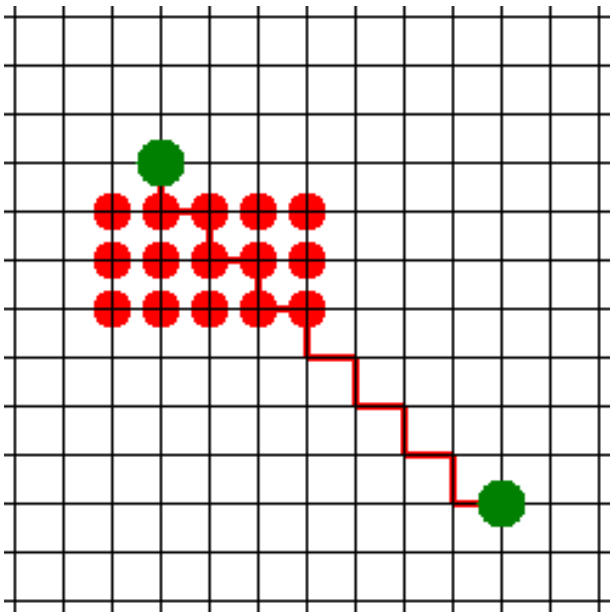

Рисунок 18. Во время движения робот находит на следующем шаге препятствие.

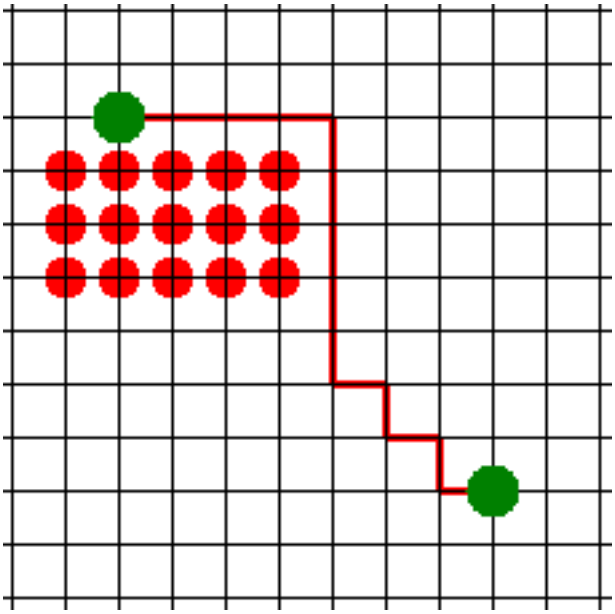

Рисунок 19. Робот обходит препятствие с помощью построения новой траектории.

4.3 Сравнение точности и скорости алгоритмов

На основе данных, полученных с помощью программного обеспечения, реализующего данные алгоритмы, можно сделать вывод, что алгоритм “A*” в большинстве случаев строит траекторию обхода препятствий за более короткое время, чем алгоритм Дейкстры (рис. 20). Длина полученного пути у алгоритмов одинакова.

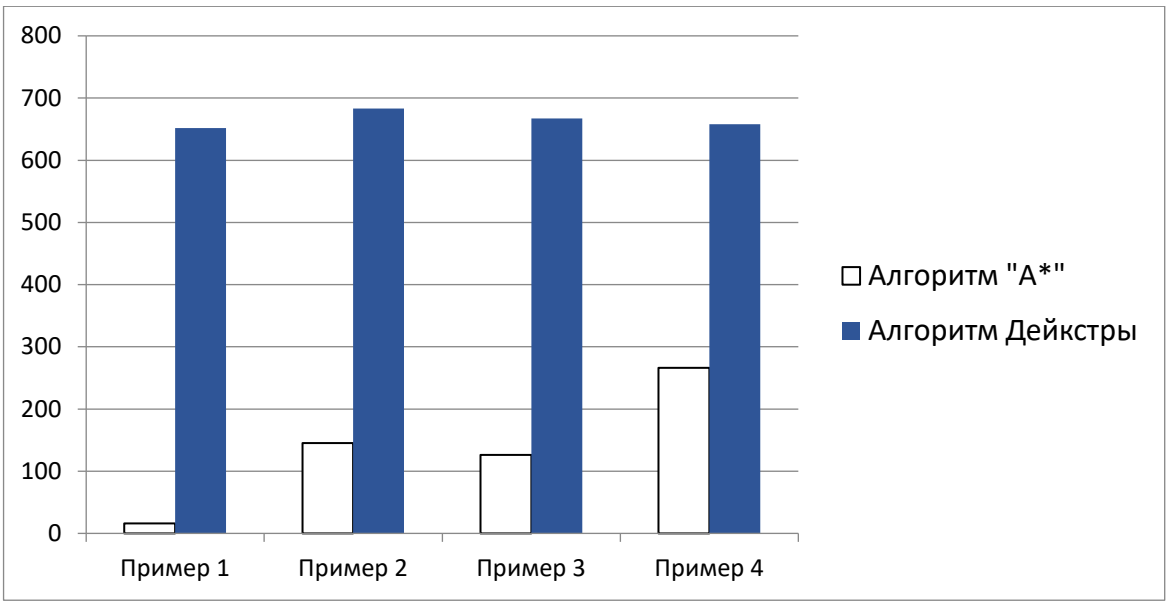

Рисунок 20. Сравнение времени работы алгоритма Дейкстры и алгоритма A.*

С помощью разработанного программного обеспечения, реализующего работу алгоритмов D*, A* и LPA* в недетерминированной среде, было проведено сравнение времени работы данных алгоритмов в различных ситуациях. Полученная информация изображена на рис. 21 (время указано в миллисекундах).

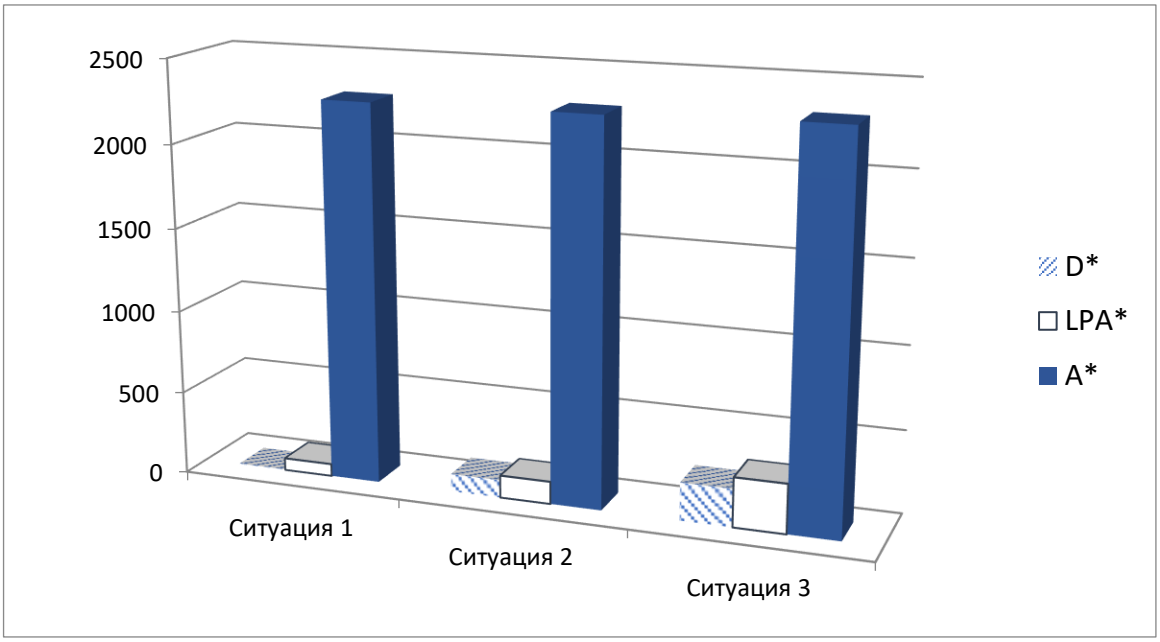

Рисунок 21. Сравнение времени работы алгоритмов в недетерминированной среде.

Выводы

- Представлена МРТС, позволяющая группе автономных роботов выполнять различные поисковые и транспортные задачи.
- Исследованы и представлены различные методы и алгоритмы группового управления
- Представлена навигационная система, позволяющая подвижному объекту автономно строить оптимальную по длине пути траекторию обхода препятствий от начальной точки до целевой.
- Исследованы и программно-реализованы рассмотренные алгоритмы. Приведены некоторые результаты работы программного обеспечения.
- Произведен сравнительный анализ по времени работы алгоритмов.
- Рассмотрена система навигации, позволяющая применить полученные теоретические сведения на практике.
- Полученные теоретические сведения успешно реализованы на реальных мобильных роботах.

Заключение

Проблема создания навигационной системы, позволяющей группе подвижных объектов совершать автономное движение в реальных средах очень важна в современном мире. Все больше задач вместо людей выполняют некоторые сервисные роботы. С течением времени большинство процессов по производству материальных ценностей, исследованию новых территорий (в том числе в космосе) и обслуживанию людей станут выполнять группы автономных роботов.

Массовое производство автономных роботов, способных работать в сложны условиях, значительно упростит жизнь людей. Никому не придется рисковать своей жизнью, выполняя работу.

Создание некоторого универсального метода, способного автоматизировать передвижение группы роботов в различных средах, будет огромным шагом на пути к созданию полностью автономных и многофункциональных роботов. В связи с этим данная задача в настоящее время действительно является актуальной и требующей нахождения более оптимальных решений по многим параметрам, таким как минимизация расхода топлива, уменьшение погрешности вычислений датчиками расстояний до объектов внешней среды и возможность создавать группы роботов, способные общими усилиями выполнять одну задачу, которую мобильный робот не сможет выполнить в одиночку.

Список литературы

1. Бойчук Л. М. Метод структурного синтеза нелинейных систем автоматического управления. М.: Энергия, 1971. 112 с.
2. Бойчук Л.М. Синтез координирующих систем автоматического управления. М.: Энергоатомиздат, 1991. 160 с.
3. Пшихопов В. Х. Позиционно-траекторное управление подвижными объектами. Таганрог: Изд-во ТТИ ЮФУ, 2009. 183 с.
4. Топчиев Б. В. Синергетическое управление мобильными роботами // Нелинейный мир, 2004. Т.2, №4. С. 239-249.
5. Montaner M. B., Ramirez-Serrano A. Fuzzy knowledge-based controller design for autonomous robot navigation // Expert Systems with Applications, 1998. Vol. 14 (1-2), P. 179-186.
6. Lee T-L., Wu C-J. Fuzzy motion planning of mobile robots in unknown environments // Journal of Intelligent and Robotic Systems, 2003. Vol. 37 (2), P. 177-191.
7. Пшихопов В. Х., Медведев М. Ю., Крухмалев В. А. Позиционно-траекторное управление подвижными объектами в трехмерной среде с точечными препятствиями // Известия ЮФУ. Технические науки, 2015. №1 (162). С. 238-250.
8. Алферов Г. В. Генерация стратегии робота в условиях неполной информации о среде // Проблемы механики и управления: Нелинейные динамические системы, 2003. №35. С. 4-23.
9. Alferov G. V., Malafeyev O. A. The robot control strategy in a domain with dynamical obstacles // Lecture Notes in Computer Science, 1996. Vol. 1093, P. 211-217.
10. Герасимов В.Н. Диссертация на соискание ученой степени кандидата технических наук «Система навигации сервисного робота в среде с динамическими препятствиями», 2015.

11. Dissanayake M.W.M.G., Newman P., Clark S., Durrant-Whyte H.F., Csorba M. A. Solution to the Simultaneous Localisation and Map Building (SLAM) Problem // Australian Centre for Field Robotics Department of Mechanical and Mechatronic Engineering The University of Sydney NSW, 2006. С 1-14.

12. Московский А.Д., Метод распознавания сцен для задачи навигации мобильных роботов // II Всероссийский научно-практический семинар "Беспилотные транспортные средства с элементами искусственного интеллекта" (БТС-ИИ-2015). Труды семинара Санкт-Петербург: из-во "Политехника-сервис", 2015. С. 66-73.

13. Игнатюк В.А., Ничипоренко С.С. Использование модели сети дорог с параметрами для прокладки кратчайшего пути для алгоритма Дейкстры // Территория новых возможностей. Вестник Владивостокского государственного университета экономики и сервиса, 2009. №3. С. 154-158.

14. Герасимов В.Н., Михайлов Б.Б. Решение задачи управления движением мобильного робота при наличии динамических препятствий // Вестник МГТУ им. Н.Э. Баумана. Сер. “Приборостроение”, 2012. С. 83-92.

15. Герасимов В.Н. Система управления движением мобильного робота в среде с динамическими препятствиями // Научно-технические ведомости СПбГПУ, 2013. №5. С. 94-102.

16. Optimal and efficient path planning for partially-known environments A Stentz - Robotics and Automation, 1994. P. 3310-3317.

17. И.А. Каляев, А.Р. Гайдук, С.Г. Капустян.И.А. Модели и алгоритмы коллективного управления в группах роботов — Москва: ФИЗМАТЛИТ, 2009. — 280с.

18. И.А. Каляев, С.Г. Капустян.И.А., А.Р. Гайдук. Самоорганизующиеся распределенные системы управления группами интеллектуальных роботов, построенные на основе сетевой модели // Управление большими системами: сборник трудов. Издательство: Институт

Проблем Управления им. В. А. Трапезникова РАН. Г. Москва. 2010. №30-1. С.605-639.

19. Шаповалов И. О. Распределенная система управления движением группы крупногабаритных объектов // Известия ЮФУ. Технические науки. Издательство: Южный федеральный университет. Ростов-на-Дону. №2 (139). 2013. С. 41-46.

20. Шаповалов И. О. Применение групп мобильных роботов в сложных транспортных задачах // Известия ЮФУ. Технические науки. Издательство: Южный федеральный университет. Ростов-на-Дону. №2 (127). 2012. С 141-146.

21. Рыжова Т. П. Диссертация на соискание ученой степени кандидата технических наук по теме «Система управления коллективом мобильных роботов». 2013.

22. Назарова А. В., Рыжова Т. П. Методы и алгоритмы мультиагентного управления робототехнической системой // Инженерный журнал: наука и инновации. Издательство: Московский государственный технический университет имени Н.Э. Баумана. Москва. № 6 (6). 2012. С. 93-105.

23. Пшихопов В. Х. Групповое управление подвижными объектами в неопределенных средах // Издательство: «Физматлит». 2015. 305 с.

24. Tucker Balch, Zia Khan and Manuela Veloso. Automatically Tracking and Analyzing the Behavior of Live Insect Colonies // DOI: 10.1145/375735.376434. 2001.

Приложение

В приложении представлены ссылки на репозиторий, содержащий программное обеспечение, реализующее алгоритмы обхода препятствий на языке программирования C#:

1. Построение траектории в детерминированной среде: https://github.com/FyodorovViktor/MotionControl1/tree/master/Motion_Control_1
2. Построение траектории в недетерминированной среде: https://github.com/FyodorovViktor/MotionControl1/tree/master/Motion_Control_2
3. Реализация методов локального планирования на мобильных роботах: https://github.com/FyodorovViktor/pathplanning