



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и управление»

КАФЕДРА «Системы автоматического управления и электротехника» (ИУЗ-КФ)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
НА ТЕМУ:

Разработка модуля анализа ситуации для мобильной автономной платформы

Студент УТС.Б-81
(Группа)

_____ Корлякова Е.Ю.
(Подпись, дата) (И.О.Фамилия)

Руководитель ВКР

_____ Краснощеченко В.И.
(Подпись, дата) (И.О.Фамилия)

Консультант

_____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант

_____ _____
(Подпись, дата) (И.О.Фамилия)

Нормоконтролер

_____ _____
(Подпись, дата) (И.О.Фамилия)

2020 г.

Аннотация

Расчетно-пояснительная записка на тему «Разработка модуля анализа ситуации для мобильной автономной платформы» включает в себя 62 страницы, 38 рисунков, 10 таблиц, 13 источников и 9 приложений.

Объектом работы является модуль анализа ситуации, реализованный как программное обеспечение для автономной мобильной платформы.

Цель работы – определение и реализация оптимальных алгоритмов поиска пути в сцене.

Поставленная задача решается с помощью создания виртуальных моделей сцены для проверки работоспособности алгоритмов анализа ситуации и оптимизации системы управления автономной мобильной платформы.

СОДЕРЖАНИЕ

Введение.....	5
1 Научно-исследовательская часть	6
1.1 Постановка задачи	6
1.2 Определение источника информации о сцене	6
1.3 Создание виртуальной сцены	9
1.3.1 Моделирование виртуальной сцены	9
1.3.2 Алгоритм Q-learning	14
1.3.3. Метод Монте-Карло	14
1.4 Поиск пути в сцене	15
1.4.1 Алгоритм полного перебора	16
1.4.2 Роевой алгоритм.....	17
1.4.3 Метод потенциальных полей.....	19
2 Проектно-конструкторская часть.....	21
2.1 Структура управления автономной платформой	21
2.2 Алгоритм построения модели сцены	27
2.3 Алгоритмы поиска пути	32
2.4 Выбор среды программирования	36
2.5 Проведение испытаний	37
2.5.1 Реализация управления автономной платформой	38
2.5.2 Реализация алгоритмов построения сцены	40
2.5.3 Реализация алгоритмов поиска пути.....	46
2.6 Анализ результатов испытаний.....	49
2.7 Экономическая часть	51
2.7.1 Экономическое обоснование дипломного проекта.....	51

2.7.2 Организация и планирование работы	51
2.7.3 Определение стоимости специального оборудования.....	52
2.7.4 Расчет основной заработной платы.....	53
2.7.5 Итоговая стоимость	54
2.8 Охрана труда и окружающей среды	54
2.8.1 Характеристика условий освещения.....	54
2.8.2 Эргономика рабочего места оператора.....	55
2.8.3 Характеристика микроклимата на рабочем месте.....	57
2.8.4 Экологическая безопасность	57
Заключение	60
Список использованных источников	61
Приложение А Микроконтроллер AtMega.....	63
Приложение Б Микрокомпьютер Raspberri Pi 3 B+	65
Приложение В Алгоритм управления автономной платформой с Atmega1281	67
Приложение Г Терминал Bluetooth связи.....	83
Приложение Д Структурные схемы программ модуля	84
Приложение Е Программы модуля анализа ситуаций.....	89
Приложение Ж Процесс поиска пути при полном переборе	101
Приложение З. Процесс поиска пути при роевом алгоритме	103
Приложение И. Процесс поиска пути при методе потенциалов.....	105

Введение

В современном обществе существует тенденция к как можно большей автоматизации различных технических систем. Растет число автономных мобильных систем, действующих в одной с человеком среде. Таким модулям необходимы устройства и алгоритмы как для позиционирования в среде, так и для безопасного движения.

Примерами таких систем могут служить различные автопилоты, автоматизированные складские системы, мобильные автономные платформы для поддержки различных функций "умного дома" и т.п. Самый распространённый и доступный способ получить информацию о сцене для автономного модуля – использовать камеру. После решения основных задач обработки видеопотока, таких как сегментация изображения, распознавание и трекинг объектов, определение их положения в пространстве, т.е. создание карты глубины, и сопоставления положения камеры в пространстве при ее перемещении стоит задача поиска пути в таком пространстве.

Моделирование системы сцена-камера позволяет отказаться от многократных испытаний на реальных объектах при отработке алгоритмов управления и, при этом, добиться желаемого качества работы реализованных алгоритмов, после чего возможно их использование на автономной платформе и на основе установленного микрокомпьютера.

Актуальность работы: модуль анализа ситуации – актуальная задача для целого класса роботов, включающего в себя транспортные платформы, системы умного дома, автоматические линии производства и так далее, в том или ином виде модуль анализа ситуации необходим для любой автоматической системы.

Цель работы – создание итерационного модуля анализа ситуации, определяющего по полученным данным оптимальный вариант дальнейшего движения.

1 Научно-исследовательская часть

1.1 Постановка задачи

Автономная платформа движется в пространстве без полного знания среды, соответственно, необходимо опираться на динамически изменяемую информацию, получаемую в процессе движения по сцене. Целью движения является нахождение и достижение заданного объекта, расположенного в пространстве сцены. Аппаратная платформа накладывает ограничения на ресурсоемкость и эффективность используемых модулей, в том числе на частоту получаемой информации.

В данной работе стоит задача создания модуля анализа ситуации, позволяющего определить направления движения автономной мобильной платформы в сцене и точку для снятия новой информации с датчиков.

Исходные данные для модуля анализа ситуации:

- карта глубины;
- сегментация объектов кадра на классы;
- положение целевых объектов в кадре;
- положение автономной платформы в пространстве.

Результатом анализа ситуации являются координаты нового положения автономной мобильной платформы в пространстве сцены.

1.2 Определение источника информации о сцене

Анализ положения автономной мобильной платформы в сцене может быть реализован на основе различных датчиков и их компоновки на базе платформы. Это могут быть различные лидары, эхолоты и камеры.

Ультразвуковые датчики для транспортных роботов основаны на идее эхолокации, что позволяет с высокой точностью определять наличие препятствий в ближайших окрестностях датчика, но для измерения дальних расстояний данный тип датчиков не подходит.

Дальномеры или лидары сканируют пространство с помощью лазерных лучей и получают объемную карту пространства, однако, традиционно имеют узкий угол зрения, дороги и сложны в эксплуатации.

Оптические датчики, имеющие в основе фоторезисторы, ограничены в условиях применения, требуют определенной освещенности сцены и используются в большинстве случаев для движения по нанесенному на поверхность сцены маршруту.

Видеокамеры имеют широкий спектр возможностей, передают информацию в интуитивно простом виде, а языки программирования имеют библиотеки для работы именно с изображениями, поэтому именно камера является важнейшим источником информации для автономной платформы.

Данные с камер и обратной связи от двигателя обеспечивают автономную платформу изначальной информацией о ее положении в пространстве сцены и окружающих объектах [1].

В силу того, что получаемый камерой автономной платформы кадр рассматривается как информация об объектах сцены для различных направлений движения, возникает искривление – нет линейной зависимости между положением точки в кадре и расстоянием до соответствующего объекта сцены. На рисунке 1 пример проекции объектов на фокальную плоскость кадра.

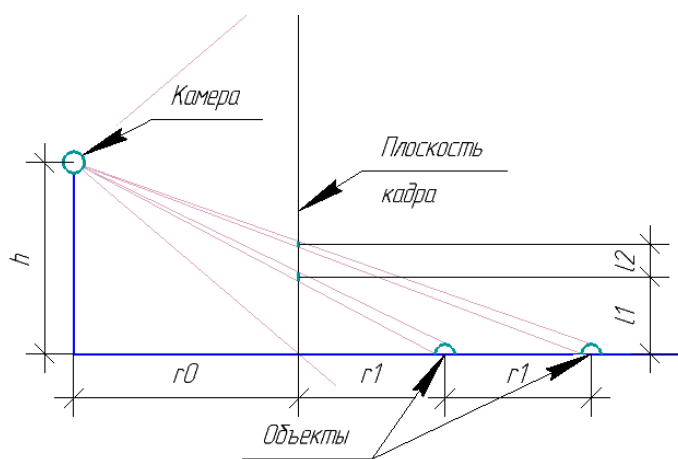


Рисунок 1 – Проекция объектов на плоскость кадра

При равном расстоянии от края выделенной области до первого объекта и расстоянии между объектами (r_1), так же, как и при равном размере объектов, их величина в кадре и соответствующие расстояния (l_1, l_2) не равны.

Определение действительного расстояния до объекта в зависимости от его положения в кадре возможно по следующей формуле:

$$R = \frac{r_0}{1-L}, \quad (1)$$

где R - расстояние до объекта на горизонтальной поверхности;

r_0 - минимальное видимое расстояние, зависящее от угла обзора камеры и высоты ее расположения над поверхностью;

L - отношение положения точки в кадре к размеру самого кадра. Стоит отметить, что эта формула верна для объектов, расположенных на горизонтальной поверхности, в таком случае, приближение к «линии горизонта», соответствующей значению $L=0.5$, достижимо при бесконечном удалении от камеры, на рисунке 2 график зависимости $L(R)$.

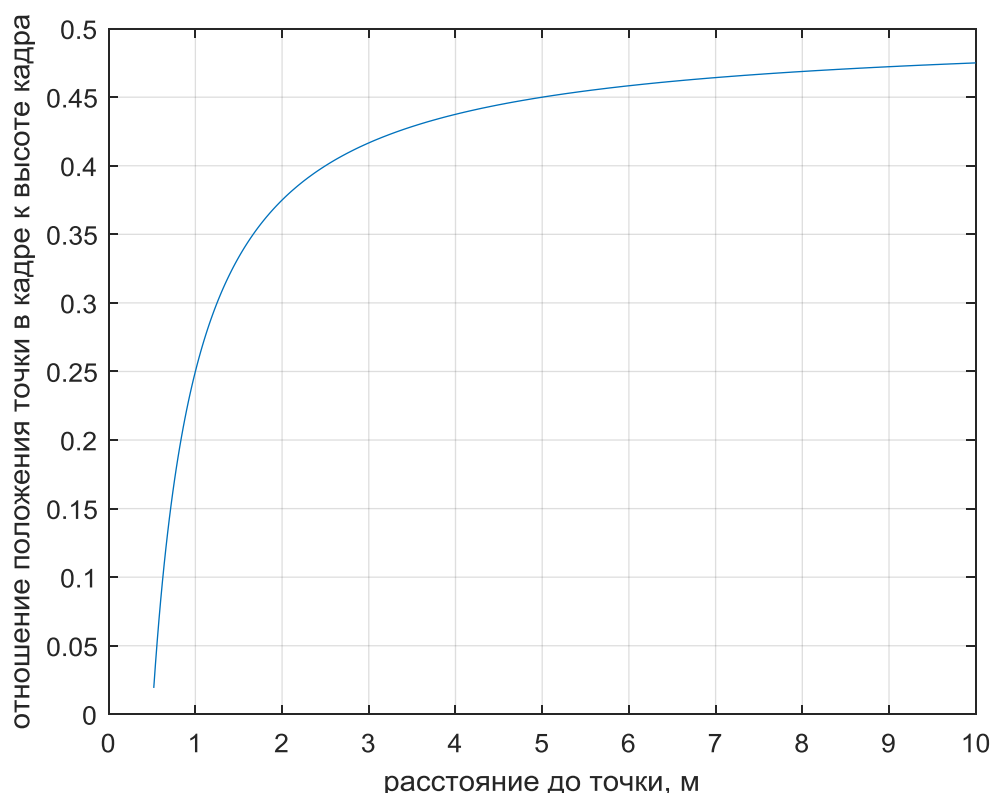


Рисунок 2 – Положение точек горизонтальной поверхности при проекции на плоскость кадра

Для объектов сложной конфигурации, расположенных в пространстве случайным образом эффективен подход с построением карты глубины [2] и сегментации изображения, что позволяет получать более или менее точную

информацию об удаленности предмета в кадре. Для последующего создания модели сцены и использования алгоритмов анализа ситуации.

1.3 Создание виртуальной сцены

1.3.1 Моделирование виртуальной сцены

Задача построения объемной виртуальной сцены важна при моделировании автоматических мобильных систем именно на этапе отладки алгоритмов анализа ситуации, определения маршрута и выделения целевых объектов, [3]. При решении поставленной задачи виртуальная сцена соотносится с положением препятствий в кадре и может рассматриваться как распределение вероятностей нахождения объектов в том или ином месте сцены.

Моделирование сложных трехмерных сцен поддерживается языками типа VRML [4], который определяет виртуальный мир через набор примитивов. Существует большое число приложений и библиотек функций для различных систем программирования, поддерживающих формат описания сцен VRML. Например, в среде MatLab для разработки сцен используют библиотека Virtual Reality Toolbox, что вместе с расширением Simulink позволяет моделировать отображение изменения систем в пространстве с течением времени. Однако библиотека доступных функций позволяет использовать только определенный набор графических примитивов [5]. В MatLab нет специальных функций для моделирования сложных поверхностей, не определенных конкретными графическими примитивами. С другой стороны, в пакете MatLab широко представлены средства создания систем технического зрения и моделирования автопилотов [6], для которых формирование виртуальной сцены позволяет оценить основные алгоритмы и принятые параметры системы технического зрения до ее аппаратной реализации. С этой точки зрения виртуальная сцена должна обладать следующими свойствами:

- легко формироваться на основе предобработанной информации или автоматически,
- обладать возможностями создавать неподвижные и подвижные объекты,

- не требовать значительных ресурсов для отображения и хранения.

Это приводит к необходимости расширить множество примитивов доступных в VRML и перейти к разработке дополнительной библиотеки функций MatLab ориентированной на задачи моделирования.

Виртуальная модель среды является результатом анализа среды на основе данных после сегментации, распознавания объектов и определения расстояния до них. При этом параметры камеры: угол обзора, функция размытия точки, размер кадра дают вероятную оценку положения препятствия в определенном месте. Есть вероятность нахождения объекта рядом с тем положением, которое видит камера автономной мобильной платформе.

При моделировании сцены это учтено как определение радиуса точности для объекта. С точки зрения компьютерных алгоритмов это распределение вероятностей нахождения препятствия, которую можно рассматривать как карту высот при поиске пути через нули, то есть, через заведомо свободные от объектов области. На рисунках 3 и 4 визуализация объектов и их возможного положения на карте и в пространстве вероятностей. При этом радиус точности изменяется в зависимости от приближения объекта к камере, для случаев на рисунках 3, 4 камера отмечена красной точкой, положение близких к ней объектов определено с большей вероятностью.

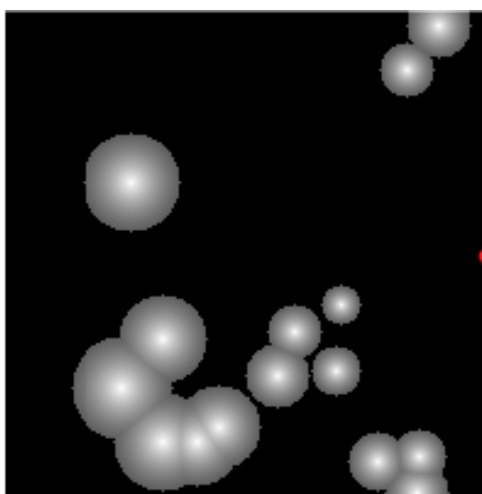


Рисунок 3 – Виртуальная модель карты

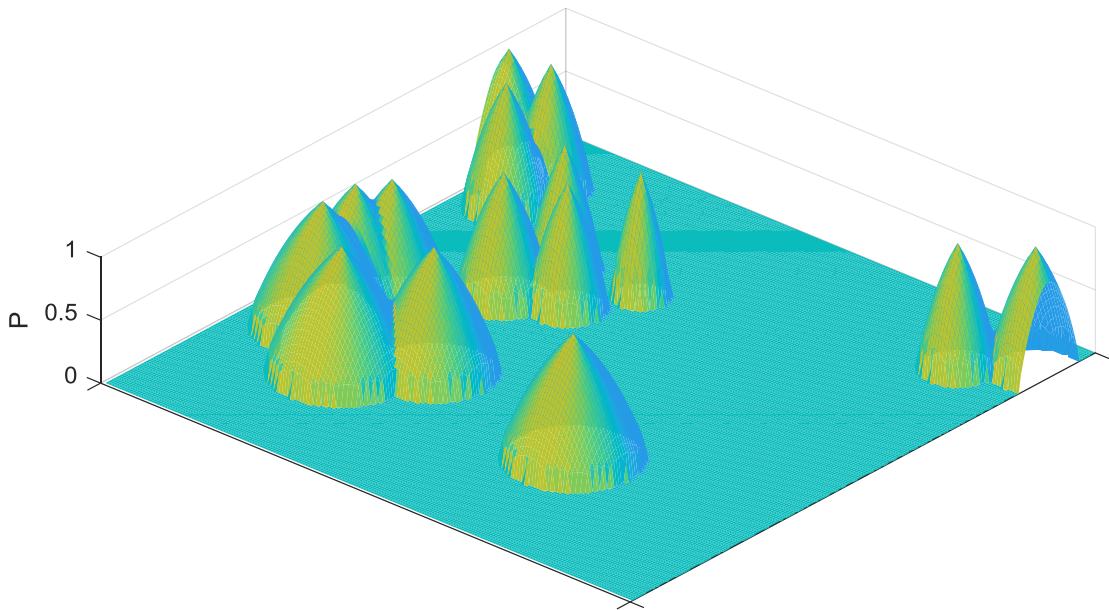


Рисунок 4 – Распределение вероятностей для виртуальной карты

В данном случае радиус точности высчитывается на основе таких параметров:

$$k_R = 1 - \frac{1}{R} - \text{относительное расстояние до объекта.}$$

$$r_T = n_T \cdot e^{-k_R}, \quad (2)$$

где r_T – радиус точности,

n_T – коэффициент радиуса точности.

При этом k_R всегда лежит в пределах $[0;1]$, а n_T , коэффициент радиуса точности, подбираемый параметр, оптимальное значение которого можно определить с помощью алгоритмов Q-learning или Монте-Карло для соответствия виртуальной модели и реальных данных.

Трассировка лучей или наблюдаемая камерой картина отличается от рисунка 3. Существует ограничение угла обзора камеры, что не позволяет определить наличие объектов вне этого угла, т.е. рассматривать области вне поля зрения, как области вероятного нахождения невидимых объектов. Так же

есть области за видимыми объектами, где, так же как и с углом зрения, нет и не может быть уверенности в отсутствии объектов, наоборот, следует рассматривать эти области, как области с высокой вероятностью нахождения в них объектов. Здесь можно провести аналогию с лучом света и тенями для точечного источника света, рисунок 5.

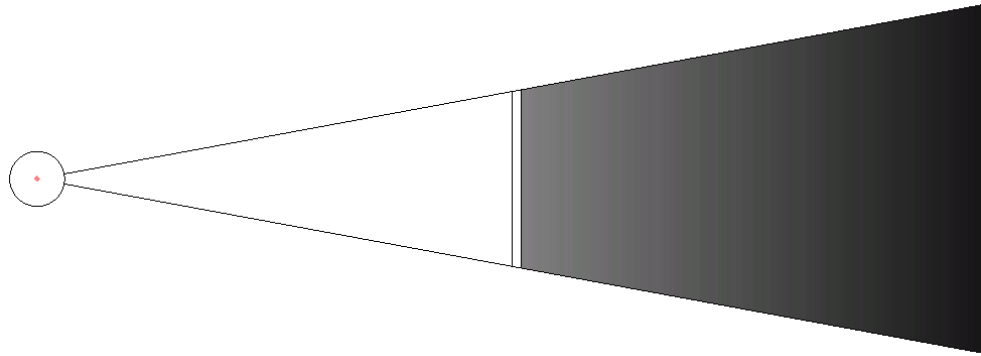


Рисунок 5 – Ход лучей от источника света к объекту и область тени

При этом, если в данной точке пространства объект расположен не с единичной вероятностью, то «тень» от точки определяется с большей вероятностью по линейному закону:

$$\Delta P = \frac{\Delta R}{m_p}, \quad (3)$$

где ΔP – разница между значением вероятности нахождения объекта в точке и в «тени»;

ΔR – расстояние между точкой объекта и точкой тени;

m_p – коэффициент вероятности тени.

При этом коэффициент вероятности тени по аналогии с коэффициентом радиуса точности может быть определен с помощью алгоритмов Q-learning или Монте-Карло для функции многих переменных, зависящей от интересующих параметров, например, m_p .

На рисункеб пример трассировки лучей и начального положения камеры.

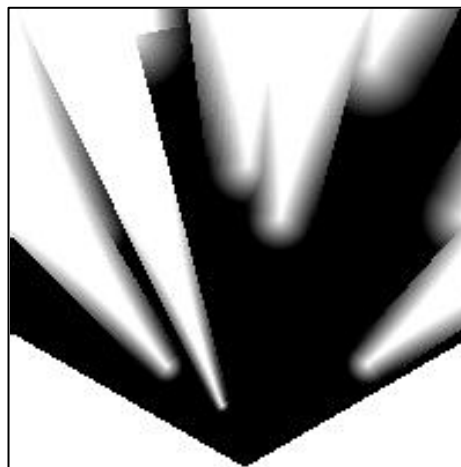


Рисунок 6 – Наблюдаемая картина виртуальной модели сцены

При изменении положения камеры и получения новой картины сцены следует соотнести с предыдущей картой, т.е. наложить два состояния друг на друга. Таким образом, увеличивается количество информации о расположении объектов в сцене и надежность этих данных. Наложение происходит путем перемножения вероятностей для двух наблюдаемых картин для определения точек с вероятным расположением объектов, после чего вероятность в этой точке определяется как максимальная вероятность из двух возможных вариантов:

$$P_i(x, y) = \max(k_{old} \cdot P_{i-1}(x, y), P_{new}(x, y)), \quad \{x, y : P_{i-1}(x, y) \cdot P_{new}(x, y) > 0\}. \quad (4)$$

Следует отметить, что доверять полученной в предыдущие итерации информации допустимо только в том случае, если объекты сцены неподвижны, тогда $k_{old} = 1$. При возможном перемещении объектов изменяется k_{old} – коэффициент забывания, т.е. вероятность для предыдущего положения учитывается с коэффициентом, меньшим единицы.

Подбор коэффициентов (2-4) может быть реализован с помощью различных методов. Например, алгоритмов Q-learning или методов Монте-Карло.

1.3.2 Алгоритм Q-learning

Q-learning – один из методов обучения с подкреплением, использующихся при работе с искусственным интеллектом. На основе получаемого вознаграждения формируется функция полезности Q , позволяющая выбирать определенную стратегию поведения или набор коэффициентов на основе полученного опыта [7].

Сам алгоритм можно представить итерационным набором шагов, это выбор данных на основе функции полезности, получение вознаграждения или штрафа, изменение функции полезности:

$$Q(s_i, a_i) = Q(s_i, a_i) + LF \cdot (r + DF \cdot \max(Q, s_{i+1}) - Q(s_i, a_i)),$$

где s - состояние системы,

a - выбранные данные,

r - вознаграждение,

LF - фактор обучения,

DF - фактор опыта.

При этом LF отвечает за степень доверия функции полезности новой информации, а DF определяет влияние предыдущих действий на следующие.

В упрощенном случае можно взять функцию такого вида:

$$Q(s_i, a_i) = (r + DF \cdot \max_{a_{i-1}} (Q(s_{i-1}, a_{i-1})))$$

1.3.3. Метод Монте-Карло

Это численный метод изучения случайных процессов, позволяющий определить вероятностные характеристики процесса, в нашем случае – модели системы. Сам метод можно разделить на такие этапы:

- Моделирование псевдослучайной последовательности с заданными корреляцией и распределением вероятностей.

- Использование этих значений в модели системы.
- Статистическая обработка результатов.

При простой структуре реализации алгоритма существует ошибка вычислений, обратно пропорциональная квадратному корню из числа рассматриваемых случаев, то есть, уменьшение ошибки в два раза требует увеличения числа испытаний не в два раза, а в четыре, поэтому этот метод чаще применяется для задач, где не нужен ответ с большой точностью.

Метод расчета значения и ошибки выводится из Центральной предельной теоремы теории вероятностей:

$$P\left(\left|\frac{\rho_N}{N} - m\right| \leq k \frac{b}{\sqrt{N}}\right) = P\left(\left|\frac{1}{N} \sum \varepsilon_i - m\right| \leq k \frac{b}{\sqrt{N}}\right) \rightarrow 2\Phi(k) - 1,$$

где ρ_N - плотность распределения;

N - число рассматриваемых случаев;

b - искомой величины

ε_i - реализации искомой величины;

m - искомое значение величины;

$\Phi(k)$ - функция распределения стандартного нормального распределения.

Для данной функции среднее значение всех реализаций будет приближенно составлять m с вероятностью $2\Phi(k) - 1$ ошибки в пределах

$$k \frac{b}{\sqrt{N}}.$$

1.4 Поиск пути в сцене

Получаемая модель сцены в виде распределения вероятностей расположения объектов информативна для стороннего наблюдателя, но сложна для восприятия автономной платформой, находящейся внутри сцены. Матричное представление вероятностей в сцене преобразуется в распределение в полярных координатах, т.е. развертку сцены с обзором в 2π радиан в плоскость относительно положения камеры. При этом для каждого выбранного

угла (направления движения) существует информация о распределении вероятностей в зависимости от расстояния до камеры. Угол и радиус определяют положение каждой точки виртуальной сцены относительно автономной платформы.

1.4.1 Алгоритм полного перебора

Как известно, решение таким способом подразумевает перебор всех возможных вариантов. В случае поиска пути это подразумевает составление длины возможного маршрута до препятствия для каждого возможного направления, учет габаритов робота при расположении вдоль выбранного пути препятствий и преимущественное направление движения, соответствующее выполнению задачи проехать максимально далеко от начальной точки, то есть нас меньше интересует вариант движения в обратную сторону. Решение задачи методом полного перебора позволяет найти наилучший из возможных путей, однако, это требует больших вычислительных ресурсов, так как необходимо рассмотреть каждое из возможных направлений и провести анализ получившейся развертки. Пример развертки для случайной карты на рисунке 7.

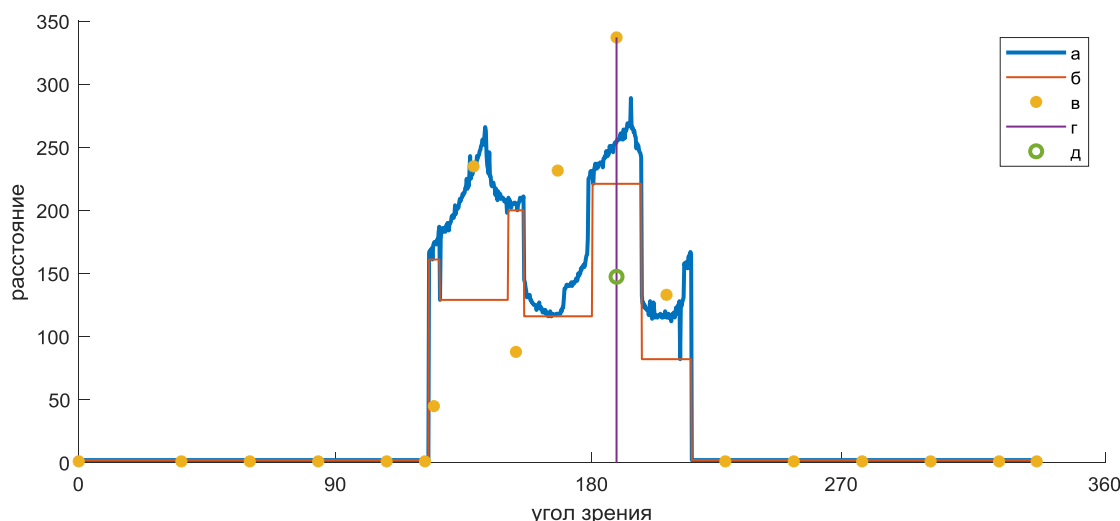


Рисунок 7 – Выбор пути полным перебором: а) расстояние до препятствия для каждого направления; б) формирование областей выбора; в) значения функции выбора для областей; г) выбранное направление; д) выбор следующей координаты для камеры.

Из рисунка 7 виден порядок выполнения алгоритма полного перебора для задачи поиска пути. Значения в точках по разным направлениям преобразуются в области движения, высчитывается оптимальная область с учетом преимущественного направления и дальности возможного пути, выбирается точка для перемещения:

$$R = \max(R(fi)) \quad fi \in \overline{0..2\pi}$$

1.4.2 Роевой алгоритм

Роевые алгоритмы основаны на имитации поведения стай, к примеру, птиц, для которых был построен первый роевой алгоритм, использующий три простых правила: во-первых, каждая частица в модели стремилась избежать столкновений с другими; во-вторых, каждая частица двигалась в том же направлении, что и находящиеся неподалеку; в-третьих, частицы стремились поддерживать одинаковое расстояние друг между другом [8]. Таким образом, роевые алгоритмы используют взаимодействие и обмен результатами для нескольких агентов.

Алгоритм колонии муравьев [9] основывается на действиях множества однотипных агентов, имеющих одну и ту же задачу, но в своем выборе основывающихся на предыдущих действиях агентов. В качестве системы вознаграждений в данном случае выступают «феромоны», добавляющие вес к принятию того или иного решения. В простейшем случае «колония» агентов путешествует по графу, увеличивая вес тех или иных ребер в том случае, если по ним чаще проходили агенты, что увеличивает вероятность решения, т.е. прокладки пути из точки А в точку Б. При этом стоит отметить, что на первых итерациях вес ребер аналогичен поиску глобального решения, тогда как при последующих итерациях происходит его уточнение.

Алгоритм искусственной пчелиной колонии – один из алгоритмов роевого интеллекта. Он основан на имитации поведения колонии пчёл, может применяться для решения дискретных и непрерывных задач глобальной оптимизации [10]. Основной принцип алгоритма, рисунок 8, основан на том,

что есть два типа «пчел» и два шага в каждой итерации. Пример на рисунке 9. Это выбор глобального направления для поиска и локализация в выбранной области.

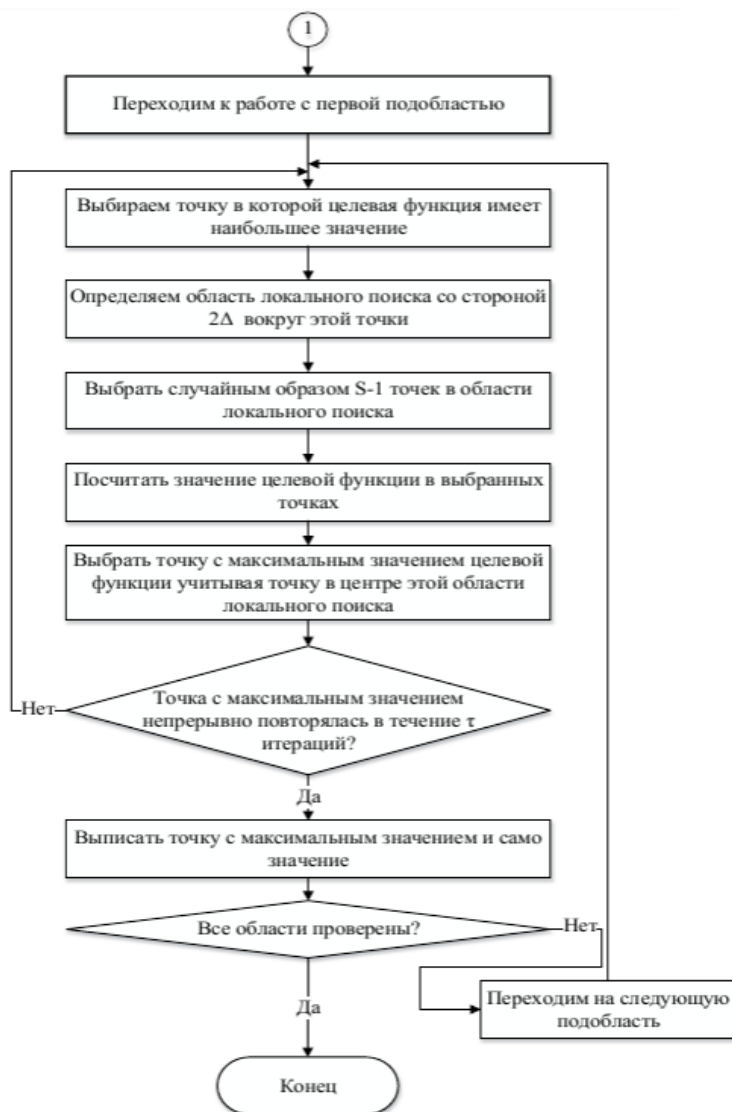


Рисунок 8 – Алгоритм искусственной пчелиной колонии

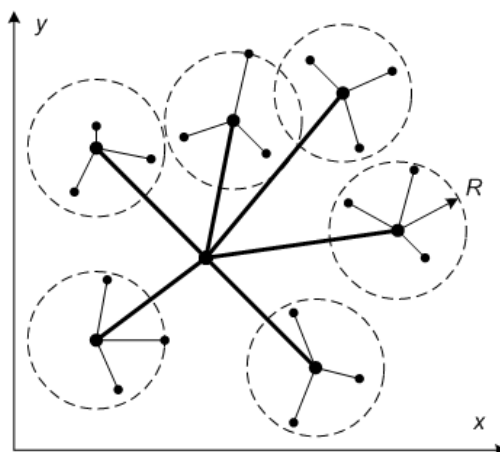


Рисунок 9 – Принцип работы алгоритма пчелиного роя

Анализ ситуации с помощью роевого алгоритма пчелинной колонии основывается на выборе нескольких случайных направлений возможного дальнейшего движения автономной платформы, «пчел-разведчиков», каждое из которых дает информацию о том, как далеко можно пройти в данном направлении, после чего для этих областей действуют «пчелы рабочие», которые аналогичным образом выбирают направления в своих областях и получают информацию о препятствиях. В следующей итерации именно лучший результат по области будет преимущественным направлением для «пчелы разведчика»:

$$\begin{aligned}
 Fi_i &= rand(0..2\pi), \quad i = \overline{1..N} \\
 fi_{ij} &= rand(Fi_i - a..Fi_i + a) \quad j = \overline{1..M} \\
 R_i &= best(R(f)) \quad f = \{Fi_i, fi_{ij}\}
 \end{aligned}$$

Для поиска решения в задаче пути роевой алгоритм позволяет сократить число вычислений, так как нет необходимости создавать полную развертку сцены, рисунок 7, а вычислять параметры только для выбранных направлений. Число итераций работы алгоритма увеличивает точность решения, но увеличивают время работы, то есть, вычислительную сложность.

1.4.3 Метод потенциальных полей

Движение в виртуальной модели с вероятностным распределением нахождения препятствий в той или иной точке можно сравнить с физическим аналогом течения воды по поверхности или, к примеру, движения частиц в электрическом поле, рисунок 10. В таком случае направление движения тела определяется значением градиента потенциального поля.

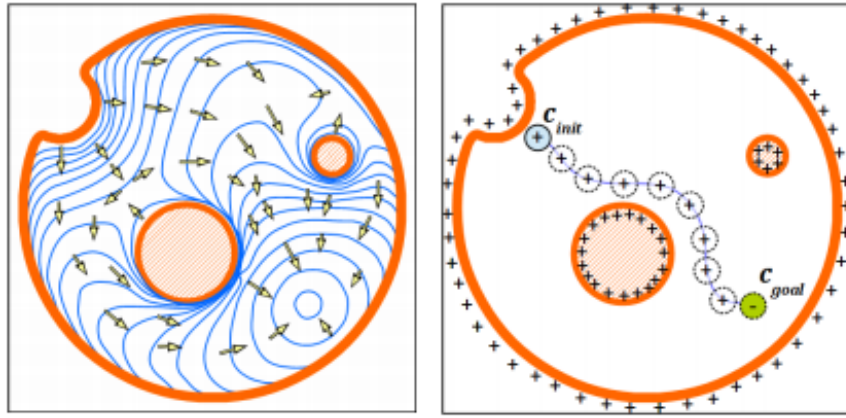


Рисунок 10 – Движение заряженной частицы в электростатическом поле

При этом движение обуславливается значениями двух функций: функции притягивающего потенциала, возрастающей при приближении к цели и функцией отталкивающего потенциала, возрастающей при приближении к препятствию[11]:

$$f = f_+ + f_-$$

$$f_+ = D \cdot d - \frac{D^2}{2},$$

где D – константа, определяющая пороговое расстояние до цели,

d – расстояние до цели в данный момент.

$$f_- = \frac{1}{2} \left(\frac{1}{d_-} - \frac{1}{D_-} \right)^2,$$

где D_- – константа, определяющая пороговое расстояние до препятствия,

d_- – расстояние до препятствия.

При этом существует опасность попадания в локальный экстремум и должна возникать сила, выталкивающая тело из этой точки. Данный метод может использоваться при решении задачи поиска пути, если рассматривать в качестве потенциала поля вероятность расположения препятствия.

На основе анализа теоретических сведений нельзя выбрать наиболее оптимальный алгоритм анализа ситуации, поэтому в дальнейшем будут рассмотрены все три варианта реализации модуля.

2 Проектно-конструкторская часть

2.1 Структура управления автономной платформой

Основным объектом для разработки алгоритмов анализа ситуации остается мобильная платформа на основе микрокомпьютера Raspberry Pi3 и микроконтроллера AtMega1281, функциональная схема представлена на рисунке 11, характеристики устройств в приложениях А, Б.

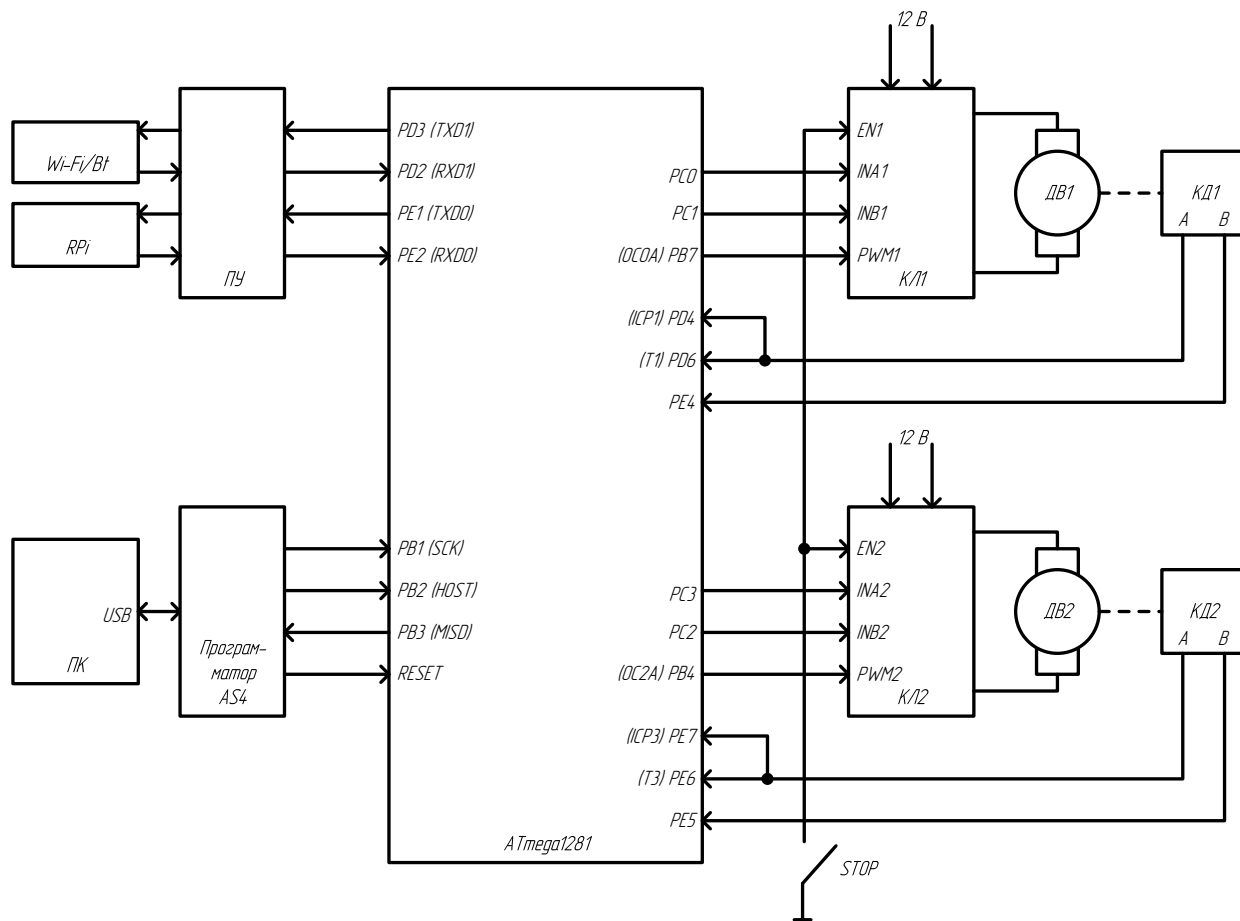


Рисунок 11 – Функциональная схема микроконтроллера автономной платформы Robot4

Для создания и тестирования модулей потребовалась реорганизация существующей системы управления двигателем автономной мобильной платформы и обеспечить связь микроконтроллера с микрокомпьютером и внешним управлением одним из возможных способов:

- по проводному каналу связи через выход USART1 микроконтроллера AtMega1281

- по беспроводному соединению с помощью модуля Bluetooth через выход USART0 микроконтроллера
- по беспроводному соединению с помощью модуля Wi-Fi через выход USART0 микроконтроллера

Движение автономной платформы на основе микроконтроллера ATmega 1281 [12] задается различными скоростями колес и направлением их движения. Рассмотрим эти моменты подробнее.

Скорость колес регулируется сигналами ШИМ с помощью двух восьмиразрядных таймеров микроконтроллера, настроенных как таймеры Fast PWM с счетчиком до 0xFF, делителем тактовой частоты и изменяемым значением регистра сравнения, регулирующего скважность сигнала.

Для данного робота в качестве начальных параметров ШИМ были выбраны следующие значения для обоих колес:

Регистр TCCR0A со значением 0x83, таблица 1.

Таблица 1 – Конфигурационный регистр TCCR0A/TCCR2A

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
1	0	0	0	0	0	1	1

Биты CS02- CS00 задают коэффициент предделителя тактовой частоты как 1/64 от тактовой частоты микроконтроллера.

Биты WGM00 и WGM01 определяют режим работы таймера-счетчика как Fast PWM.

Биты FOC0, COM01 и COM00 задают разрешение работы таймера и изменения значения регистра сравнения.

Регистр сравнения OCR0/OCR2 с начальным значением 0x4F регулирует параметры скважности сигнала, рисунок 12, что позволяет задавать и менять скорость двигателей.

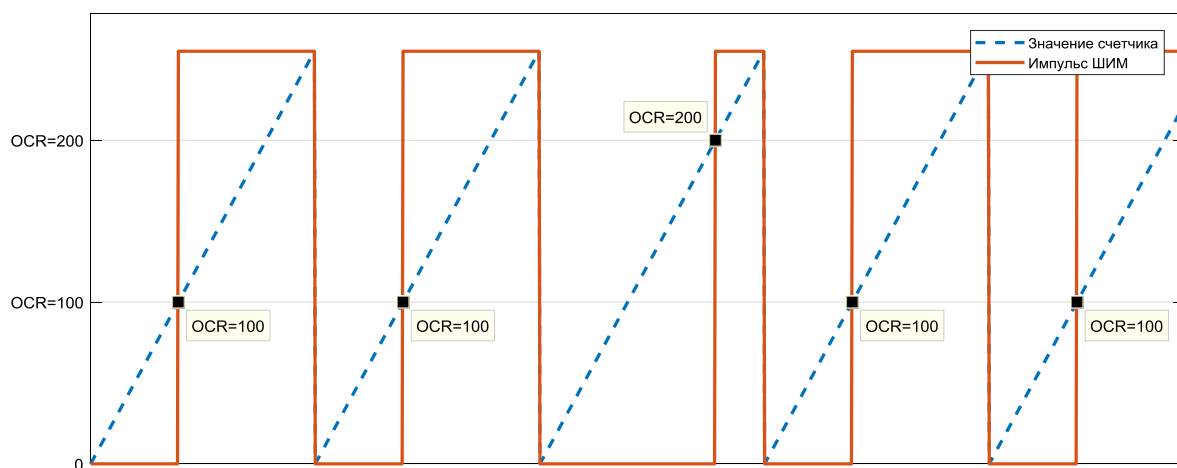


Рисунок 12 – Сигнал ШИМ для Fast PWM

Частота сигнала ШИМ зависит от частоты микроконтроллера, коэффициента делителя и разрядности таймера:

$$f_{PWM} = \frac{f_{МК}}{N \cdot (0xFF + 1)} = \frac{14.74 \cdot 10^6}{64 \cdot 256} = 900 \text{ Hz}$$

А скважность сигнала влияет на мощность работы двигателя.

Направление движения колес регулируется сигналами на ключах КЛ1 и КЛ2, рисунок 11, подачей сигналов на порты ввода-вывода C0-C3, таблица 2.

Таблица 2 – Определения направления движения

	Направление движения		Остановка	
	Вперед	Назад		
Port C0/C3	1	0	1	0
Port C1/C2	0	1	1	0

Подаваемые на двигатель команды определяются при обработке поступающих на микроконтроллер команд по прерываниям от внешних интерфейсов USART и подключенных к ним устройств, например, Raspberry. Для кодировки команд используются символы латиницы, что связано с особенностями приема и передачи данных через USART, которые обрабатываются в операторе множественного выбора, структурная схема алгоритма на рисунке 13.

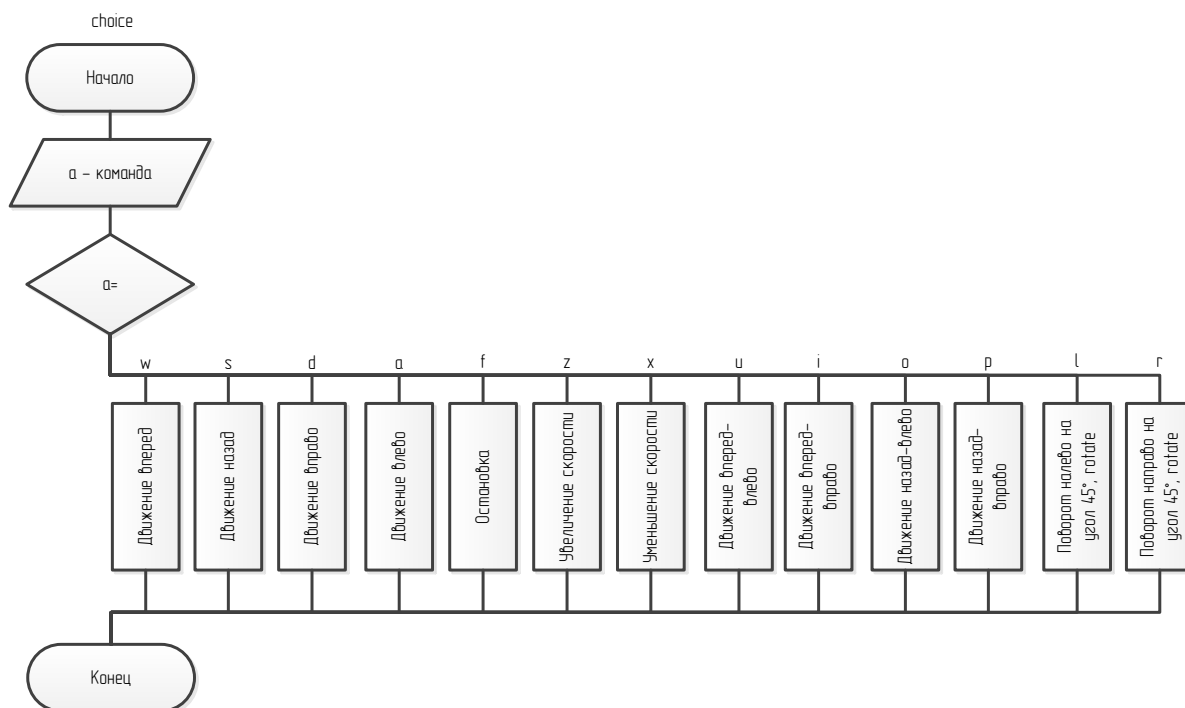


Рисунок 13 – Структурная схема алгоритма выбора направления движения

При этом команды поворота реализованы в двух видах: поворот на заданный угол в 45° вокруг неподвижного колеса и поворот вокруг центра оси колес до получения команды остановки. Второй способ обладает меньшей точностью, т.к. существует ненулевое время обработки сигналов, например, кадров видеопотока, не позволяющих устройству отправить на микроконтроллер сигнал при достижении нужной позиции. Увеличить точность можно уменьшением скорости работы колес с помощью соответствующих команд. На рисунке 14 пример поворота платформы вокруг колеса и вокруг центра оси колес.

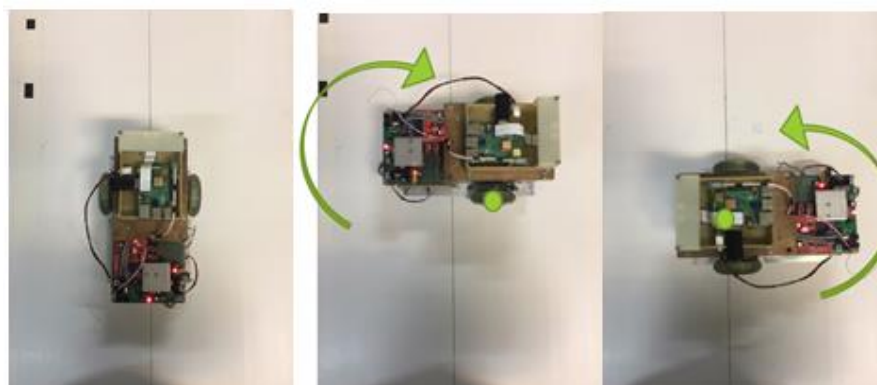


Рисунок 14 – Начальное положение робота, поворот автономной платформы вокруг неподвижного колеса и вокруг своей оси

Следует отметить, что двигатели, отвечающие за левое и правое колеса мобильной автономной платформы не идентичны. Так, при подаче на них одинаковых параметров ШИМ скорость колес различна, поэтому возникает задача стабилизации. Решение данной проблемы основано на работе шестнадцатиразрядного счетчика по режиму сброса при совпадении. В таком случае сброс таймера Timer5 и вызов прерывания осуществляется раз в 0.1с , что позволяет оперативно изменять значение ШИМ двигателя и регулировать скорость колеса. Таймер позволяет настроить прерывания на частоты вплоть до $7 \cdot 10^{-8}\text{с}$ при сбросе на 0x01, но такая скорость является неэффективной ввиду конечного времени выполнения операций программы и очень малого изменения счетчиков энкодера, на основе разности которых высчитывается необходимое изменение значения ШИМ двигателя.

Механизм стабилизации построен на основе концепции Q-learning, которая базируется на функциях штрафа и вознаграждения. Функция штрафа построена на размере расхождения скорости колес по датчикам и адаптирует коэффициент масштаба при формировании управления для каждого колеса. Это позволяет обеспечить выравнивание истинной скорости колес.

Счетчиками энкодеров являются шестнадцатиразрядные таймеры микроконтроллера Timer1 и Timer3, при этом один оборот колеса генерирует 5500 импульсов, а соотношение физических размеров колес и автономной платформы позволяет рассчитать количество импульсов для поворота на заранее заданный угол при неподвижном колесе или при движении колес в противоположные стороны. Например, 4 оборота колеса вокруг неподвижного второго колеса позволяют сделать круг в 2π .

При создании функции поворота платформы на заданный угол рассматривалась возможность перенастройки таймеров-счетчиков энкодеров на сброс при совпадении для заданного числа, однако в ходе написания алгоритма и программного кода функции оказалось, что подобный метод является переусложнением кода и ведет к нестабильной работе микроконтроллера, поэтому был создан алгоритм с доворотом вторым колесом после завершения

поворота на угол, больший заданного. На рисунке 15 структурная схема данного алгоритма.

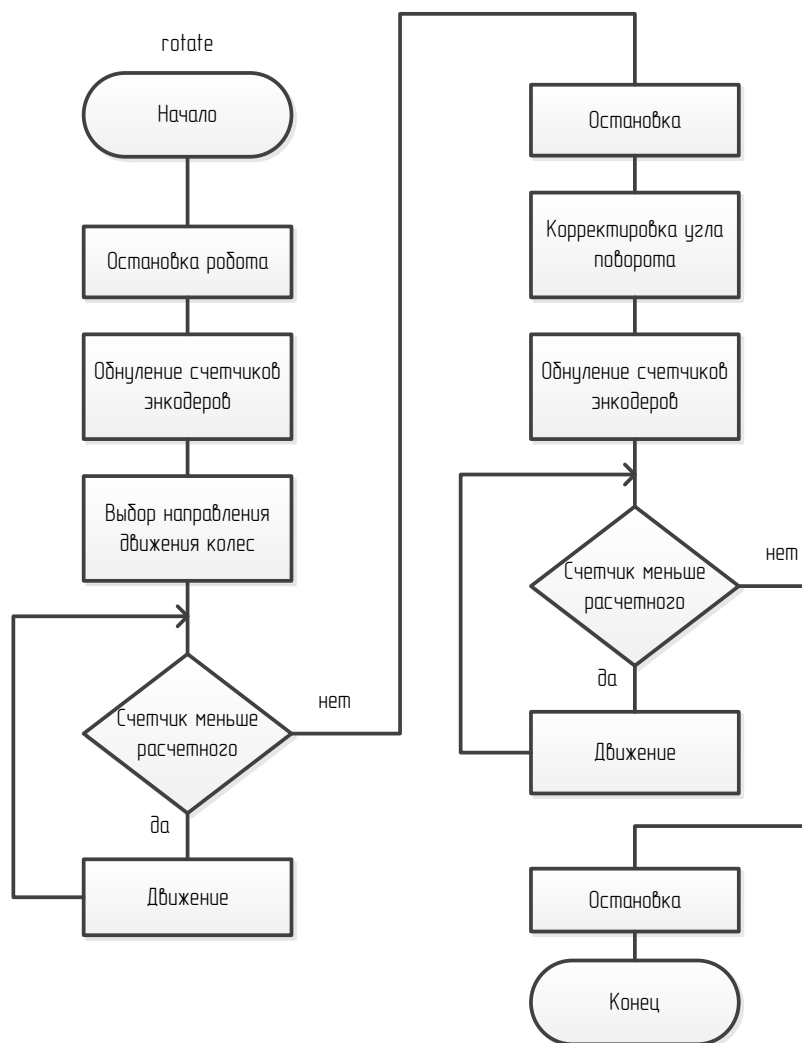


Рис. 15 – Структурная схема алгоритма поворота на заданный угол

Таким образом, после полной настройки микроконтроллера при запуске тело основной функции представляет из себя ожидание прерывания с внешних интерфейсов или собственных таймеров микроконтроллера ATmega1281. На рисунке 16 структурные схемы соответствующих алгоритмов основной функции и прерываний по USART.

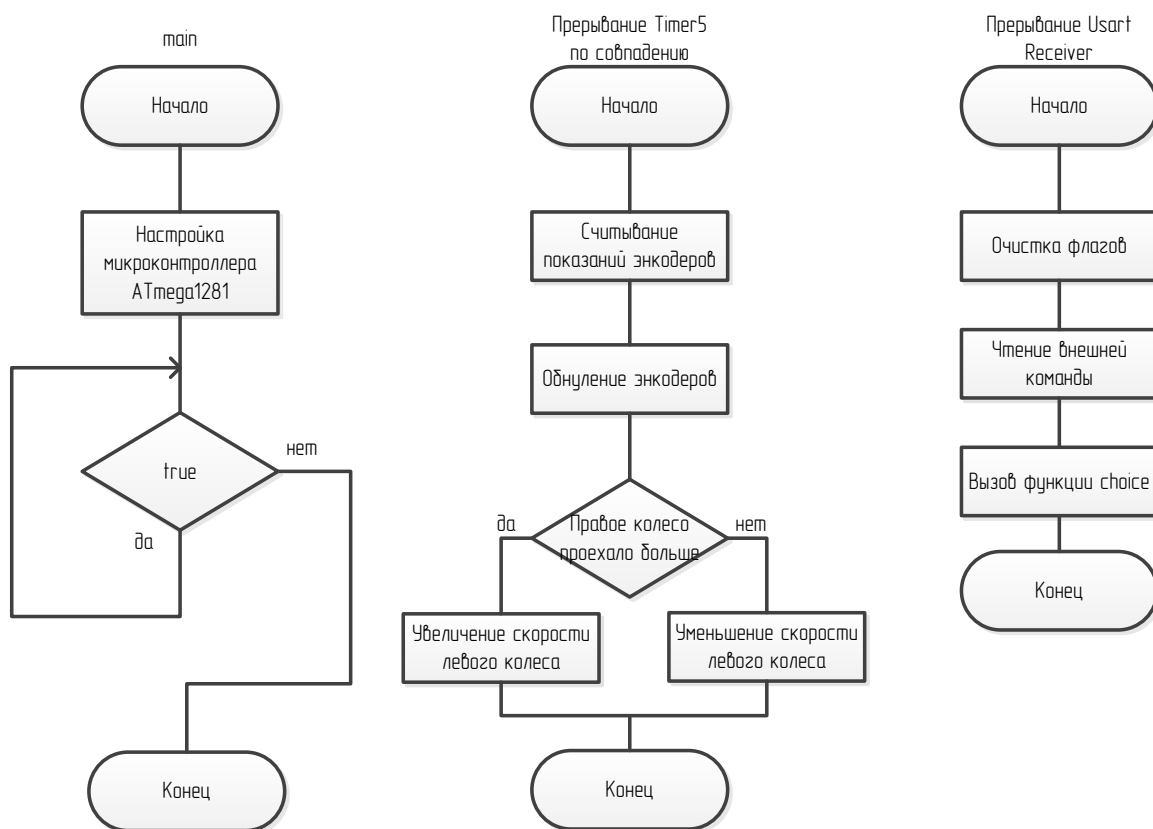


Рис. 16 – Структурные схемы алгоритмов прерываний и основная функция main

Представленные на рисунках 14-16 функции позволяют реализовать полное управление автономной платформой с помощью микроконтроллера ATmega1281 при получении команд извне: от микрокомпьютера Raspberry Pi 3+, установленного на платформе или с помощью подключенных к COM-порту модулей Bluetooth и WiFi и подключенного внешнего устройства.

Все реализованные коды для микроконтроллера ATmega 1281 приведены в приложении В.

2.2 Алгоритм построения модели сцены

Получаемая от камеры и автономной мобильной платформы информация после предобработки соответственно определяет положение робота в пространстве, [13] дает информацию о расположении объектов и стен в кадре и наличии или отсутствии в кадре цели движения. Получение такой информации основано на алгоритмах позиционирования, построения карты глубины, распознавания и сегментации, после чего эти данные используются

для построения сцены и дальнейшего применения алгоритмов выбора направления движения.

Рассмотрим алгоритм обработки и преобразования полученной информации о кадре в модель реальной сцены с распределением вероятностей объектов в ней аналогично варианту создания виртуальной сцены.

Для сегментированного изображения с тремя классами объектов (пол, препятствие, стена) определяется размер участков «пол» в кадре, на основе карты глубины и формулы (1) рассчитывается действительная длина свободного пространства перед роботом и определяется наличие объектов и/или стен сцены. Эти данные преобразуются в соответствующее положение границ объектов в модели сцены и годятся для дальнейшего определения точности камеры, то есть, создания вероятностного распределения, и наложения на информацию о предыдущих кадрах, на рисунке 17 структурная схема алгоритма создания модели сцены.

Регулирование коэффициентов n_T , m_p и k_{old} (2-4), как коэффициентов радиуса точности, вероятности тени и забывания соответственно, позволяет добиться соответствия между реальной и виртуальной сценой. При этом количество шагов до достижения цели движения и минимальное расстояние между траекторией движения и препятствием можно рассматривать как выходные параметры функции поиска пути, требующие достижения экстремума путем изменения выбранных коэффициентов. При этом число шагов должно быть наименьшим, а расстояние до препятствий – наибольшим. На данном этапе построения виртуальной сцены рассматривается вариант неподвижных объектов, поэтому k_{old} считается известной и равной 1.

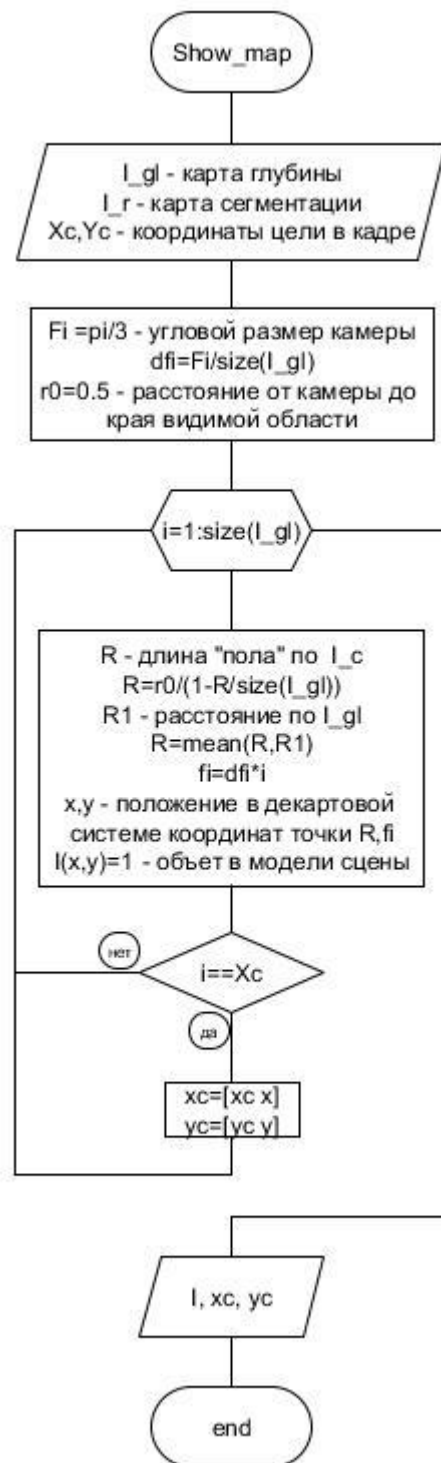


Рисунок 17 – Структурная схема построения модели сцены по кадру

Метод Монте-Карло и алгоритм Q-learning в таких условиях позволяют определить оптимальные параметры n_T и m_p , где задачу можно описать как минимизацию соотношения числа шагов к минимальному расстоянию до препятствия:

$$r = \frac{con}{r_{\min}},$$

где con и r_{\min} получены при выполнении основной программы алгоритма поиска пути при данных n_T и m_p . В таком случае для выбора параметров нас интересует комбинация коэффициентов, дающая минимальный r при использовании метода Монте-Карло и достижение оптимального повторяющегося результата для алгоритма Q-learning, где новое значение коэффициента берется как сумма предыдущего значения и случайной величины, диапазон которой зависит от r :

$$n = n + rand(1) \cdot r - \frac{r}{2}$$

$$m = m + rand(1) \cdot r - \frac{r}{2}$$

В таком случае новое значение коэффициента лежит тем ближе к предыдущему, чем лучшим решением было это предыдущее. Полученные результаты поиска коэффициентов представлены в таблице 3.

Таблица 3 – подбор коэффициентов различными методами

Метод	Коэффициент	
	n_T	m_p
Монте-Карло	26	98
Q-learning	30	105.1

Для дальнейшего тестирования алгоритма движения были взяты такие параметры: $n_T = 30$ $m_p = 100$ $k_{old} = 1$, т.е. наиболее удаленное препятствие может располагаться в области диаметром $2n_T = 60$ единиц относительно видимого положения, вероятность объекта в «тени» видимого объекта увеличивается линейно в зависимости от радиуса с коэффициентом $\frac{1}{m_p} = 0.01$, а объекты сцены не меняют своего положения с течением времени.

Информация о положении камеры и объектов в пространстве сцены последовательно преобразуется в вероятностное расположение препятствия в

области его известного положения на основе значения коэффициента радиуса точности, при этом определяется область соответствующего размера и создается шаблон вероятностного распределения для объекта, структурная схема алгоритма на рисунке 18,а). Наложение шаблонов на виртуальную модель основано не на сложении вероятностей в одной и той же точке, а на выборе максимального значения из нескольких пересекающихся шаблонов как вероятности нахождения объекта в этой точке пространства.

Определение наблюдаемой камерой картины зависит от коэффициента вероятности тени и соответствует структурная схема на рисунке 18,б), после чего полученная информация о вероятности объектов за видимым объектом аналогичным образом накладывается на известную карту распределения вероятностей с учетом коэффициента забывания, что позволяет принимать во внимание полученную ранее информацию.

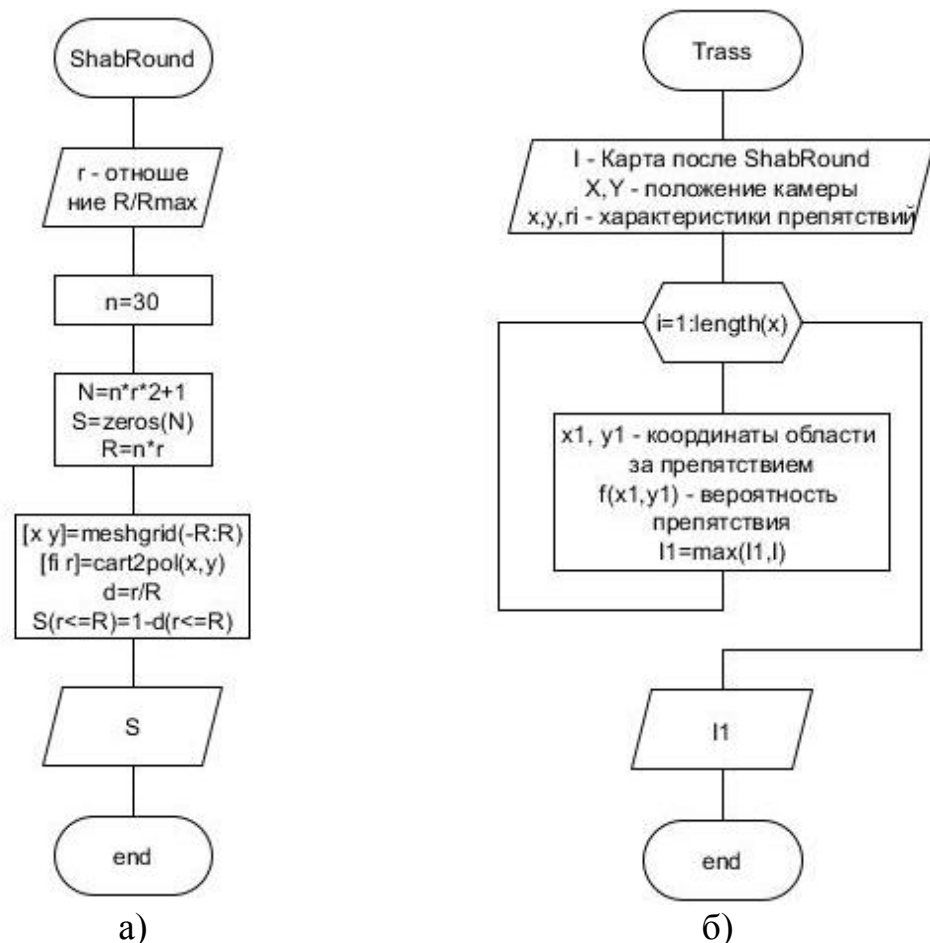


Рисунок 18 – Структурные схемы: а) расчета вероятности препятствия в радиусе точности; б) расчета наблюдаемой картины

2.3 Алгоритмы поиска пути

Определение следующего положения камеры в пространстве сцены основывается на цели движения. Выделим две задачи движения: достижение заданной цели или обход сцены для поиска цели.

Например, рассмотрим задачу достижения максимально удаленной от начального положения грани сцены или заданного объекта, расположенного на сцене. Для этого на оптимальный выбор направления оказывает влияние положение цели. При этом движение строго на цель может быть невозможно, если камеру и цель разделяет препятствие. Единственное различие в алгоритмах решения двух представленных задач заключается в изменении целевой функции:

- При наличии цели она задает направление на нее как наиболее оптимальное
- При отсутствии цели все направления являются одинаково оптимальными для дальнейшего движения.

Сам алгоритм поиска пути имеет три различные реализации.

В случае использования алгоритма полного перебора, рисунок 19,а), рассматривается вся наблюдаемая сцена, с учетом предыдущих наблюдений, и вычисляется максимальное расстояние, на которое можно пройти, для каждого направления. После чего эти значения объединяются в области и вычисляется функция оптимальности для центра области с учетом коэффициента оптимальности:

$$h_i = df\tilde{f}_i \cdot r_i \cdot c(\tilde{f}_i),$$

где h_i - функция оптимальности для области i ,

$df\tilde{f}_i$ - угловой размер области,

r_i - дальность возможного пути в этой области,

$c(\tilde{f}_i)$ - коэффициент оптимальности для направления на область.

Коэффициент оптимальности как функция от угла направления изменяется в зависимости от задачи. Если следует достичь определенной грани, то максимальное значение коэффициент должен принимать при соответствующем значении угла. Например, для достижения дальней от камеры грани, расположенной с точки зрения сцены по направлению $\frac{\pi}{2}$ можно использовать гармоническую функцию $c(fi) = \sin(fi)$, которая принимает значение 1 на $\frac{\pi}{2}$ и уменьшается при отдалении от этого угла. Таким образом направления на заданную грань становятся более предпочтительными, чем соседние.

Использование роевого алгоритма, в данном случае, алгоритма пчелиного улья, рисунок 19,б), позволяет уйти от необходимости рассматривать каждую точку сцены. Это итерационный алгоритм, каждая итерация которого опирается на два шага:

- Выбирается несколько направлений случайным образом, для которых рассчитывается возможная длина пути, это соответствует поведению ос-разведчиков
- В областях в районе этих направлений выбираются направления для «ос-рабочих», где выбирается наилучшая для каждой области точка с учетом коэффициента оптимальности $c(fi)$, аналогичного такому коэффициенту при полном переборе.
- В следующей итерации направлением «осы-разведчика» является лучший результат по области.

Алгоритм оканчивается при неухудшении максимума среди всех направлений.

Метод потенциальных полей, рисунок 19,в), преобразует распределение вероятностей в значения потенциалов в каждой точке пространства, при этом

потенциалы приобретают положительные или отрицательные значения в зависимости от того, рассматривается ли создающий их объект как препятствие или целевой объект сцены, где целевому объекту соответствуют отрицательные значения. В таком случае путь прокладывается с учетом неувеличения потенциала во время движения до следующей точки, а положение целевого объекта является точкой устойчивого равновесия для системы.

Использование алгоритмов пчелиного роя для этого метода позволяет минимизировать количество рассматриваемых областей, а выбранная точка определяется как сумма потенциалов по направлению.

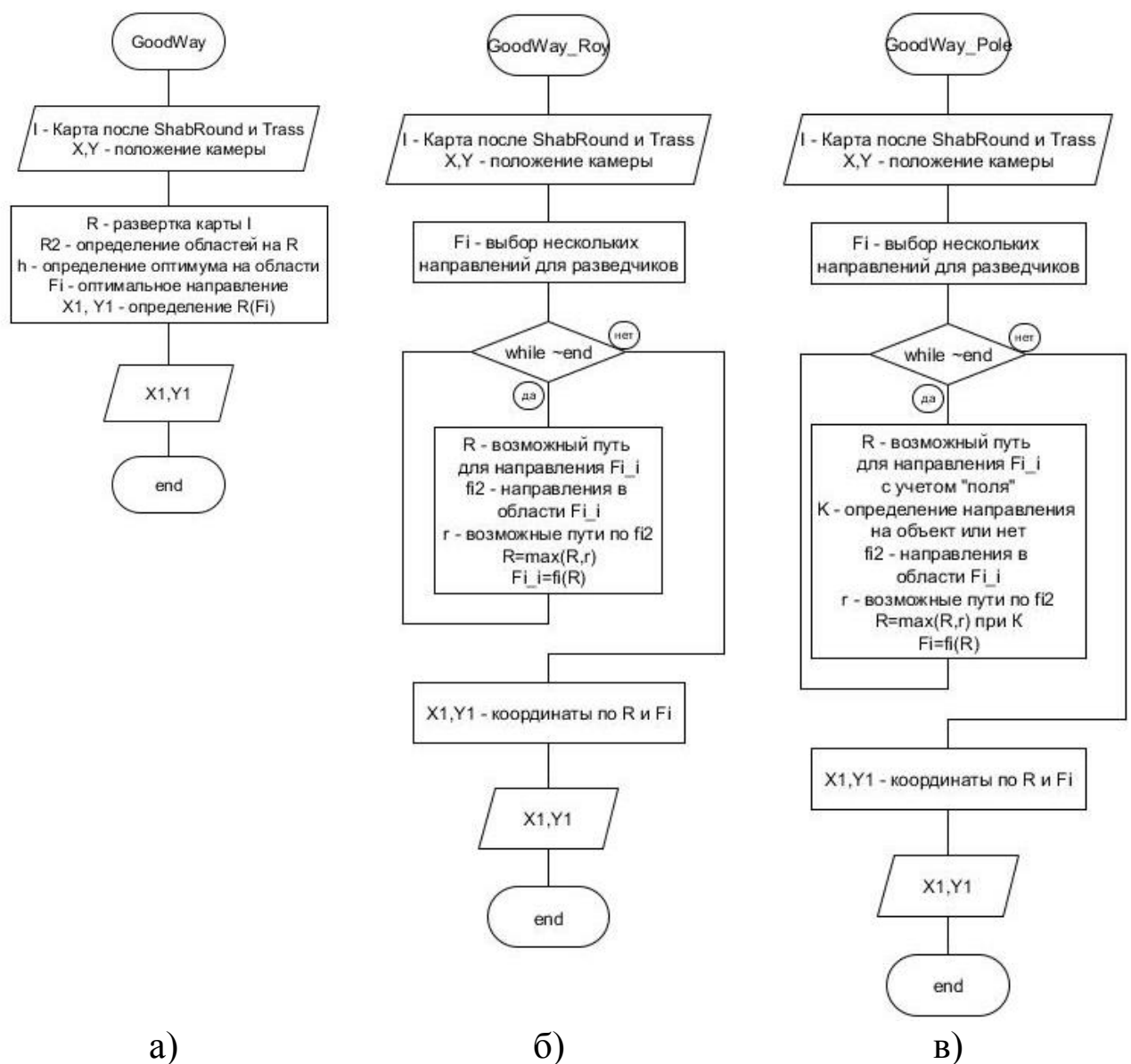


Рисунок 19 – Структурные схемы определения оптимального направления: а) методом полного перебора; б) методом роевого алгоритма; в) методом потенциальных полей с использованием роевого алгоритма

В приложении Д развернутые структурные схемы алгоритмов, в приложении Е реализованные программы.

Отметим сложность представленных выше алгоритмов, таблица 4. Алгоритм полного перебора проигрывает алгоритму пчелиного роя и методу потенциальных полей, так как требует вычисления информации о каждом возможном направлении движения автономной мобильной платформы.

Таблица 4 – Вычислительная сложность алгоритмов

Алгоритм	Вычислительная сложность
Полный перебор	$O(n^3)$
Роевой алгоритм	$O(n^2)$
Метод потенциальных полей	$O(n^2)$

Если алгоритм полного перебора и роевой алгоритм определяют направление движения на целевой объект как функцию, позволяющую выбрать из нескольких равновероятных вариантов оптимальный, то метод потенциальных полей ориентируется на само положение объекта в сцене. При этом алгоритм полного перебора и метод потенциальных полей учитывают близость препятствий и габариты автономной платформы.

2.4 Выбор среды программирования

Для реализации алгоритмов модуля анализа ситуации ключевым является выбор программного обеспечения, обеспечивающего работу с изображениями, возможность проверки работоспособности алгоритмов, получения информации из других источников и анализа полученных результатов. Формат предобработанных данных, используемых модулем, включает в себя изображения и матрицы.

Пакет MATLAB позволяет реализовывать математические модели любой сложности, при этом он гибок в плане изменения этих моделей и получения и анализа результатов и имеет набор библиотек, необходимых для удобной работы с предобработанными данными. Во многом пакет MATLAB оптимален именно для работы с матрицами и изображениями - основной получаемой модулем информации.

Этот пакет является платным, что может снизить его привлекательность при коммерческом использовании, но для научных исследований и учебы разработчики предлагают широкий спектр различных вариантов, например, MATLAB Student R2020a, предоставляющий студентам доступ к среде за минимальную стоимость.

Если MATLAB в большей степени математический пакет программирования, то Python в первую очередь язык программирования, где реализован в той или иной степени математический пакет, аналогичный MATLAB. Стоит отметить, что именно поэтому Python менее удобен для создания и проверки работы математических моделей, используемых для модуля,

Язык программирования C++, как один из основных используемых языков программирования, должен рассматриваться для реализации алгоритмов модуля. Однако в этом языке программирования математический пакет реализован еще менее полно и удобно, чем в Python, что влияет на скорость создания алгоритмов, их оптимальность и работоспособность.

Несмотря на то, что из трех рассмотренных ПО только пакет MATLAB распространяется платно, именно он является наиболее оптимальной средой для разработки модуля анализа ситуации. В дальнейшем, для интеграции различных модулей автономной мобильной платформы, оптимальные алгоритмы могут быть реализованы в Python благодаря математическому пакету и удобной работе с микрокомпьютером Raspberry Pi.

2.5 Проведение испытаний

В рамках выпускной квалификационной работы проведены эксперименты для трех задач:

1. Задача управления автономной платформой с помощью микроконтроллера AtMega;
2. Задача построения модели сцены по предобработанным данным;
3. Задача поиска пути в сцене.

Первая задача предполагает проверку работоспособности мобильной платформы и разработанного программного обеспечения микроконтроллера в задачах балансировки и движения по сцене.

Необходимое оборудование для этого эксперимента:

- сцена – стенд и объекты на нем;
- автономная мобильная платформа;
- источник управляющих сигналов – телефон с Bluetooth.

По результат эксперимента следует провести анализ работы системы.

Вторая задача основана на записи движения автономной платформы и последующей обработке кадров. Для маршрута, использованного в первом эксперименте и предобработанной информации (карта глубины, сегментация, положение робота в пространстве, определение целевого объекта в кадре) реализованный алгоритм построения модели сцены должен определить положение препятствий и создать распределение вероятностей истинного расположения препятствий для дальнейшей работы робота.

Третья задача заключается в итерационном анализе ситуации при движении в виртуальной сцене при различных алгоритмах анализа. Оценкой скорости работы служит число шагов до достижения заданной цели - дальней границы сцены или объекта на ней. Оценкой точности – обход препятствий.

Для второго и третьего экспериментов необходима вычислительная платформа (ноутбук или компьютер).

2.5.1 Реализация управления автономной платформой

В ходе отладки было отмечено наличие нежелательных помех при использовании проводного соединения микроконтроллера и микрокомпьютера, поэтому в этом эксперименте работа над автономной платформой предполагает использование беспроводного соединения.

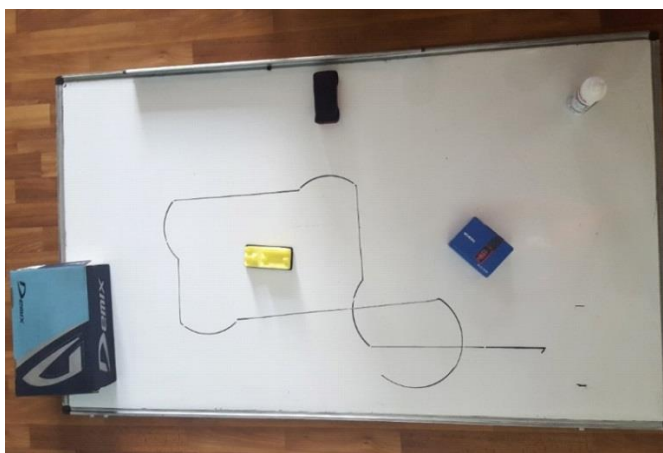
Использование модуля Bluetooth для внешнего управления оптимально, так как есть большой выбор программ для различных устройств, повторяющих работу терминала, в приложении Г характеристики выбранной программы, которая позволяет установить связь с устройством, принимать и получать данные в реализованной системе команд, основанной на буквах латинского алфавита. В таблице 4 приведены буквенные символы команд и выполняемые при этом действия. Команды можно разделить на несколько категорий, таких как движение, поворот, остановку и изменение скорости движения.

Таблица 4 – Команды управления мобильной платформой

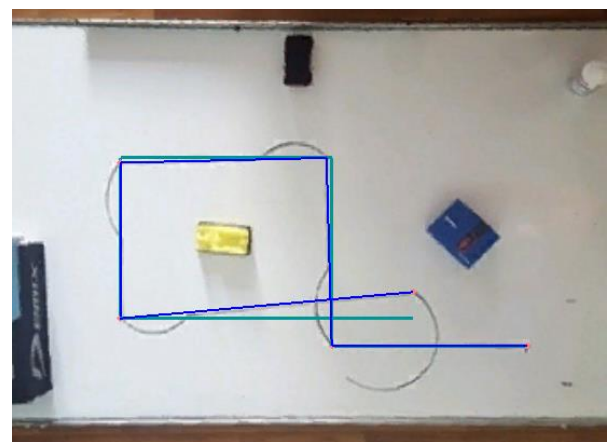
Группа	Символ	Команда
Движение	w	Движение вперед
	s	Движение назад
	u	Движение вперед и влево
	i	Движение вперед и вправо
	o	Движение назад и влево
	p	Движение назад и вправо
Группа	Символ	Команда
Остановка	f	Полная остановка

Поворот	l	Поворот налево вокруг неподвижного колеса на 45°
	r	Поворот направо вокруг неподвижного колеса на 45°
	d	Поворот вправо вокруг центра оси колес
	a	Поворот влево вокруг центра оси колес
Изменение скорости	x	Уменьшение скорости колес
	z	Увеличение скорости колес

Проверка работоспособности алгоритма балансировки скорости колес мобильного робота проводится по внешнему управлению с помощью Bluetooth, рисунок 20.



а)



б)

Рисунок 20 – Маршрут движения мобильного робота во время испытаний и сравнение с предполагаемым «идеальным» маршрутом

Необходимо провести платформу по сложной траектории (зеленая линия на рисунке 20,б) с соблюдением требуемых по заданию углов поворота и прямолинейности движения на остальных участках. Реальная траектория (синяя линия на рисунке 20,б) фиксируется с использованием маркера, который оставляет след на плоскости стенда.

Эксперименты показывают высокую скорость передачи информации, эффективную стабилизацию и выравнивание скоростей колес, высокую маневренность и управляемость системы.

В ходе серии экспериментов были определены следующие показатели движения автономной платформы:

Поворот на установленный для эксперимента угол в 90° совершается с допустимой погрешностью

Худший результат – различие между идеальным и реальным направлениями в 6° при эффекте накапливания ошибки – автономная платформа трижды поворачивала в одну сторону.

Функция стабилизации колес позволяет сохранять направление движения по прямой после первоначальной стабилизации. Соответственно, имеет смысл перед началом основной работы автономной платформы производить однократную стабилизацию скорости колес для дальнейшей корректной работы.

2.5.2 Реализация алгоритмов построения сцены

Построение модели сцены как распределения вероятностей расположения препятствий в пространстве основано на получаемой информации о положении робота и целевого объекта, если он видим в кадре, сегментации кадра на классы объектов и карты глубины кадра.

Важным параметром камеры является угол обзора. Для этого был сделан тестовый кадр на известном расстоянии от объекта при известном размере объекта, схема эксперимента на рисунке 21, результат на рисунке 22, в качестве объекта использована линейка с делениями на расстоянии 25 см от камеры, после чего определен угол обзора по формуле:

$$\varphi = 2 \arctan \left(\frac{r}{h} \right), \quad (5)$$

где φ - угол обзора;

r - линейный размер объекта;

h - расстояние до объекта.

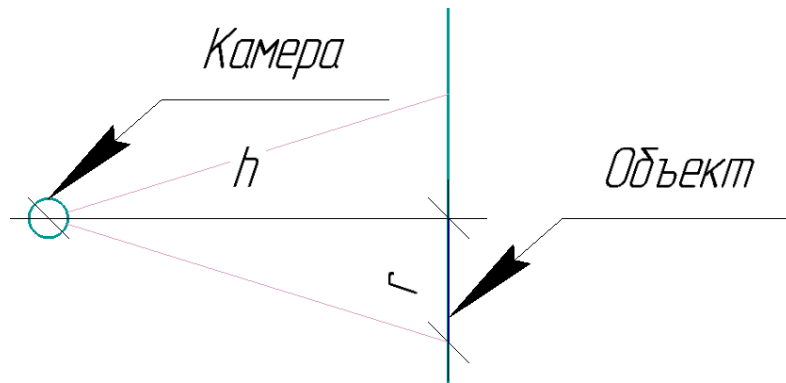


Рис. 21 – Схема определения углового размера камеры



Рисунок 22 – Определение углового размера камеры

Соответственно для формулы (5) получим:

$$r = 12 \text{ см}$$

$$h = 25 \text{ см}$$

$$\varphi = 2 \arctan\left(\frac{r}{h}\right) = 2 \arctan\left(\frac{12}{25}\right) \approx 51^\circ$$

Угловой размер кадра необходим для определения отношений расстояний между объектами в кадре.

Для анализа ситуации достаточно информации из точек маршрута, обозначенных на рисунке 23, которые предполагают остановку автономной платформы для выполнения дальнейших операций. При этом для каждого поворота автономной платформы рассматриваются два кадра, до и после выполнения маневра, так как в данном эксперименте маршрут движения робота задается внешним управлением и не в полной мере соответствует видимой

области, соответственно, необходима информация о новом направлении движения.

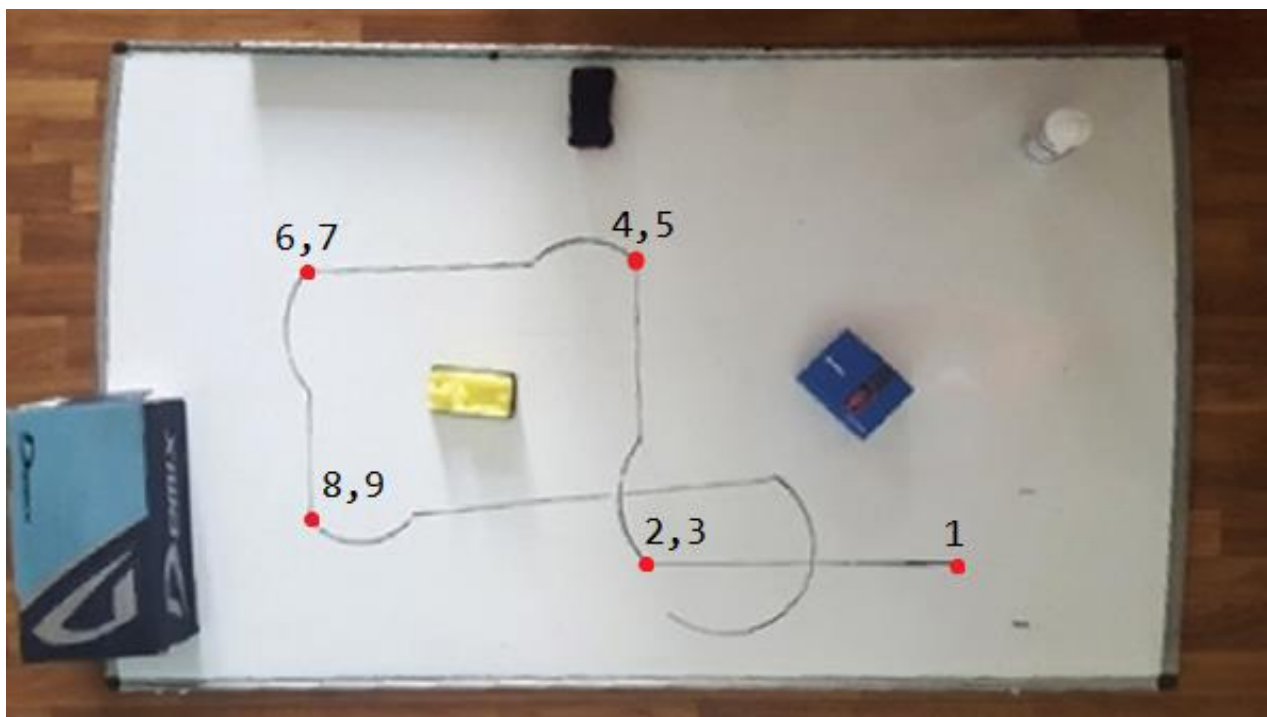
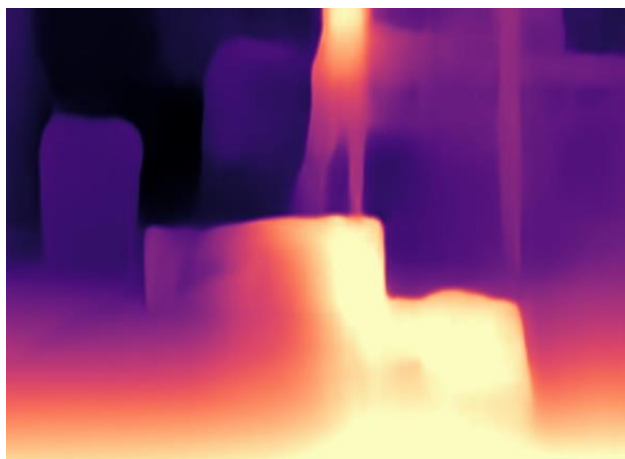


Рисунок 23 – маршрут движения мобильной автономной платформы

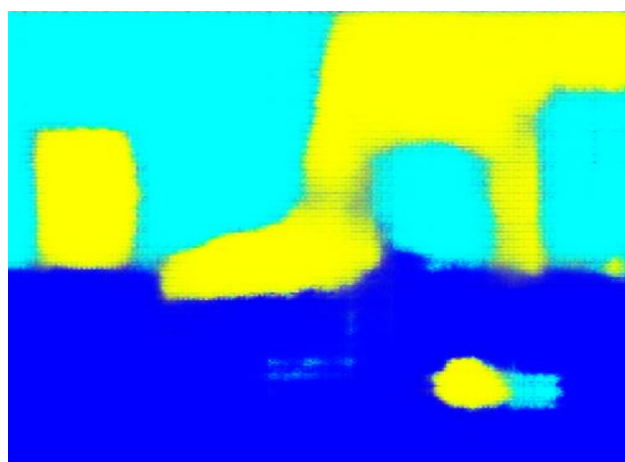
Рассмотрим информацию, получаемую после предобработки на примере первой точки первой точки, рисунок 24.



Рис. 24 – Кадр видеоряда для точки 1



а)



б)

Рис. 25 – Кадр видеоряда: а) сам кадр; б) карта глубины; в) сегментация

Для изначального кадра, рисунок 24, существует карта глубины, рисунок 25,а), и карта сегментации, рисунок 25,б). При этом в кадре нет целевых объектов, а положение в относительных координатах от начального равно, соответственно, $(0,0)$.

По полученной информации разработанный алгоритм расчета распределения вероятностей расположения объектов в сцене создает модель сцены, рисунок 26, точка 1. При этом известная информация в первой точке ограничена областью видимости, т.е. углом обзора камеры, тогда как для последующих кадров(рисунок 26, точки 2-9) происходит наложение новой информации на известную.

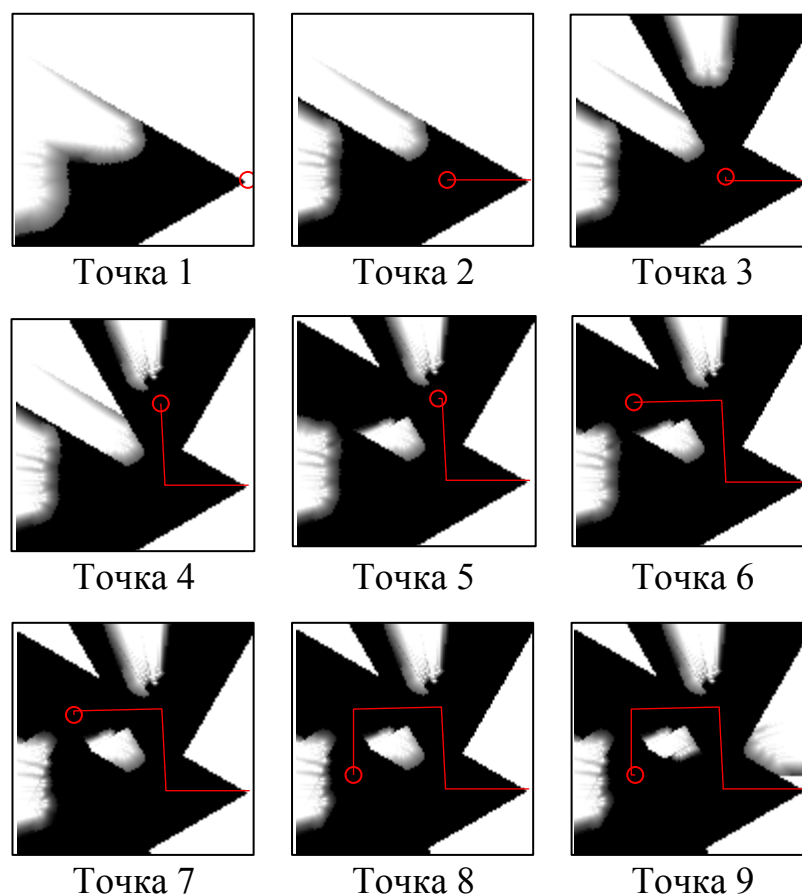


Рисунок 26 – Распределение вероятностей расположения препятствий для модели сцены

Из рисунков 23 и 26 можно сделать вывод, что алгоритм создания модели сцены корректен и отображает положение объектов в этой модели согласно их положению в сцене, что позволяет использовать модель для анализа ситуации и выбора направления дальнейшего движения.

Рассмотрим реализацию создания модели сцены для кадра с объектом. Для шестой точки (рисунки 23, 26) определим получаемую информацию: положение целевого объекта, для демонстрации выделенного на кадре желтым прямоугольником, карта глубины и карта сегментации, рисунки 27-30.



Рисунок 27 – Кадр в точке 6



Рисунок 28 – Положение целевого объекта в кадре

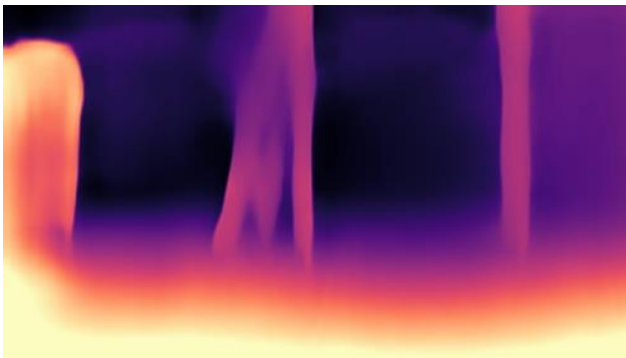


Рисунок 29 – Определение глубины кадра

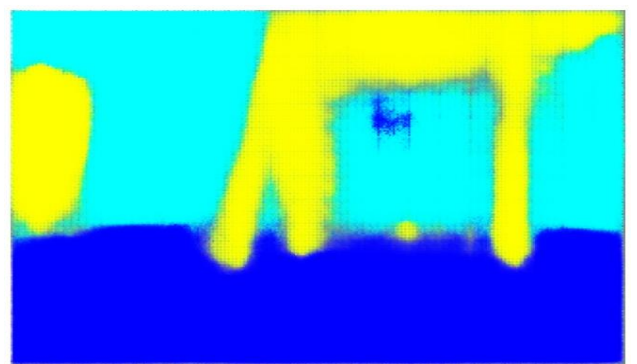


Рисунок 30 – Сегментация кадра на классы пол, объекты, стены

Выполнение алгоритма определяет предполагаемое положение точек в пространстве, после чего создает карту распределения вероятностей для этих точек, рисунок 31. Для рисунка 31 зеленым отмечен целевой объект, красным – положение камеры.

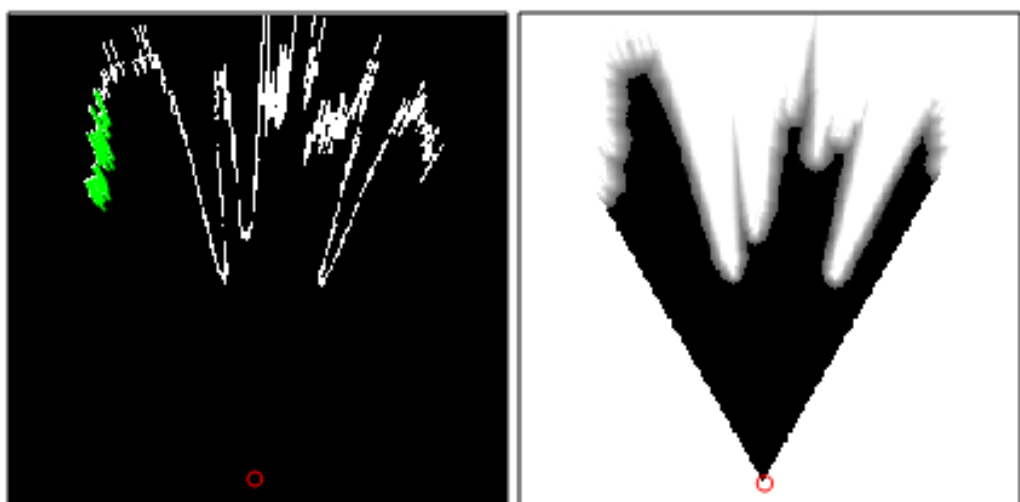


Рисунок 31 – Положение препятствий кадра и распределение вероятностей их реального положения в пространстве

Стоит отметить, что сегментация кадра на классы и карта глубины по отдельности не дают достаточно полной информации о препятствиях, при анализе ситуации нельзя полагаться только на один из источников информации, тогда как совместное использование сегментации и карты глубины по сегментированным областям позволяет построить модель сцены с вероятностным распределением препятствий, которое можно соотнести с реальным положением объектов.

2.5.3 Реализация алгоритмов поиска пути

Рассмотрим задачу прохождения в глубину карты для виртуальной сцены, на которой белыми точками обозначено расположение препятствий. Выполнением задания считается появление камеры в области заданного «финиша» – дальней от начального положения камеры стенки сцены. При размере сцены в 200 единиц областью решения считается расстояние от заданной грани, не превышающее 20 единиц. В соответствии с описанными выше алгоритмами в каждом шаге-итерации проводится моделирование области видимости камеры, с помощью функций ShabRound и Trass, после чего полученная карта сопоставляется с предыдущими результатами и функция GoodWay определяет наилучшую следующую точку. При этом считается, что новое направление камеры соответствует вектору ее движения.

Для функции полного перебора для сцены на рисунке 32, где красным отмечено начальное положение камеры, на рисунке 33 проложенный за шесть шагов путь и конечная карта с наложениями областей зрения камеры из разных точек. В приложении Ж представлены подробные карты для каждой итерации.

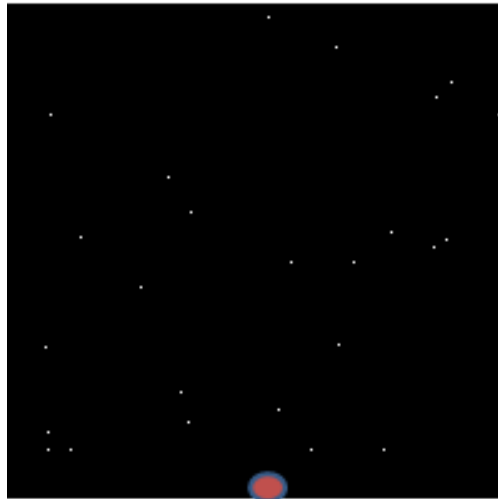


Рисунок 32 – Модель сцены с препятствиями для алгоритма полного перебора

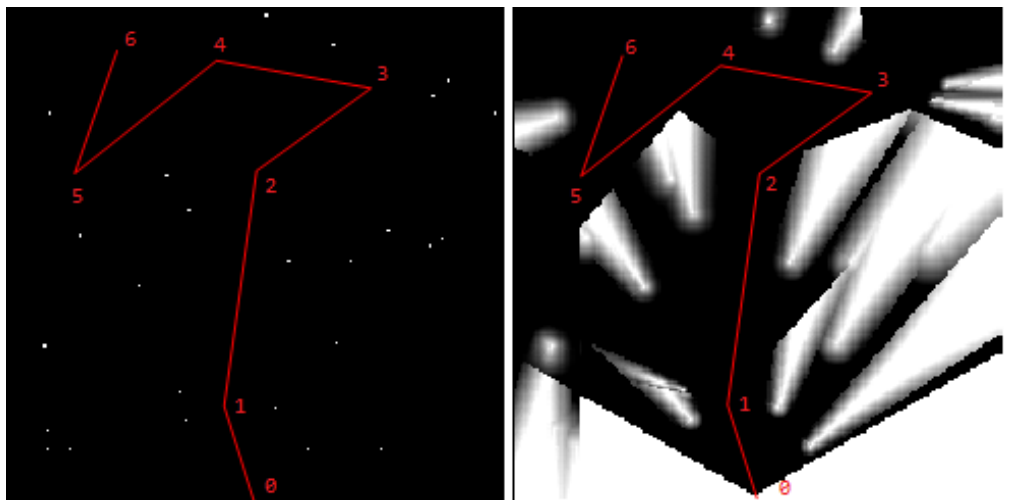


Рисунок 33 – Проложенный алгоритмом маршрут и изученные области в последней точке для алгоритма полного перебора

При этом можно отметить, что с точки зрения наблюдателя оптимальнее было бы рассматривать движение из точки 4, рисунок 33, вверх, однако для программы в этой итерации было более предпочтительно движение в область, предоставляющую в дальнейшем больше действий для маневра. Один из путей устранения таких ситуаций заключается в изменении коэффициента оптимальности $c(f_i)$ в зависимости от расстояния до цели.

Для функции GoodWay, основанной на алгоритме роя соответствующие рисунки с картой и проложенным путем – 34 и 35, а так же приложение 3 с итерациями процесса.

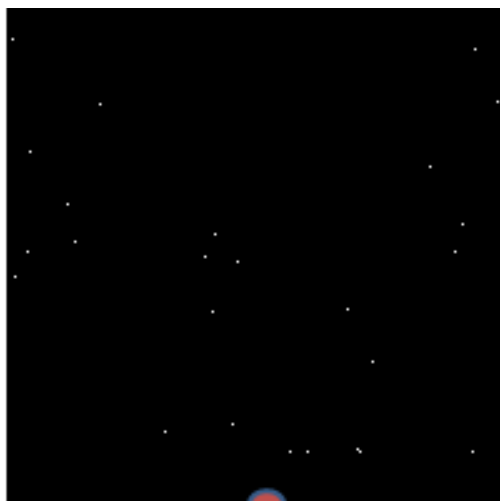


Рисунок 34 – Модель сцены с препятствиями для роевого алгоритма

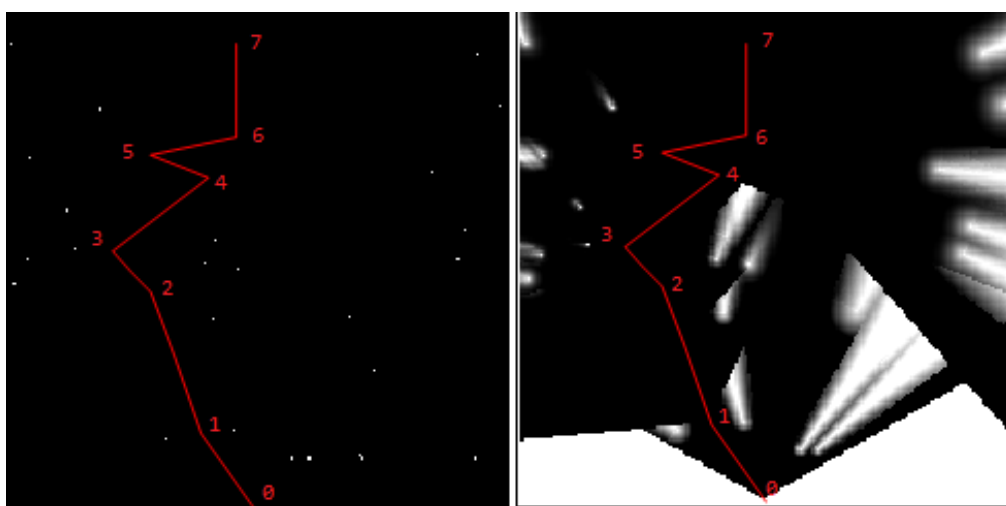


Рисунок 35 – Проложенный алгоритмом маршрут и изученные области в последней точке для роевого алгоритма

Можно отметить, что виртуальные сцены сравнимой конфигурации с равным числом препятствий оба алгоритма прошли за практически то же число шагов. При этом для алгоритма роя можно сделать то же замечание. Со стороны наблюдателя шаг 5 избыточен и не необходим, тем более, что после этого камера возвращается в точку 6.

Для функции GoodWay, основанной на методе потенциальных полей и алгоритме роя соответствующие рисунки с картой и проложенным путем – 36 и 37, а так же приложение И с итерациями процесса.

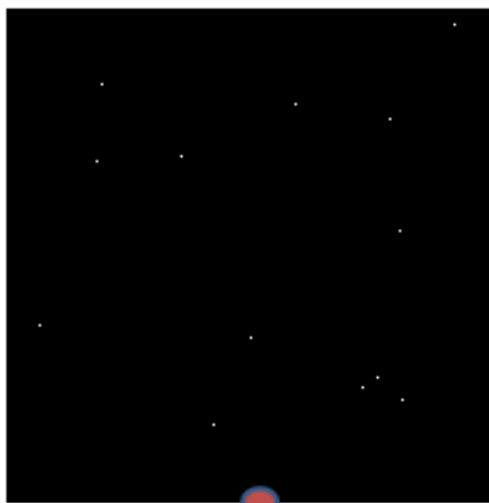


Рисунок 36 – Модель сцены с препятствиями для роевого алгоритма

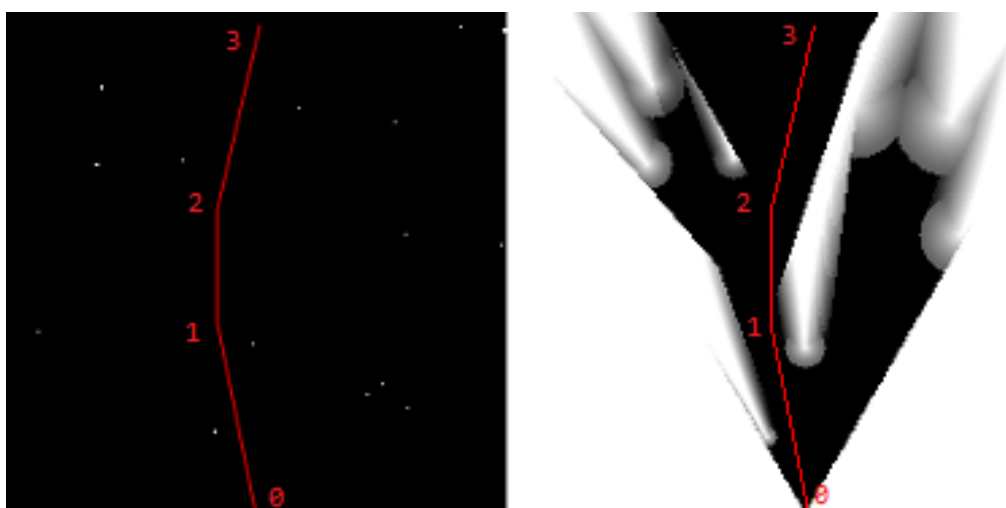


Рисунок 37 – Проложенный алгоритмом маршрут и изученные области в последней точке для метода потенциалов

Метод потенциальных полей проводит камеру по сцене сравнимой конфигурации за меньшее число шагов и более оптимальным маршрутом, чем в случае роевого алгоритма или полного перебора.

2.6 Анализ результатов испытаний

Модель сцены и виртуальная сцена обладают схожими характеристиками для использования в сравнении модулей анализа ситуации не снимаемые камерой данные, а созданные вычислительной машиной сцены, что упрощает проведение экспериментов и позволяет перенести выбранный алгоритм на работу с моделью сцены.

Рассмотрим задачи прохождения до заданной грани и обхода сцены для разного числа препятствий на сцене. Для примера взяты сцены с 5, 15 и 25 объектами и для каждого варианта сто раз запущены алгоритмы полного перебора, пчелиного роя и потенциальных полей. Усредненные результаты и дисперсия для испытаний в таблице 5.

Таблица 5 – Среднее число шагов M и дисперсия D для достижения цели

Цель:	Достижение заданной грани					
Число препятствий, шт.	Полный перебор, шагов		Роевой алгоритм, шагов		Метод потенциальных полей, шагов	
	M	D	M	D	M	D
25	6.8	3	7.5	3.2	5.9	2.3
15	4.4	2.6	5.7	2.5	3.8	1.4
5	3.2	2.7	2.9	2.5	2.2	1.2
Цель:	Обход сцены					
Число препятствий, шт.	Полный перебор, шагов		Роевой алгоритм, шагов		Метод потенциальных полей, шагов	
	M	D	M	D	M	D
25	16.1	6.2	17.4	6.3	13.7	4.7
15	15.9	5.8	13.3	6.1	12.4	3.8
5	12.5	5.3	10.4	5.9	10.2	3.9

Из таблицы 5 можно сделать вывод о том, что в целом роевой алгоритм и алгоритм полного перебора показывают сопоставимую сложность и продолжительность полученного пути, а метод потенциальных полей прокладывает маршрут за меньшее число шагов, т.е. более оптимально. При этом вычислительная сложность роевого алгоритма и метода потенциальных полей ниже, так как не предусматривает определение расстояний по каждому направлению.

В целом метод потенциальных полей показал себя лучшим из всех рассматриваемых алгоритмов по параметрам скорости достижения цели и вычислительной сложности алгоритмов.

2.7 Экономическая часть

2.7.1 Экономическое обоснование дипломного проекта

В дипломном проекте проводится научно-исследовательская работа по разработке модели системы поиска пути на основе анализа положения автономной платформы относительно объектов сцены. Выполним расчеты затрат на выполнение научно-исследовательской работы (НИР), в том числе, на изучение теоретических положений, разработку и реализацию алгоритмов поиска пути в модели сцены, построенной на основе входных данных автономной платформы или при реализации виртуальной сцены.

Научно-исследовательская работа проведена одним инженером-программистом. Возникающие при выполнении научно-исследовательской работы затраты можно выделить в следующие пункты:

1. затраты по основной заработной плате;
2. затраты по дополнительной заработной плате;
3. затраты по отчислениям на социальное страхование;
4. расходы на научные и производственные командировки; оплата работ, выполняемых сторонними организациями; прочие расходы.

При выполнении работы можно выделить следующие этапы:

1. разработка технического задания;
2. изучение темы;
3. научно-теоретическое исследование;
4. научно-практическое исследование (написание программы);
5. оформление работы.

2.7.2 Организация и планирование работы

Все этапы работы можно разделить на три части:

1. Исследовательская часть (теоретическая база);
2. Конструкторская часть (разработка алгоритмов, написание программ);
3. Настройка, выпуск сопроводительной документации.

Для оптимального планирования работы разрабатывается календарный график, который показывает время начала каждой работы относительно начала проекта и её продолжительность.

Для каждого конкретного вида работы определяется трудоемкость и исполнители. В рамках научно-исследовательской работы все работы выполняются из расчета возможностей одного инженера-программиста, план работы в таблице 6.

Таблица 6 – Планирование работы по теме НИР

№	Перечень работ	Исполнители	Трудоёмкость работы (дни)
1	Разработка ТЗ	Инженер-программист	1
2	Изучение темы	Инженер-программист	3
3	Научно-теоретическое исследование	Инженер-программист	5
4	Научно-практическое исследование	Инженер-программист	5
5	Разработка документации	Инженер-программист	5
Итого дней:			19

2.7.3 Определение стоимости специального оборудования

В расчет стоимости материалов и покупных изделий. входит стоимость материалов, покупных изделий, комплектующих и других материальных ценностей, расходуемых непосредственно в процессе выполнения работ.

Для данного проекта используется: аппаратно-вычислительная платформа (компьютер) и среда программирования MATLAB Student R2020a. При этом в затратах не учитывается аппаратная платформа мобильного автономного робота и камера, так как для данной научно-исследовательской работы необходимые данные с камеры поступают после предобработки. Смета представлена в таблице 7.

Таблица 7 – Смета на приобретение оборудования и ПО

№	Наименование изделий	Количество, шт	Стоимость, руб.	Общая стоимость руб.
1	Аппаратно-вычислительная платформа	1	29500	29500
2	MATLAB Student R2020a	1	10500	10500
Итого стоимость оборудования:				40 000

2.7.4 Расчет основной заработной платы

1. Основная заработная плата исполнителей (инженера-программиста) рассчитывается исходя из числа исполнителей, оклада и времени на выполнения работ.

Для среднего месячного оклада, рассчитанного на 21 день, вычисляется заработная плата исполнителя, таблица 8.

Таблица 8 – Заработная плата исполнителя

№	Исполнители	Месячный оклад, руб.	Оклад, руб./день	Количество отработанных дней	Выплаты, руб.
1	Инженер-программист	30 000	1400	19	27200
Итого выплат по основной зарплате:					27200

Выплаты по основной заработной плате составляют 27 200 руб.

2. Дополнительная заработная плата по данному виду работ не предусматривается.

3. На 2020 год общая сумма страховых платежей составляет 30 процентов от фонда оплаты труда. Из них 22 процента - обязательный пенсионный взнос. В Фонд социального страхования идут 2,9 процентов, в фонд обязательного медицинского страхования – 5,1 процента. Расчет начислений в таблице 9.

Таблица 9 – Начисления на заработную плату

№	Наименование начисления на заработную плату - сумма страховых платежей	Процент	Начисления, руб.
1	Пенсионный фонд	22%	5984
2	ФСС - Фонд социального страхования	2,9%	788.80
3	ФОМС - Фонд обязательного медицинского страхования	5,1%	1387.20
Итого начислений:			8160

4. Расходы на научные и производственные командировки; оплата работ, выполняемых сторонними организациями; прочие расходы для данной работы не предусматриваются.

Таким образом общая сумма затрат на разработку алгоритмов анализа ситуации рассматривается как калькуляция статей расходов, таблица 10.

Таблица 10 – Калькуляция статей расходов

№	Наименование статей расхода	Затраты, руб.
1	Специальное оборудование и ПО	40 000
2	Основная зарплата	27200
3	Дополнительная зарплата	-
4	Сумма страховых платежей	8160
5	Расходы на научные и производственные командировки	-
6	Оплата работ, выполняемых сторонними организациями	-
7	Прочие затраты	-
Итого затрат:		75360

2.7.5 Итоговая стоимость

Данный раздел выпускной квалификационной работы посвящен калькуляции статей расхода на реализацию проекта, в том числе на расчет стоимости оборудования, заработной платы инженера-программиста и соответствующих страховых начислений.

Разработка проекта занимает 19 дней, из них по 5 дней на теоретическое и практическое исследования.

Итоговая стоимость разработки модуля анализа ситуации для автономной платформы составляет 75 360 рублей.

2.8 Охрана труда и окружающей среды

2.8.1 Характеристика условий освещения

Рабочее место исполнителя должно отвечать требованиям по освещенности, эргономике и микроклимату.

Для рабочего кабинета исполнителя определены следующие параметры::

1. Площадь помещения – 5 кв. м

в) зона легкой досягаемости ладони; г) оптимальное пространство для грубой ручной работы; д) оптимальное пространство для тонкой ручной работы

Рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места изображена на рисунке 38, она предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

Рассмотрим оптимальное размещение компьютера, предметов труда и документации на рабочем столе:

1. Дисплей размещается в зоне а (в центре);
2. Клавиатура — в зоне г/д;
3. Системный блок размещается в зоне б (слева);
4. Принтер находится в зоне а (справа);
5. Документация:
 - в зоне легкой досягаемости ладони — в (слева) — литература и документация, необходимая при работе;
 - в выдвижных ящиках стола — литература, неиспользуемая постоянно.

Так же при проектировании письменного стола следует учитывать следующее:

- высота стола должна быть выбрана с учетом возможности сидеть свободно, в удобной позе, при необходимости опираясь на подлокотники;
- нижняя часть стола должна быть сконструирована так, чтобы программист мог удобно сидеть, не был вынужден поджимать ноги;
- поверхность стола должна обладать свойствами, исключающими появление бликов в поле зрения программиста;
- конструкция стола должна предусматривать наличие выдвижных ящиков (не менее 3 для хранения документации, листингов, канцелярских принадлежностей, личных вещей).

Важным элементом рабочего места программиста является кресло. Оно выполняется в соответствии с ГОСТ 21.889-76.

Конструкция рабочего стола и кресла соответствует условиям эргономики.

2.8.3 Характеристика микроклимата на рабочем месте

Следующие показатели характеризуют микроклимат рабочей зоны и должны соответствовать требованиям ГОСТ 12.1.005-88:

- температура воздуха;
- относительная влажность воздуха;
- скорость движения воздуха.

Значения параметров микроклимата:

- *в холодный период*: оптимальная температура воздуха 18-21 °С, допустимая температура: 17-21 °С;
- *в теплый период*: оптимальная температура воздуха 20-25 °С, допустимая температура: 13-28 °С;
- оптимальная относительная влажность воздуха месте в холодный и теплый период: 40-60%, допустимая: до 75%.
- скорость движения воздуха на рабочем месте в холодный и теплый период: 0,1м/с.

Данные параметры поддерживаются в кабинетах, используемых для работы с автономной мобильной платформой, соответственно, нормы по микроклимату выполняются.

2.8.4 Экологическая безопасность

Наиболее опасным фактором, воздействующим на окружающую среду при разработке данного проекта, является наличие люминесцентных ламп в помещении. Эти лампы, при неправильной утилизации оказывают вредное воздействие на окружающую среду. Чтобы снизить вред, оказываемый на окружающую среду, лампы утилизируют.

Ртутные лампы и люминесцентные ртутьсодержащие трубки (далее – ртутьсодержащие лампы) представляют собой газоразрядные источники света, принцип действия которых заключается в следующем: под воздействием электрического поля в парах ртути, закачанной в герметическую стеклянную трубку, возникает электрический разряд, сопровождающийся ультрафиолетовым излучением. Нанесенный на внутреннюю поверхность люминофор преобразует ультрафиолетовое излучение в видимый свет.

Люминесцентные лампы относят к отходам 1 класса опасности, которые являются чрезвычайно опасными отходами. Степень вредного влияния отходов подобного класса опасности на окружающую среду очень высокая. Когда на окружающую среду воздействуют отходы первого класса опасности, то, во-первых она необратимо нарушается, а во-вторых период её восстановления отсутствует.

Одним из наиболее опасных факторов, загрязняющих окружающую среду, является ртуть и её пары. Органолептические свойства паров ртути таковы, что пораженный не замечает, как вдыхает их. При этом поражается центральная нервная и мочевыделительная системы. Ртуть, попавшая в окружающую среду, препятствует нормальному функционированию организма. Длительное нахождение ртути в условиях экосистемы приводит к её деградации.

Правила утилизации и правила работы со ртутью соответствуют нормам: СанПиН 2.1.7.1322-03 «Гигиенические требования к размещению и обезвреживанию отходов производства и потребления»;

СП 4607-88 «Санитарные правила при работе со ртутью, ее соединениями и приборами с ртутным заполнением» (утв. Главным государственным санитарным врачом СССР 04.04.1988).

Для освещения помещения рабочей зоны используются люминесцентные лампы типа ЛПО 18 в количестве 2 шт.

Согласно методическим рекомендациям «Справочные материалы по удельным показателям образования важнейших видов отходов производства потребления» 1997 г.

Количество отработанных ламп ЛПО 18 вычислим по формуле:

$$Q_{p.л.} = \frac{nTK}{N_{cc}}$$

где n – количество установленных ламп;

T – время работы лампы в сутках;

K – количество рабочих дней в году;

N_{cc} – норматив срока службы одной люминесцентной лампы.

$$Q_{p.л.} = \frac{2 \cdot 8 \cdot 252}{15000} = 0.25 \approx 1 шт.$$

Вес отработанной лампы составит 190 г.

Помещение для хранения отделено от производственных и жилых помещений, а также защищено от воздействия химических агрессивных веществ, различных атмосферных осадков, влажности и воды, которая может поступать по поверхности или же из-под грунта.

Отработанные люминесцентные лампы складируют в специальной таре в отдельном помещении, закрытом от доступа посторонних лиц до сдачи их на утилизацию, в таком случае обеспечивается охрана окружающей среды и минимизируется влияние опасных и вредоносных элементов.

Заключение

В ходе выполнения выпускной квалификационной работы был разработан модуль анализа ситуации, позволяющий определить направление движения автономной мобильной платформы в сцене. В рамках выполнения этой работы были решены следующие задачи:

- оптимизирована система управления автономной мобильной платформой и выбран способ связи с микроконтроллером для подачи управляющих команд;
- реализованы функции создания модели сцены по предобработанным данным;
- реализована модель вероятностного распределения истинного положения объектов сцены на основе получаемой информации;
- реализованы алгоритмы анализа сцены и поиска пути;
- проведены исследования алгоритмов в виртуальной среде.

Из трех рассмотренных алгоритмов поиска пути (метода полного перебора, пчелиного роя и потенциальных полей) для модуля анализа ситуации наиболее оптимален метод потенциальных полей, обладающий следующими характеристиками:

- наименьшая вычислительная сложность алгоритмы;
- алгоритм учитывает наличие препятствий вдоль выбранного маршрута;
- наименьшее число шагов для достижения заданной цели.

При этом выполняются требования по ресурсоемкости модуля и частоте получения новой информации – новые данные необходимы только при изменении направления движения автономной мобильной платформы, что существенно снижает нагрузку на аппаратную структуру.

Список использованных источников

1. Корлякова Е.Ю., Корлякова М.О., Лохмачев Н.В., Трушков Д.С. Анализ моделей распознавания образов в системах технического зрения для мобильных автономных платформ / Научные технологии в приборостроении и развитии инновационной деятельности в вузе. Калуга, 13-15 ноября 2018.: Материалы Всероссийской научно-технической конференции. Том 2 –Калуга: Издательство МГТУ им. Н. Э. Баумана, 2018.– С. 13-17
2. Козлов, В.Л. Методики повышения точности измерения расстояний на основе корреляционного анализа стереоизображения / В.Л. Козлов // Приборы и методы измерений. — 2018. — № 1. — С. 48-55. — ISSN 2220-9506. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/journal/issue/309262> (дата обращения: 31.01.2020). — Режим доступа: для авториз. пользователей.
3. Корлякова Е.Ю., Корлякова М.О., Вишнякова А.Н. Построение виртуальной сцены в среде MatLab / Научные технологии в приборостроении и развитии инновационной деятельности в вузе Калуга, 14-16 ноября 2017.: Материалы Всероссийской научно-технической конференции. Том 1 –Калуга: Издательство МГТУ им. Н. Э. Баумана, 2017.– С. 217-220
4. Bourabai Research [Электронный ресурс]: Технология виртуальной реальности VRML / Дата обращения: 23.12.2019.
5. MATLAB FOR STUDENTS// [Электронный ресурс] URL:https://matlab.ru/upload/Konkursy/2017-teacher-01-MATLAB_FOR_STUDENTS.pdf (дата обращения : 10.12.2019)
6. Automated Driving System Toolbox//[Электронный ресурс] URL: <https://matlab.ru/products/automated-driving-system-toolbox>(дата обращения : 12.02.2020)
7. А. Г. Ченцов, А. М. Григорьев, “Оптимизирующие мультивставки в задачах маршрутизации с ограничениями” / Вестн. Удмуртск. ун-та. Матем. Мех. Компьют. науки, 28:4 (2018), 513–530

8. Craig Reynolds, “Flocks, Herds, and Schools: A Distributed Behavioral Model” // Computer Graphics, 21(4), с. 25–34, 1987 г.
9. Н. Д. Ганелина, В. Д. Фроловский, “Исследование методов построения кратчайшего пути обхода отрезков на плоскости” / Сиб. журн. вычисл. матем., 9:3 (2016), 241–252
10. Алаева Д.Р. Метод пчелиного роя для решения задач на поиск экстремума функции / Вестник науки и образования. Физико-математические науки, 40:4 (2018), том 1, с. 14-19
11. Казаков К.А., Семенов В.А. Обзор современных методов планирования движения. / Труды ИСП РАН, том 28, вып. 4, 2016, с. 241-294
12. Корлякова Е.Ю., Лохмачев Н.В., Трушков Д.С. Управление движением мобильной платформы в условиях замкнутых помещений / Научные технологии в приборостроении и машиностроении и развитие инновационной деятельности в вузе. Калуга, 19-21 ноября 2019.: Материалы Всероссийской научно-технической конференции. Том 1 –Калуга: Издательство МГТУ им. Н. Э. Баумана, 2019.– С. 209-211
13. Гришин Р.И., Нестеренко Е.С., Корлякова М.О. Разработка системы технического зрения для анализа среды на базе мобильной платформы / Научные технологии в приборостроении и машиностроении и развитие инновационной деятельности в вузе. Калуга, 13-15 ноября 2018.: Материалы Всероссийской научно-технической конференции. Том 2 –Калуга: Издательство МГТУ им. Н. Э. Баумана, 2018.– С. 21-23

Приложение А Микроконтроллер AtMega

Микроконтроллер AtMega1281 является высокопроизводительным восьмиразрядным микроконтроллером на базе микросхемы AVR RISC и имеет 54 линии ввода-вывода общего назначения, 32 рабочих регистра общего назначения, счетчик реального времени, 2 восьмиразрядных таймера-счетчика и 4 шестнадцатиразрядных.

Ввод-вывод данных на AtMega1281 организован с помощью портов USART и COM-портов управляющих устройств. Данный микроконтроллер имеет два порта USART, один из которых предназначен для управления с Raspberry Pi, а второй используется для передачи данных через Bluetooth или Wi-Fi.

На рисунке А.1 представлено изображение чипа AtMega1281. Его характеристики можно видеть в таблице А.1.

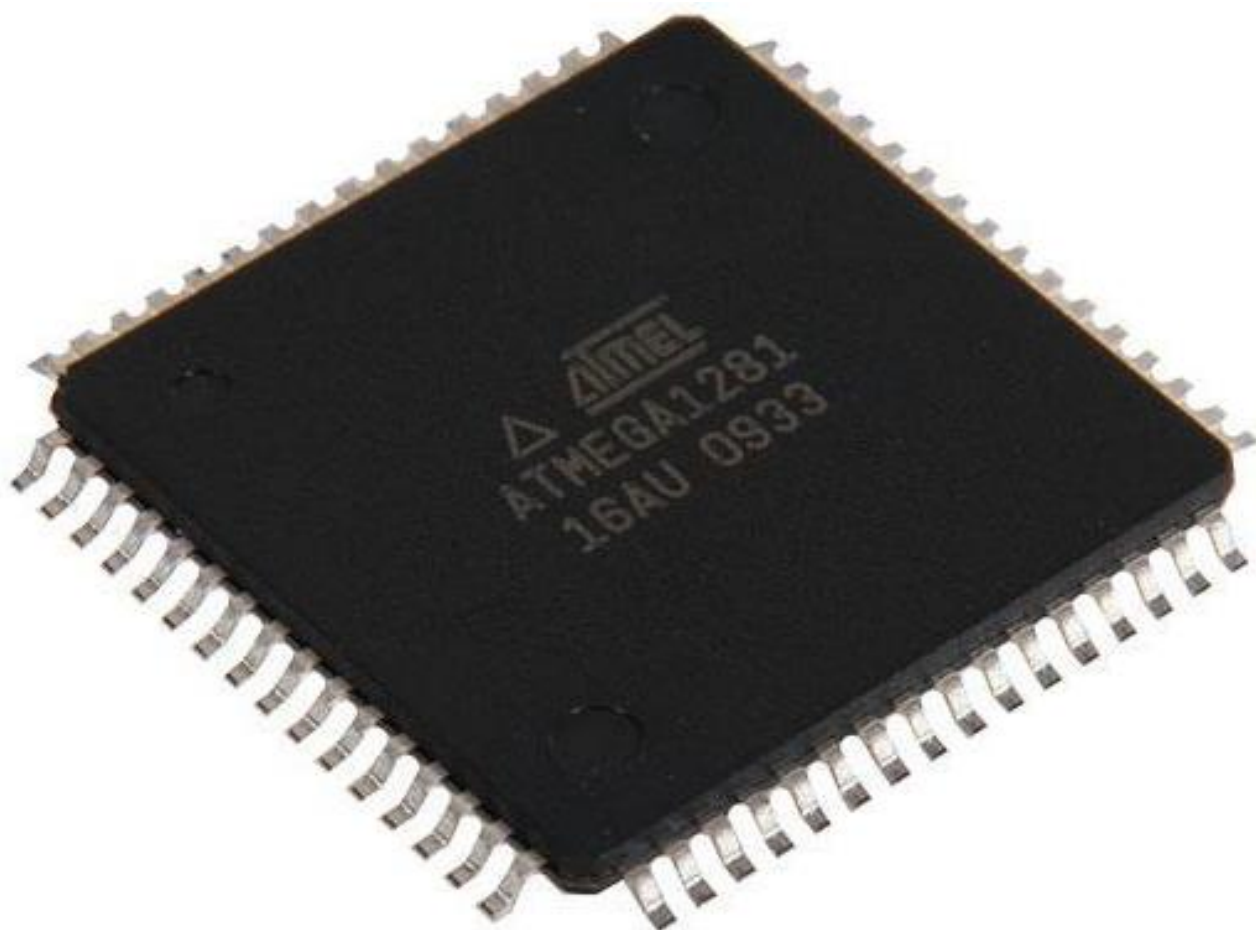


Рисунок А.1 – Внешний вид микрочипа Atmega1281

Таблица А.1 – Характеристики Atmega1281

ЦПУ: Ядро	<u>AVR</u>
ЦПУ: F,МГц	от 0 до 16
Память: Flash,КБайт	128
Память: RAM,КБайт	8
Память: EEPROM,КБайт	4
I/O (макс.),шт.	54
Таймеры: 8-бит,шт	2
Таймеры: 16-бит,шт	4
Таймеры: Каналов ШИМ,шт	9
Таймеры: RTC	Да
Интерфейсы: UART,шт	2
Интерфейсы: SPI,шт	1
Аналоговые входы: Разрядов АЦП,бит	10
Аналоговые входы: Каналов АЦП,шт	8
Аналоговые входы: Быстродействие АЦП,kSPS	76.9
Аналоговые входы: Аналоговый компаратор,шт	1
V _{CC} ,В	от 1.8 до 5.5
I _{CC} ,мА	16
T _A ,°C	от -40 до 85
Корпус	<u>TQFP-64 MLF</u> (VQFN) 64

Приложение Б Микрокомпьютер Raspberri Pi 3 B+

Микрокомпьютер Raspberry Pi 3 Model B+ - это функциональная сборка, оптимальная для построения автоматизированных систем благодаря своей компактности и большому объему оперативной памяти. Он снабжен контроллерами Wi-Fi и Bluetooth и может работать с внешними устройствами ввода-вывода, в том числе с камерой. Микрокомпьютер Raspberry Pi служит для отправки команд на AtMega1281. К нему подключена камера для определения положения в пространстве.

На рисунке Б.1 представлен внешний вид микрокомпьютера Raspberry Pi 3+, а в таблице Б.1 – его характеристики. Проанализировав их, можно удостовериться в том, что данный микрокомпьютер способен справиться с задачей обработки видеопотока в реальном времени.



Рисунок Б.1 – Внешний вид Raspberry Pi 3+

Таблица Б.1 – Характеристики Raspberry Pi 3 B+

Система на кристалле (SoC)	Broadcom BCM2837B0 (CPU + GPU + RAM)
Процессор	64-битный четырёхъядерный ARMv8 Cortex-A53 процессор с тактовой частотой 1.4 ГГц; 16 КБ cache L1 и 512 КБ cache L2
Графический процессор	Двухъядерный процессор (GPU) VideoCore IV® (3D GPU @ 300 МГц, видео GPU @ 400 МГц) поддерживает стандарты OpenGL ES 2.0, OpenVG, MPEG-2, VC-1 и способен кодировать, декодировать и выводить Full HD-видео (1080p, 30 FPS, H.264 High-Profil)
ОЗУ	1 ГБ SDRAM LPDDR2
Ethernet	10/100/1000 Мбит Gigabit Ethernet (через USB 2.0) (контроллер LAN7515 — USB 2.0 Hub и Ethernet)
Wi-Fi/Bluetooth	2.4 ГГц и 5 ГГц IEEE 802.11.b/g/n/ac WI-FI и Bluetooth 4.2 Low Energy (BLE), обеспечиваемые микросхемой Cypress CYW43455
Видео вход	1 x CSI-2 для подключения камеры по интерфейсу MIPI
USB-порты	4 порта USB 2.0 через USB hub в Microchip LAN7515
Периферия	40 портов ввода-вывода общего назначения (GPIO), UART (Serial), I ² C/TWI, SPI с селектором между двумя устройствами; пины питания: 3,3 В, 5 В и земля.
ОС	Ubuntu, Debian, Fedora, Arch Linux, Gentoo, RISC OS, Android, Firefox OS, NetBSD, FreeBSD, Slackware, Tiny Core Linux, Windows 10 IOT

Приложение В Алгоритм управления автономной платформой с Atmega1281

```
// Last version of our code. Changed at 5'th of November. Kate
/*****
NIRobot_4
YTC.B-71 :)

Chip type           : ATmega1281
Program type        : Application
Clock frequency     : 14,740000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 2048

functions:
char getcharl(void) and void putcharl(char c) - are for USART1, created
bu this program
char getchar(void) and void putchar(char c) - are for USART0, created bu
this program
void putnumber( int count) - put big integers (decimal!)

interrupt [TIM5_COMPA] void timer5_compa_isr(void) - left wheel's PWM
change to stabilize when it move forward ('w')

void rotate(char c) - turn left('l') or right('r') //make america great
again and this code so

void choice(char f) - switch f for move

//Main part
void main(void)

letters and commands:
f                                flag
'w' : // Move forward!          w //flag for interrupt Timer5
's' : // Move back!             w (s)
'd' : // Move left!             d
'a' : // Move right!            a
'f' : // Move stop!            f

'z' : // Move fast!
'x' : // Move slow!

'u' : // Move forward-left      f
'i' : // Move forward-right     f
'o' : // Move back-left        f
'p' : // Move back-right       f

'l' : // Turn left, 45          f
'r' : // Turn right, 45         f
*****/

#include <megal281.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
```

```

#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

//usarts
void putchar1(char c);
void choice(char f) ;
char getchar1(void);
char getchar(void);
char f,flag;//,*str; //command and flag
// USART0 Receiver buffer
#define RX_BUFFER_SIZE0 8
char rx_buffer0[RX_BUFFER_SIZE0];

#if RX_BUFFER_SIZE0<256
unsigned char rx_wr_index0,rx_rd_index0,rx_counter0;
#else
unsigned int rx_wr_index0,rx_rd_index0,rx_counter0;
#endif

// This flag is set on USART0 Receiver buffer overflow
bit rx_buffer_overflow0;

// USART0 Receiver interrupt service routine
interrupt [USART0_RXC] void usart0_rx_isr(void)
{
    #asm("cli")
    UCSR0A =0x00;
    f=UDR0;
    choice(f);
    #asm("sei")
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART0 Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
    char data;
    while (rx_counter0==0);
    data=rx_buffer0[rx_rd_index0];
    if (++rx_rd_index0 == RX_BUFFER_SIZE0) rx_rd_index0=0;
    #asm("cli")
    --rx_counter0;
    #asm("sei")
    return data;
}
#pragma used-
#endif

// USART0 Transmitter buffer
#define TX_BUFFER_SIZE0 8
char tx_buffer0[TX_BUFFER_SIZE0];

#if TX_BUFFER_SIZE0<256

```

```

unsigned char tx_wr_index0,tx_rd_index0,tx_counter0;
#else
unsigned int tx_wr_index0,tx_rd_index0,tx_counter0;
#endif

// USART0 Transmitter interrupt service routine
interrupt [USART0_TXC] void usart0_tx_isr(void)
{
    if (tx_counter0)
    {
        --tx_counter0;
        UDR0=tx_buffer0[tx_rd_index0];
        if (++tx_rd_index0 == TX_BUFFER_SIZE0) tx_rd_index0=0;
    };
}

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART0 Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
    while (tx_counter0 == TX_BUFFER_SIZE0);
    #asm("cli")
    if (tx_counter0 || ((UCSR0A & DATA_REGISTER_EMPTY)==0))
    {
        tx_buffer0[tx_wr_index0]=c;
        if (++tx_wr_index0 == TX_BUFFER_SIZE0) tx_wr_index0=0;
        ++tx_counter0;
    }
    else
        UDR0=c;
    #asm("sei")
}
#pragma used-
#endif

// USART1 Receiver buffer
#define RX_BUFFER_SIZE1 8
char rx_buffer1[RX_BUFFER_SIZE1];

#if RX_BUFFER_SIZE1<256
unsigned char rx_wr_index1,rx_rd_index1,rx_counter1;
#else
unsigned int rx_wr_index1,rx_rd_index1,rx_counter1;
#endif

// This flag is set on USART1 Receiver buffer overflow
bit rx_buffer_overflow1;

// USART1 Receiver interrupt service routine
interrupt [USART1_RXC] void usart1_rx_isr(void)
{
    #asm("cli")
    UCSR1A =0x00;
    f=UDR1;
    choice(f);
    #asm("sei")
}

// Get a character from the USART1 Receiver buffer
#pragma used+
char getchar1(void)
{

```

```

char data;
while (rx_counter1==0);
data=rx_buffer1[rx_rd_index1];
if (++rx_rd_index1 == RX_BUFFER_SIZE1) rx_rd_index1=0;
#asm("cli")
--rx_counter1;
#asm("sei")
return data;
}
#pragma used-
// USART1 Transmitter buffer
#define TX_BUFFER_SIZE1 8
char tx_buffer1[TX_BUFFER_SIZE1];

#if TX_BUFFER_SIZE1<256
unsigned char tx_wr_index1,tx_rd_index1,tx_counter1;
#else
unsigned int tx_wr_index1,tx_rd_index1,tx_counter1;
#endif

// USART1 Transmitter interrupt service routine
interrupt [USART1_TXC] void usart1_tx_isr(void)
{

if (tx_counter1)
{
--tx_counter1;
UDR1=tx_buffer1[tx_rd_index1];
if (++tx_rd_index1 == TX_BUFFER_SIZE1) tx_rd_index1=0;
};
}

// Write a character to the USART1 Transmitter buffer
#pragma used+
void putchar1(char c)
{
while (tx_counter1 == TX_BUFFER_SIZE1);
#asm("cli")
if (tx_counter1 || ((UCSR1A & DATA_REGISTER_EMPTY)==0))
{
tx_buffer1[tx_wr_index1]=c;
if (++tx_wr_index1 == TX_BUFFER_SIZE1) tx_wr_index1=0;
++tx_counter1;
}
else
UDR1=c;
#asm("sei")
}
#pragma used-

// Declare your global variables here

char f0,f1, fc0, fc1;//,*str; //command and flag
int count1,count3,i; // impulse counters
int count10,count30;
int PWML,PWMR; //PWM main parameters
int C_PWML, C_PWMR, C_PWM=50; //PWM add parameters

//Timer 5 interrupt //Left wheel's PWM change to stabilize
// Timer 5 output compare A interrupt service routine
interrupt [TIM5_COMPA] void timer5_compa_isr(void)
{
#asm("sei")
count1 = TCNT1L;//low part of left wheel impulse

```

```

count3 = TCNT3L; //low part of right wheel impulse
count10 = TCNT1H; //high part of left wheel impulse
count30 = TCNT3H;
TCNT3H=0x00;TCNT1H=0x00; //clear low impulse counters
TCNT3L=0x00;TCNT1L=0x00; //clear low impulse counters

//stabilize when move forward
if ( flag == 'w') {

if ( (count30-count10) >=1){ //right is more faster
    PWML=PWML+2;
}
else if ( (count10-count30) >=1) {
    PWML=PWML-2;
}
else if ( (count3-count1) >= 1 ){ //right is faster
    PWML=PWML+1; //++; // rise left PWM
}
else if ( (count1-count3) >= 1 ) { //left is faster
    PWML=PWML-1; //--; // low left PWM
}
OCR0A=PWML; // left PWM
}

} //end of Timer 5 interrupt

//

//input string from integer
void putnumber( int count)
{
    char *str;
    itoa(count,str);
    for (i=0;i<=strlen(str);i++)
    {
        putchar1(str[i]);
    }
}

//

// turn left or right
void rotate(char c)
{
    int c0,c1,c2,c3,counter_l,counter_h;
    int angle_H=0x0A, angle_L=0xBE; // 0x157C = 5500 impulses = 1
full wheel rotation //0x0ABE = 2750 = 1/2 full wheel rotation
    //stop mashine
    PORTC.0 = 0;
    PORTC.1 = 0;

    PORTC.2 = 0;
    PORTC.3 = 0;
    OCR0A=0x93; //normal speed
    OCR2A=0x93;
    flag='f';
    TCNT3H=0x00;TCNT1H=0x00; //clear impulse counters high
    TCNT3L=0x00;TCNT1L=0x00; //clear impulse counters low

    if (c=='r') //turn right        left pwm work
    {
        c0=1;

```

```

        c1=0;
        c2=0;
        c3=0;
    }
    else //(c=='l') turn left      right pwm work
    {
        c0=0;
        c1=0;
        c2=0;
        c3=1;
    }
    counter_h=0;
    counter_l=0;
    while ((counter_h<(angle_H))||(counter_l<angle_L))
    {
        if (c=='r')    //turn right
        {
            counter_l=TCNT1L;    //counters for left wheel
            counter_h=TCNT1H;
        }
        else //(c=='l') turn left
        {
            counter_l=TCNT3L;    //counters for right wheel
            counter_h=TCNT3H;
        }

        PORTC.0 = c0;
        PORTC.1 = c1;
        PORTC.2 = c2;
        PORTC.3 = c3;

        if ((counter_h>(angle_H))&&(counter_l>angle_L))
        {
            break;
        }
    }
    //stop mashine
    PORTC.0 = 0;
    PORTC.1 = 0;

    PORTC.2 = 0;
    PORTC.3 = 0;
    flag='f';
    //antiterror
    if (c=='r')    //turn right      left pwm work
    {
        c0=0;
        c1=0;
        c2=1;
        c3=0;
        angle_L=TCNT3L;
        angle_H=TCNT3H;
    }
    else //(c=='l') turn left      right pwm work
    {
        c0=0;
        c1=1;
        c2=0;
        c3=0;
        angle_L=TCNT1L;
        angle_H=TCNT1H;
    }
    TCNT3H=0x00;TCNT1H=0x00; //clear impulse counters high
    TCNT3L=0x00;TCNT1L=0x00; //clear impulse counters low

```



```

counter_h=0;
counter_l=0;
while ((counter_h<(angle_H))||(counter_l<angle_L))
{
    if (c=='l')    //toturn right
    {
        counter_l=TCNT1L;    //counters for left wheel
        counter_h=TCNT1H;
    }
    else //(c=='r') toturn left
    {
        counter_l=TCNT3L;    //counters for right wheel
        counter_h=TCNT3H;
    }

    PORTC.0 = c0;
    PORTC.1 = c1;
    PORTC.2 = c2;
    PORTC.3 = c3;

    if ((counter_h>(angle_H))&&(counter_l>angle_L))
    {
        break;
    }
}
//stop mashine
PORTC.0 = 0;
PORTC.1 = 0;

PORTC.2 = 0;
PORTC.3 = 0;
flag='f';
OCR0A=PWML;    //normal speed
OCR2A=PWMR;
}

// case
void choice(char a)
{
    putchar1(a);
    switch (a) {
        case 'w' :    // Move forward!
            //Start Timer5
            TCCR5B=0x0D;    //Timer5 parameters
            OCR5AH=0x05;
            OCR5AL=0x9F;
            TIMSK5=0x02;    //Timer5 interrupt

            PORTC.0 = 1;
            PORTC.1 = 0;

            PORTC.2 = 0;
            PORTC.3 = 1;

            C_PWML=0;
            C_PWMR=0;

            flag='w';
            break;

        case 's' :    // Move back!
            //Start Timer5
            TCCR5B=0x0D;    //Timer5 parameters
            OCR5AH=0x05;

```

```

OCR5AL=0x9F;
TIMSK5=0x02; //Timer5 interrupt

PORTC.0 = 0;
PORTC.1 = 1;

PORTC.2 = 1;
PORTC.3 = 0;

C_PWML=0;
C_PWMR=0;

flag='w'; //try
break;

case 'd' : // Move left!
//Stop Timer5
TCCR5B=0x00; //Timer5 parameters
OCR5AH=0x00;
OCR5AL=0x00;
TIMSK5=0x00; //Timer5 interrupt

PORTC.0 = 1;
PORTC.1 = 0;

PORTC.2 = 1;
PORTC.3 = 0;

C_PWML=0;
C_PWMR=0;

flag='d';
break;

case 'a' : // Move right!
//Stop Timer5
TCCR5B=0x00; //Timer5 parameters
OCR5AH=0x00;
OCR5AL=0x00;
TIMSK5=0x00; //Timer5 interrupt

PORTC.0 = 0;
PORTC.1 = 1;

PORTC.2 = 0;
PORTC.3 = 1;

C_PWML=0;
C_PWMR=0;

flag='a';
break;

case 'f' : // Move stop!
//Stop Timer5
TCCR5B=0x00; //Timer5 parameters
OCR5AH=0x00;
OCR5AL=0x00;
TIMSK5=0x00; //Timer5 interrupt

PORTC.0 = 0;
PORTC.1 = 0;

PORTC.2 = 0;

```

```

PORTC.3 = 0;

C_PWML=0;
C_PWMR=0;

flag='f';
break;

case 'z' : // Move fast!
    PWML++;
    PWMR++;
    break;

case 'x' : // Move slow!
    PWML--;
    PWMR--;
    break;

case 'u' : // Move forward-left
    //Stop Timer5
    TCCR5B=0x00; //Timer5 parameters
    OCR5AH=0x00;
    OCR5AL=0x00;
    TIMSK5=0x00; //Timer5 interrupt

    C_PWML=0;
    C_PWMR=C_PWM;

    PORTC.0 = 1;
    PORTC.1 = 0;

    PORTC.2 = 0;
    PORTC.3 = 1;

    flag='f';
    break;

case 'i' : // Move forward-right
    //Stop Timer5
    TCCR5B=0x00; //Timer5 parameters
    OCR5AH=0x00;
    OCR5AL=0x00;
    TIMSK5=0x00; //Timer5 interrupt

    C_PWML=C_PWM;
    C_PWMR=0;

    PORTC.0 = 1;
    PORTC.1 = 0;

    PORTC.2 = 0;
    PORTC.3 = 1;

    flag='f';
    break;

case 'o' : // Move back-left
    //Stop Timer5
    TCCR5B=0x00; //Timer5 parameters
    OCR5AH=0x00;
    OCR5AL=0x00;
    TIMSK5=0x00; //Timer5 interrupt

```

```

C_PWML=0;
C_PWMR=C_PWM;

PORTC.0 = 0;
PORTC.1 = 1;

PORTC.2 = 1;
PORTC.3 = 0;

flag='f';
break;

case 'p' : // Move back-right
//Stop Timer5
TCCR5B=0x00; //Timer5 parameters
OCR5AH=0x00;
OCR5AL=0x00;
TIMSK5=0x00; //Timer5 interrupt

C_PWML=C_PWM;
C_PWMR=0;

PORTC.0 = 0;
PORTC.1 = 1;

PORTC.2 = 1;
PORTC.3 = 0;

flag='f';
break;

case 'l' : // Turn left, 45
//Stop Timer5
TCCR5B=0x00; //Timer5 parameters
OCR5AH=0x00;
OCR5AL=0x00;
TIMSK5=0x00; //Timer5 interrupt

rotate('l');
C_PWML=0;
C_PWMR=0;

flag='f';
break;

case 'r' : // Turn right, 45
//Stop Timer5
TCCR5B=0x00; //Timer5 parameters
OCR5AH=0x00;
OCR5AL=0x00;
TIMSK5=0x00; //Timer5 interrupt

rotate('r');
C_PWML=0;
C_PWMR=0;

flag='f';
break;

default:
break;
} //and of switch

```

```

        OCR0A=PWML+C_PWML;    // left PWM
        OCR2A=PWMR+C_PWMR;    // right PWM

    }

    //

    //Main part
    void main(void)
    {
        // Crystal Oscillator division factor: 1
        #pragma optsize-
        CLKPR=0x80;
        CLKPR=0x00;
        #ifdef _OPTIMIZE_SIZE_
        #pragma optsize+
        #endif

        // Input/Output Ports initialization
        // Port A initialization
        // Func7=In  Func6=In  Func5=In  Func4=In  Func3=In  Func2=In  Func1=In
Func0=In
        // State7=T  State6=T  State5=T  State4=T  State3=T  State2=T  State1=T
State0=T
        PORTA=0x00;
        DDRA=0x00;

        // Port B initialization
        // Func7=Out  Func6=In  Func5=In  Func4=Out  Func3=In  Func2=In  Func1=In
Func0=In
        // State7=0  State6=T  State5=T  State4=0  State3=T  State2=T  State1=T
State0=T
        PORTB=0x00;
        DDRB=0x90;

        // Port C initialization
        // Func7=In  Func6=In  Func5=In  Func4=In  Func3=Out  Func2=Out  Func1=Out
Func0=Out
        // State7=T  State6=T  State5=T  State4=T  State3=0  State2=0  State1=0
State0=0
        PORTC=0x00;
        DDRC=0x0F;

        // Port D initialization
        // Func7=In  Func6=In  Func5=Out  Func4=In  Func3=In  Func2=In  Func1=In
Func0=In
        // State7=T  State6=T  State5=0  State4=T  State3=T  State2=T  State1=T
State0=T
        PORTD=0x00;
        DDRD=0x20;

        // Port E initialization
        // Func7=In  Func6=In  Func5=In  Func4=In  Func3=In  Func2=In  Func1=Out
Func0=In
        // State7=T  State6=T  State5=T  State4=T  State3=T  State2=T  State1=0
State0=T
        PORTE=0x00;
        DDRE=0x02;

        // Port F initialization
        // Func7=In  Func6=In  Func5=In  Func4=In  Func3=In  Func2=In  Func1=In
Func0=In
        // State7=T  State6=T  State5=T  State4=T  State3=T  State2=T  State1=T
State0=T

```

```

PORTF=0x00;
DDRF=0x00;

// Port G initialization
// Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State5=T State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;

// Timer 0
// Left wheel's PWM
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 230,313 kHz
// Mode: Fast PWM top=FFh
// OC0A output: Non-Inverted PWM
// OC0B output: Disconnected
TCCR0A=0x83; // 1000 0011
TCCR0B=0x03;
TCNT0=0x00;
OCR0A=0x4F;
OCR0B=0x00;

// Timer 2
// Right wheel's PWM
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 230,313 kHz
// Mode: Fast PWM top=FFh
// OC2A output: Non-Inverted PWM
// OC2B output: Disconnected
ASSR=0x00;
TCCR2A=0x83;
TCCR2B=0x03;
TCNT2=0x00;
OCR2A=0x4F;
OCR2B=0x00;

// Timer 1
// Left wheel's impulse counter
// Timer/Counter 1 initialization
// Clock source: T1 pin Falling Edge
// Mode: Normal top=FFFFh
// Noise Canceler: Off
// Input Capture on Falling Edge
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x06;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

```

```

OCR1CH=0x00;
OCR1CL=0x00;

// Timer 3
// Right wheel's impulse counter
// Timer/Counter 3 initialization
// Clock source: T3 pin Falling Edge
// Mode: Normal top=FFFFh
// Noise Canceler: Off
// Input Capture on Falling Edge
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Timer 3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x06;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;

// Timer 5
// Time counter for Timer 1 and 3
// Timer/Counter 5 initialization
// Clock source: System Clock
// Clock value: 14,395 kHz
// Mode: CTC top=OCR5A
// OC5A output: Discon.
// OC5B output: Discon.
// OC5C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 5 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR5A=0x00;
TCCR5B=0x0D;
TCNT5H=0x00;
TCNT5L=0x00;
ICR5H=0x00;
ICR5L=0x00;
OCR5AH=0x05; //59F - 1439 Hz -> time=0.1c - time for interrupt
OCR5AL=0x9F;
OCR5BH=0x00;
OCR5BL=0x00;
OCR5CH=0x00;
OCR5CL=0x00;

// Timer 4
// no used
// Timer/Counter 4 initialization

```

```

// Clock source: System Clock
// Clock value: Timer 4 Stopped
// Mode: Normal top=FFFFh
// OC4A output: Discon.
// OC4B output: Discon.
// OC4C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 4 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR4A=0x00;
TCCR4B=0x00;
TCNT4H=0x00;
TCNT4L=0x00;
ICR4H=0x00;
ICR4L=0x00;
OCR4AH=0x00;
OCR4AL=0x00;
OCR4BH=0x00;
OCR4BL=0x00;
OCR4CH=0x00;
OCR4CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;

// PCINT0 interrupt: Off
// PCINT1 interrupt: Off
// PCINT2 interrupt: Off
// PCINT3 interrupt: Off
// PCINT4 interrupt: Off
// PCINT5 interrupt: Off
// PCINT6 interrupt: Off
// PCINT7 interrupt: Off
// PCINT8 interrupt: Off
// PCINT9 interrupt: Off
// PCINT10 interrupt: Off
// PCINT11 interrupt: Off
// PCINT12 interrupt: Off
// PCINT13 interrupt: Off
// PCINT14 interrupt: Off
// PCINT15 interrupt: Off
// PCINT16 interrupt: Off
// PCINT17 interrupt: Off
// PCINT18 interrupt: Off
// PCINT19 interrupt: Off
// PCINT20 interrupt: Off
// PCINT21 interrupt: Off
// PCINT22 interrupt: Off
// PCINT23 interrupt: Off

```



```

PCMSK0=0x00;
PCMSK1=0x00;
PCMSK2=0x00;
PCICR=0x00;

// Timer/Counter 0 Interrupt(s) initialization
TIMSK0=0x00;
// Timer/Counter 1 Interrupt(s) initialization
TIMSK1=0x00;
// Timer/Counter 2 Interrupt(s) initialization
TIMSK2=0x00;
// Timer/Counter 3 Interrupt(s) initialization
TIMSK3=0x00;
// Timer/Counter 4 Interrupt(s) initialization
TIMSK4=0x00;
// Timer/Counter 5 Interrupt(s) initialization
TIMSK5=0x02;

// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 115200
UCSR0A=0x00;
UCSR0B=0x98; //18 - D8 (R,T) - 98 (R) 1001 1000
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x07;

// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: On
// USART1 Mode: Asynchronous
// USART1 Baud Rate: 9600
UCSR1A=0x00;
UCSR1B=0x98; // 1001 1000
UCSR1C=0x06; // 0000 0110
UBRR1H=0x00;
UBRR1L=0x5F; // 0F - 57600 //5F - 9600

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
ADCSRB=0x00;

#asm("sei")
count1=0;
count3=0;
PWML=OCR0A;
PWMR=OCR2A;
C_PWML=0;
C_PWMR=0;
//PWML,PWMR - PWM main parameters
//C_PWML, C_PWMR - PWM add parameters
// Realy main part
PORTB.7 = 1;//left wheel start
PORTB.4 = 1;//right wheel start
choice('f');//stop
while (1)
{
    } //end of main while

```

```
} //end of main main
```

Приложение Г Терминал Bluetooth связи

Для организации беспроводной связи с использованием модуля Bluetooth необходимо линейно-ориентированное приложение терминала/консоли для микроконтроллеров, arduinos и других устройств с последовательным интерфейсом/UART, подключенных с помощью преобразователя bluetooth в serial к управляющему устройству. В случае внешнего управления автономной платформой использовался планшет Samsung под Android. На рисунке Г.1 выбранное приложение.

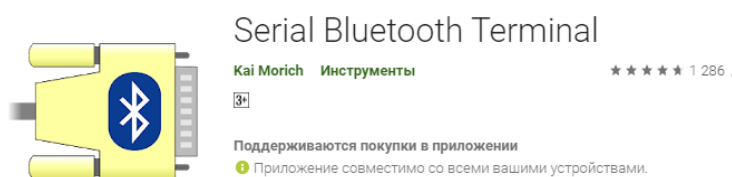


Рисунок Г.1 – Иконка приложения и краткая информация о нем

Мобильное приложение Serial Bluetooth Terminal бесплатно и доступно для свободного скачивания в сети или на площадках магазинов Android и Apple, как наиболее распространенных платформах.

Основные характеристики приложения:

Размер 1.4МБ

Версия приложения 1.31

Версии Bluetooth Bluetooth Classic; Bluetooth LE

Версия прошивки телефона 4.3 и выше

Настраиваемая пользователем конфигурация управляющих команд позволяет создать удобный интерфейс для работы с автономной мобильной платформой, рисунок Г.2.

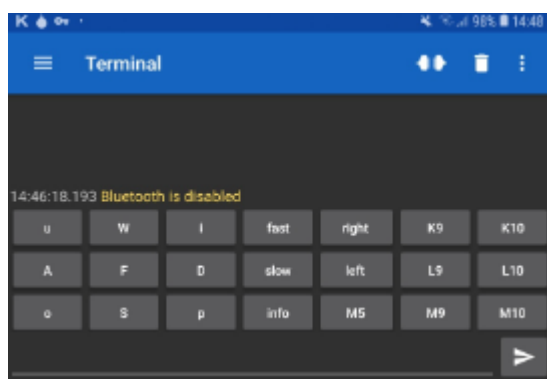


Рисунок Г.2 – Окно программы при пользовательских настройках

Приложение Д Структурные схемы программ модуля

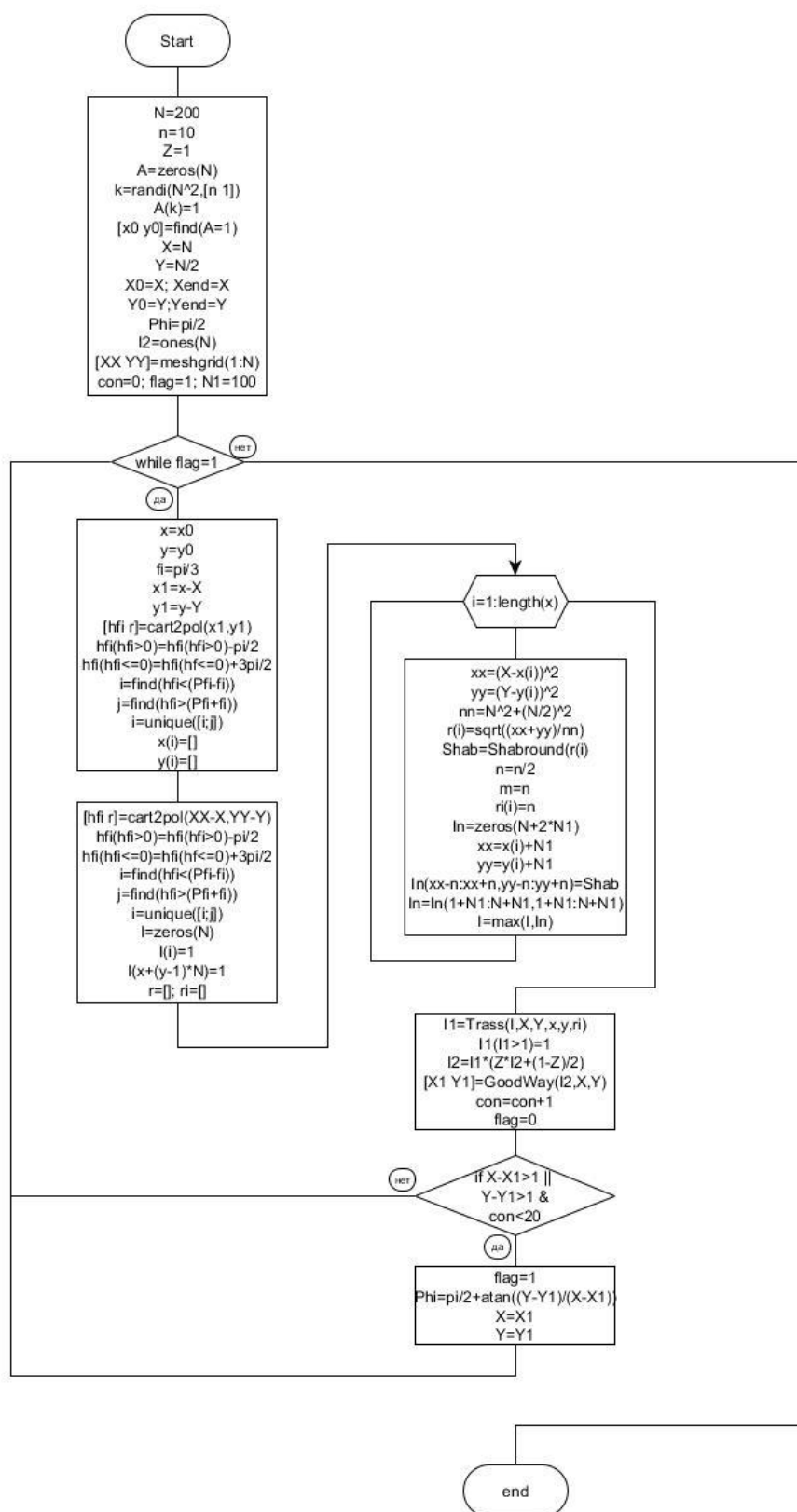


Рисунок Д.1 – Структурная схема программы тестирования модуля

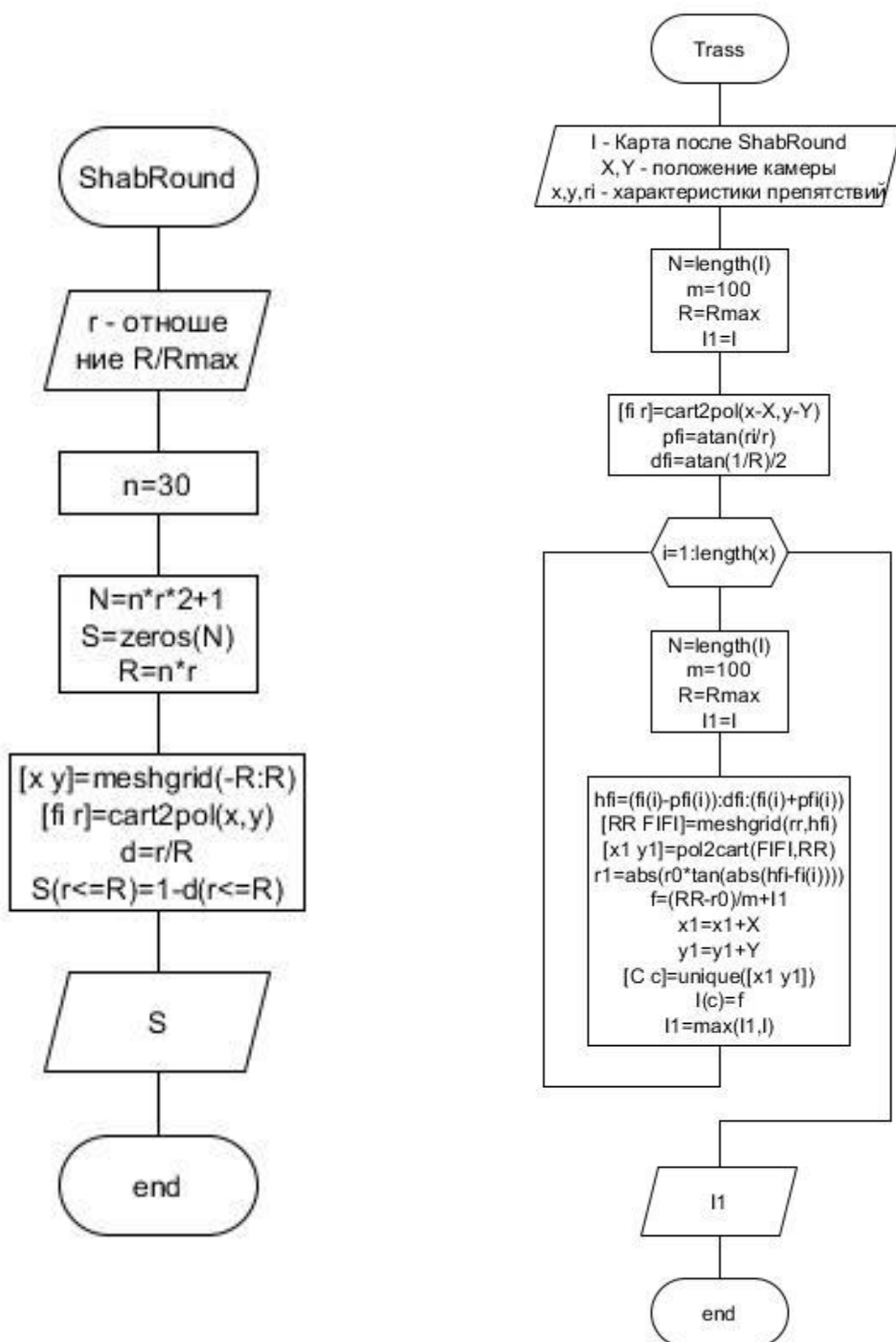


Рисунок Д.2 – Структурные схемы функций создания модели сцены из точки наблюдения

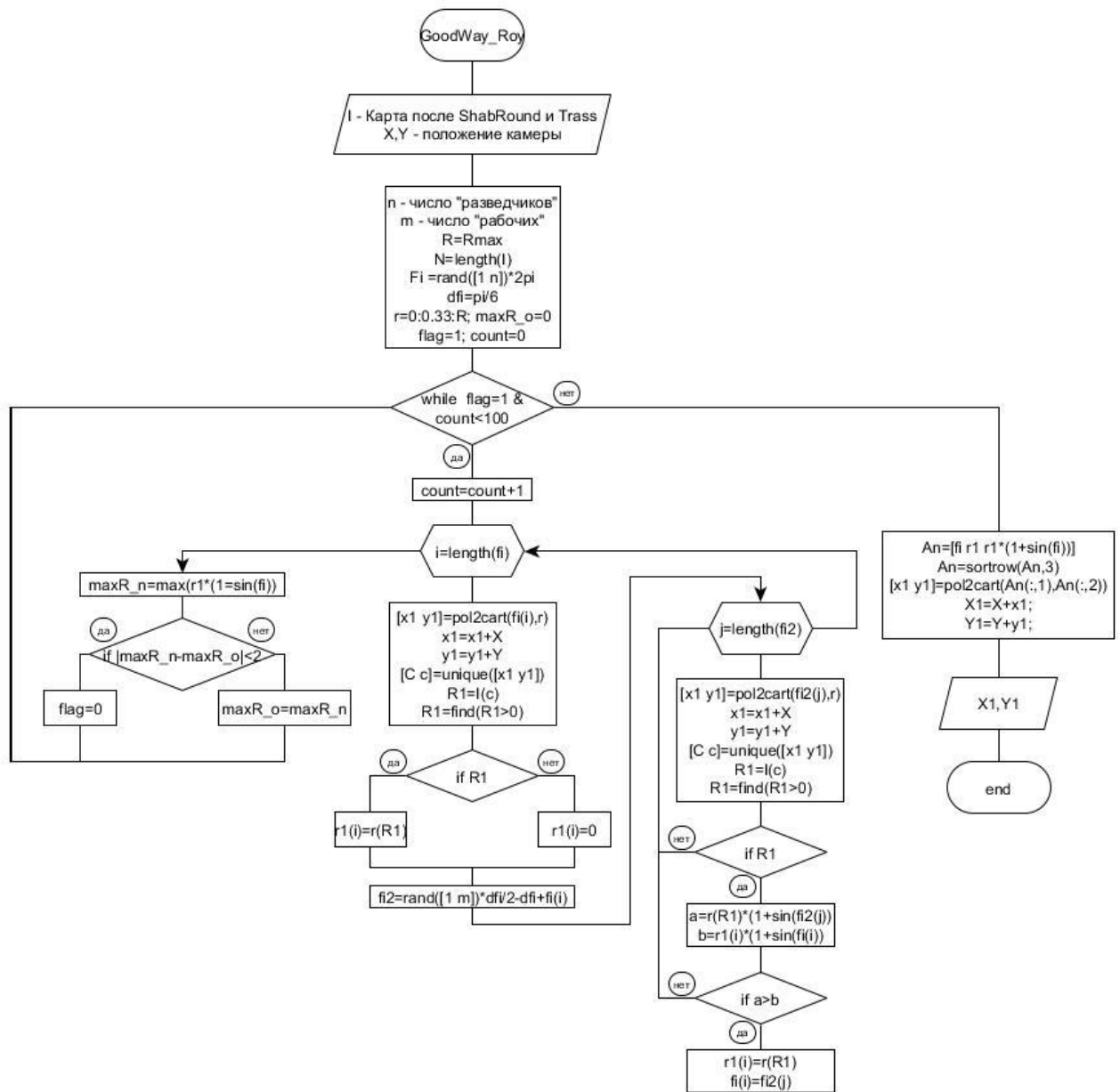


Рисунок Д.4 – Структурная схема функции поиска пути по алгоритму пчелиного роя

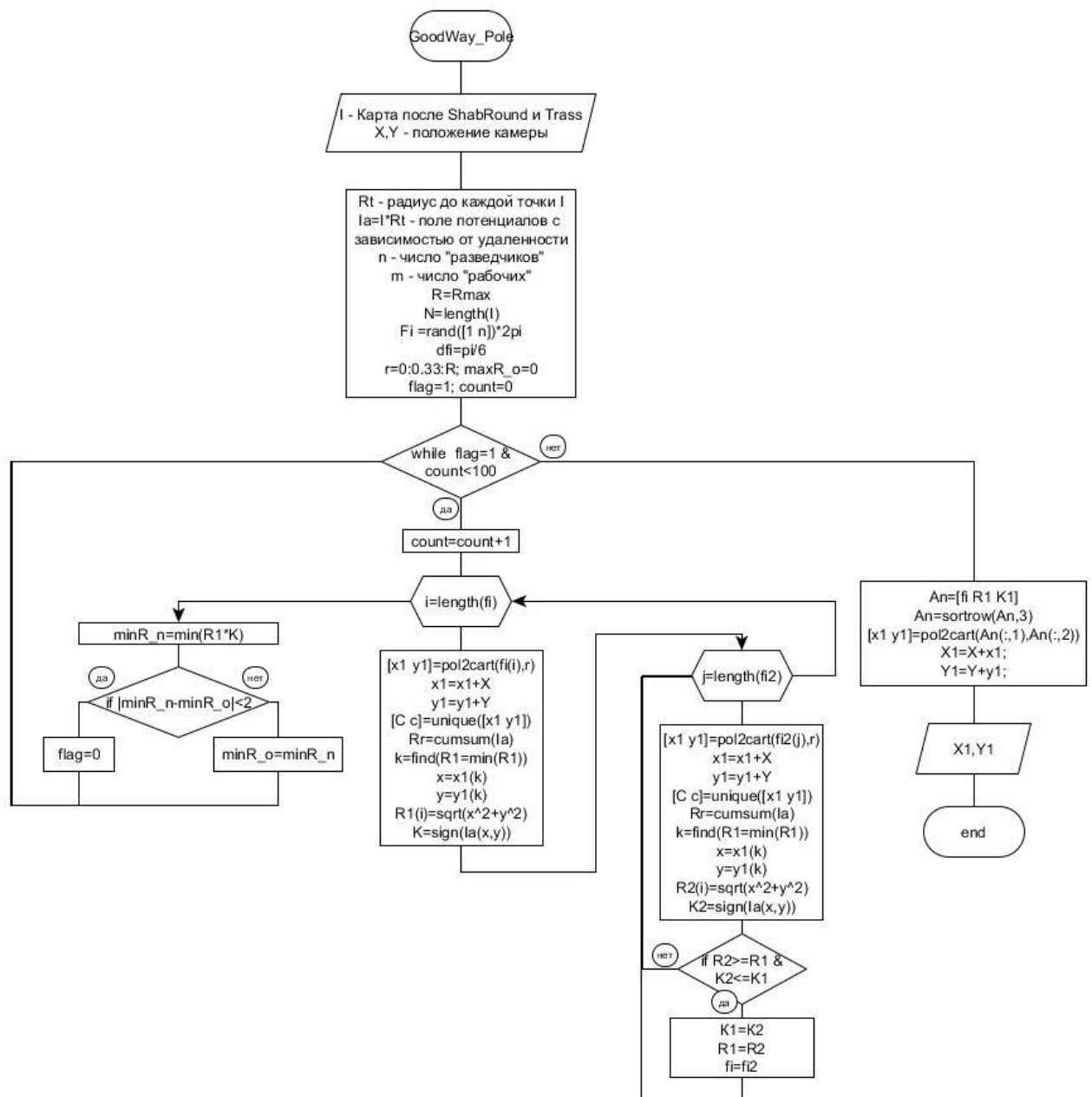


Рисунок Д.5 – Структурная схема функции поиска пути методом потенциалов

Приложение Е Программы модуля анализа ситуаций

```
clear all
clc
N=200;
n=15;
Z=1;%0.75;%коэффициент забывания
A=zeros(N);
A(randi(N^2,[n 1]))=1;
[x0,y0]=find(A==1);
x0(x0>(N-20))=N-20;
A=zeros(N);
A(x0+(y0-1)*N)=1;

figure(1)
imshow(A)

%% камера в 200;100
X=N;Y=round(N/2);
X0=X;Y0=Y;
Xend=X;Yend=Y;
[x0,y0]=find(A==1);

%% зациклим
flag=1;
Pfi=pi/2; % направление зрения

N1=100;
I2=ones(N);
Ind=zeros(N+2*N1);

[XX,YY]=meshgrid(1:N);
con=0;cof=0;
while flag==1
x=x0;y=y0;
%% область зрения:
fi=pi/3; % если не видно 30 градусов

[hfi,r]=cart2pol(XX-X,YY-Y);
hfi2=hfi;
hfi2(hfi>0)=hfi(hfi>0)-pi/2;
hfi2(hfi<=0)=hfi(hfi<=0)+3*pi/2;
hfi=hfi2;
i=find(hfi<(Pfi-fi));
i=unique([i; find(hfi>(Pfi+fi))]);
%% карта глубины с точки зрения робота:
I=zeros(N);
I(i)=1;
I=I';
r=[];ri=[];
Ia=I+A;
[x y]=find(A.*(1-I)==1);
I=Ia;
for i=1:length(x)
    r(i)=sqrt((X-x(i))^2+(Y-y(i))^2)/sqrt(N^2+(N/2)^2); %0..1
    Shab=ShabRound(r(i));
    [n m]=size(Shab);
    n=fix(n/2);
    m=fix(m/2);
    ri(i)=n;
    if ri(i)==0
        ri(i)=1;
    end
end
```

```

end
In=zeros(N+2*N1);
In( x(i)+N1-n:x(i)+N1+n , y(i)+N1-m:y(i)+N1+m )=Shab;

In=In(1+N1:N+N1,1+N1:N+N1);

I=max(I, In);

end
I1=Trass(I,X,Y,x,y,ri);

I1(I1>1)=1;
I2=(I1.*(Z*I2+(1-Z)/2)); %забывание
figure(3)
hold off;
imshow(A+I2)
hold on
plot(Yend,Xend,'r')

%
cof=0;
while cof<3
[X1 Y1] = GoodWay_Roy( I2,X,Y );
con=con+1;

Y1(X1>=N)=[];
X1(X1>=N)=[];
Y1(Y1>=N)=[];
X1(X1>=N)=[];

Y1(X1<1)=[];
X1(X1<1)=[];
Y1(Y1<1)=[];
X1(Y1<1)=[];
nu=5;
Y1( abs(X-X1)<nu )=[];
X1( abs(X-X1)<nu )=[];
Y1( abs(Y-Y1)<nu )=[];
X1( abs(Y-Y1)<nu )=[];

if length(find( X1<nu ) )
    flag=0;
    Y1=Y1(find( X1<nu ));
    Y1=Y1(1);
    X1=X1(find( X1<nu ));
    X1=X1(1);
    Xend=[Xend;X1];Yend=[Yend;Y1];
    cof=3;
else

if length(X1)>=1
    flag=1;
    conti=1;
    nu=10;
    xend=round(Xend/nu)+1;
    yend=round(Yend/nu)+1;
    xend=xend+( yend-1 )*N;
    cofi=0;
    while conti>0 && cofi<=length(X1)
        [a b]=find( repmat(xend,[1 length(xend)])==xend');
        if length(a)>length(xend)
            flag=0;

```

```

        conti=-2;
    else
        x=round(X1(conti)/nu)+1;
        y=round(Y1(conti)/nu)+1;
        x=x+(y-1)*N;
        if length(find(xend==x))>=1
            cofi=cofi+1;
        else
            X1=X1(conti);
            Y1=Y1(conti);
            conti=0;
        end
    end
end
if conti==0

Pfi=atan( abs((Y-Y1) / (X-X1)) );
if X1<=X && Y1>=Y
    Pfi=atan( abs((X-X1) / (Y-Y1)) );
elseif X1>=X && Y1>=Y
    Pfi=Pfi-pi/2;
elseif X1>=X && Y1<=Y
    Pfi=atan( abs((X-X1) / (Y-Y1)) )+pi;
else
    Pfi=Pfi+pi/2;

end
X=X1;
Y=Y1;
Xend=[Xend;X];Yend=[Yend;Y];
cof=3;
else
    flag=0;
end
if con>=30
    flag=0;
end
else
    cof=cof+1;
    if cof==3
        flag=0;
    end
end
end

end

figure(1)
hold on
plot(Yend,Xend,'o-g')
con

%% используемые функции
% function S=ShabRound(r) %r=0..1
% %% функция для шаблона круга с градиентом
% % чем меньше r тем меньше круг, r - соотношение удаленности и максимальной
удаленности
% end

```

```

% function S=Trass(I,X,Y,x,y,ri)
% %% функция для зарисовки в невидимую область всего за объектом (функция
определения зон с вероятностью объекта в них)
% % поступает карта с кругами, положение камеры, положение центра кругов и их
радиус
% end

% function [X1 Y1] = GoodWay_Roy( I,X,Y)
%роевой алгоритм пчелинной колонии для поиска следующей точки на карте I
% function [X1 Y1] = GoodWay_Pole( I,X,Y)
%роевой алгоритм пчелинной колонии для поиска следующей точки на карте I
% function [X1 Y1] = GoodWay( I,X,Y)
% алгоритм перебора для поиска следующей точки на карте I

I=imread('D:\BMSTU\I.jpg'); %сам кадр
I_gl=imread('D:\BMSTU\I_gl.jpg'); %глубина
I_r=imread('D:\BMSTU\I_r.jpg'); %кадр с сегментацией
load A_ob;
%кадр с рамкой.
I_r(1:10,:,:)=[];
I_r(:,1:10,:,:)=[];
I_r(484:493,:,:)=[];
I_r(:,647:656,:,:)=[];

%image(I_r);

a=A_ob;

[X Y ~]=size(I);

[X_gl Y_gl ~]=size(I_gl);
if X_gl~=X
    I=imresize(I,[X_gl Y_gl]);
    [X Y ~]=size(I);
    I_r=imresize(I_r,[X_gl Y_gl]);
    I_r=rgb2ind(I_r,3);
end

A=a.*[X Y X Y]; %вершины области с объектом-целью для I

cv=[];co=[];
R=[];
N=0;
for i=1:Y
    Ix=I_r(:,i);
    r=length(find(Ix==1));
    R(i)=round(0.5*0.1/(0.1-r/(X_gl)*2*0.1)*100);

    if length(find(Ix==2))
        cv(i)=1;
    end
    if length(find(Ix==0))
        co(i)=1;
    end

    if length(find(Ix==0))==0
        N=max(N,R(i));
    end
end

```

```

end
R(R>N)=N;
R(abs(R-N)<N/20)=N;
co(R>=N)=0;
cv(R>=N)=1;

%figure(2); plot(R)
H=round(N)+1;
R=H-abs(R-N);
%figure(3); plot(R)

Is=zeros(2*H);
X0=fix(length(Is)/2);
Y0=X0;

Fi=pi/3;
dfi=Fi/(Y-1);
fi=[0:dfi:Fi]-Fi/2;
Xc=[];Yc=[];
for i=1:Y
if co(i)==1
    if R(i)~=N
        [xa ya]=pol2cart(fi(i),R(i));
        x1=round(xa)+X0;y1=round(ya)+Y0;
        x1=length(Is)-x1;
        Is(x1,y1)=1;

        if i<Y
            if R(i+1)~=R(i)
                if R(i)>R(i+1)
                    r1=R(i+1):0.6:R(i);
                else
                    r1=R(i):0.6:R(i+1);
                end
                [xa ya]=pol2cart(fi(i),r1);
                x1=round(xa)+X0;
                y1=round(ya)+Y0;
                x1=length(Is)-x1;
                c=x1+(y1-1)*length(Is);
                Is(c)=1;
            end
        end
    end
end

if i<A(4) %если искомый объект
    Xc=[Xc x1];
    Yc=[Yc y1];
end

% figure(5);
% imshow(Is)
if cv(i)==1
    [xa ya]=pol2cart(fi(i),R(i));
    y1=round(ya)+Y0;
    x1=round(H-N); %разница между H и размером картины
    Is(x1,y1)=1;
end

end

figure(5);

```

```

imshow(Is)
hold on; plot(Yc,Xc,'g')

function S=ShabRound(r) %r=0..1
%% функция для шаблона круга с градиентом
% чем меньше r тем меньше круг, r - соотношение удаленности и максимальной
удаленности
n=30; % 15 n - коэффициент для размера получившегося круга
N=fix(n*r)*2+1;

S=zeros(N);
R=fix(N/2);
[x y]=meshgrid(-fix(n*r):fix(n*r));
[fi,r] = cart2pol(x,y);
d=1-exp(-r/R); %по гауссу
d=flip(d);
S(r<=R)=1-d(r<=R);
if N<=1
    S=ones(3);
end
end

function S=Trass(I,X,Y,x,y,ri)
%% функция определения зон с вероятностью объекта в них
% поступает карта с кругами, положение камеры, положение центра кругов и их
радиус
N=length(I);
m=100; %коэффициент расчета плотности вероятности
R=sqrt(max(X,abs(N-X))^2+max(Y,abs(N-Y))^2); % максимально удаленная точка
I1=I;
%% версия 2.1 рабочая
[fi,r] = cart2pol(x-X,y-Y); %определение координат препятствий относительно
робота
pfi=atan(ri./r'); %угол раскрытия препятствия
dfi=atan(1/R)/2; % максимальный угол для различия двух ближайших клеток на краю
for i=1:length(x) % для каждого препятствия расчет свой
    I=zeros(N); % шаблон для этого расчета
    r0=r(i); % радиус препятствия
    hfi=(fi(i)-pfi(i)):dfi:(fi(i)+pfi(i)); % разбивка углов препятствия
    rr=r0:0.1:R; % разбивка радиусов за препятствием
    [RR, FIFI]=meshgrid(rr,hfi); % составления матрицы углов и радиусов
    [x1, y1]=pol2cart(FIFI,RR); % переход из полярных координат в координаты
декартовы для точек препятствия и зоны за ним
    r1=abs((tan(abs(hfi-fi(i)))*r0)); % расчет расстояний до основного диаметра
препятствия
    f=(RR-r0)./m+repmat( (I1(x(i),y(i))*(1-r1/ri(i)))',1,length(rr)); % расчет
вероятностей существования

    x1=reshape(x1,[],1); % переразбивка координат и вероятностей в вектор
    y1=reshape(y1,[],1);
    f=reshape(f,[],1);
    x1=round(x1)+X;y1=round(y1)+Y; % перенос относительно точки X,Y
    f( y1<=0 ) =[]; % убиение выходящих за пределы координат
    x1( y1<=0 ) =[];
    y1( y1<=0 ) =[];
    f( x1<=0 ) =[];
    y1( x1<=0 ) =[];
    x1( x1<=0 ) =[];
    f( y1>N ) =[];

```

```

x1( y1>N ) =[];
y1( y1>N ) =[];
f( x1>N ) =[];
y1( x1>N ) =[];
x1( x1>N ) =[];

XY=[x1 y1]; % выбивание уникальных номеров
[C, c]=unique(XY,'rows');

if c
    XY=XY(c,:);
    f=f(c);
    c=XY(:,1)+( XY(:,2)-1 )*N; % пересчет в номера элементов матрицы
    I(c)=f; % создание шаблона видимости для препятствия
end
I1=max(I1,I); % объединение шаблонов на основной карте
end
S=I1; % результат
End

function [X1 Y1] = GoodWay( I,X,Y )
%% функция выбора направления движения по картинке с вероятностями.
% возвращает следующую точку для осмотра
N=length(I);
R=sqrt(N^2+(N/2)^2);
I0=I;

%% развертка
dfi=atan(1/R); % максимальный угол для различия двух ближайших клеток на краю
r=0:0.33:R;
count=0;

% fi=pi/3;
% PFI=PFI+pi/2;
% fi=PFI-fi:dfi:PFI+fi;
fi=0:dfi:2*pi;
cofi=0.01+sin((0:0.01*dfi:2*pi)/2);%[ 0.01:0.01/round(length(fi)/2-1):1 1:(-
0.01/round(length(fi)/2-1)):0.01];
R1=ones( [5*round(R) length(fi)] );
Nn=round((N-X)*cofi);
for hfi=fi
    count=count+1;
    [x1 y1]=pol2cart(hfi,r);
    x1=round(x1)+X;y1=round(y1)+Y;
    x1( y1<=0 ) =[];
    y1( y1<=0 ) =[];
    y1( x1<=0 ) =[];
    x1( x1<=0 ) =[];
    x1( y1>N ) =[];
    y1( y1>N ) =[];
    y1( x1>N ) =[];
    x1( x1>N ) =[];
    XY=[x1' y1'];
    [C c]=unique(XY,'rows');
    if c
        c=C(:,1)+( C(:,2)-1 )*N;
        R1(5*round(R)-length(c)+1:5*round(R),count)=I(c)';
        %R1(1:length(c))=sort(I(c)');
    end
end
end

```

```

X1=0;Y1=0;

%% выбор оптимального пути. по максимальному столбику R1 и ширине этого столбика
% чем дальше от нуля соседние столбики, тем уже может быть наш.
R2=R1;
R2(R2>0)=1;
count=1;
h=[];
for i=1:length(fi)
    a=flip(R2(:,i));
    a(1)=0;
    a=find(a>0);
    if a
        r2(i)=a(1);
    else
        r2(i)=1;
    end
    if i>1 && abs(r2(i)-r2(i-1))>=50
        r2(count:i-1)=min(r2(count:i-1));

        h=[h; i-count r2(i-1)+Nn(count+round((i-count)/2))];
        count=i;
    end
    if (i-count)>=100
        r2(count:i-1)=min(r2(count:i-1));

        h=[h; i-count r2(i-1)+Nn(count+round((i-count)/2))];
        count=i;
    end
end

end
r2(count:i)=min(r2(count:i));
h=[h; i-count r2(i-1)+Nn(count+round((i-count)/2))];
if count>1
    if h(1,1)==1
        h(2,1)=h(2,1)+1;
        h(1,:)=[];
    end
    if find(h(:,1)==1)
        k=find(h(:,1)==1);
        for i=length(k):-1:1
            h(k(i)-1,1)=h(k(i)-1,1)+h(k(i),1); %против выбросов
        end
        h(k,:)=[];
        if h(1,1)==2
            h(2,1)=h(2,1)+2;
            h(1,:)=[];
        end
        if find(h(:,1)==2)
            k=find(h(:,1)==2);
            for i=length(k):-1:1
                h(k(i)-1,1)=h(k(i)-1,1)+h(k(i),1); %против выбросов
            end
            h(k,:)=[];
        end
    end
end
h=[h h(:,2).*h(:,1) h*0 h*0 h*0];
h(1,4)=fi(round(h(1,1)/2));
for i=2:length(h(:,1))
    h(i,4)=fi(sum(h(1:i-1,1))+round(h(i,1)/2));
    h(i,5)=cofi(sum(h(1:i-1,1))+round(h(i,1)/2));
    h(i,6)=sum(h(1:i-1,1))+round(h(i,1)/2);
end

```



```

end
h(:,3)=h(:,3).*h(:,5);
h( h(:,2)<5,3)=1;
n=find(h(:,3)==max(h(:,3)));

h=sortrows(h,3,'descend');

n=1:size(h,1);
phi=h(n,4);
Rphi=h(n,2)*2/3;
[x1 y1]=pol2cart(phi,Rphi);
X1=X+round(x1);Y1=Y+round(y1);
else
    X1=X-round(3/4*min(r2));Y1=Y;
End

```

```

function [X1 Y1] = GoodWay_Roy( I,X,Y)
%роевой алгоритм пчелинной колонии для поиска следующей точки на карте I

```

```

N=length(I);
R=sqrt(N^2+(N/2)^2);
I0=I;

n=15;%число разведчиков
m=15;%число рабочих

fi=rand([1 n])*2*pi;
dfi=pi/6;
r=0:0.33:R;
maxR_o=0;
flag=1;count=0;
while flag==1 && count<100
count=count+1;
    %разведчики
    for i=1:length(fi)
[x1 y1]=pol2cart(fi(i),r);
x1=round(x1)+X;y1=round(y1)+Y;
x1( y1<=0 )=[];
y1( y1<=0 )=[];
y1( x1<=0 )=[];
x1( x1<=0 )=[];
x1( y1>N )=[];
y1( y1>N )=[];
y1( x1>N )=[];
x1( x1>N )=[];
XY=[x1' y1'];
[C c]=unique(XY,'rows');
if c
    c=C(:,1)+( C(:,2)-1 )*N;
    R1=I(c)';
    R1=flip(R1);
    R1(1)=0;
    R1=find(R1>0);

    if R1
        r1(i)=r(R1(1));
    else
        r1(i)=0;
    end
end
end

```

```

%рабочие
fi2=rand([1 m])*dfi/2-dfi+fi(i);
for j=1:length(fi2)
    [x1 y1]=pol2cart(fi2(j),r);
    x1=round(x1)+X;y1=round(y1)+Y;
    x1( y1<=0 )=[];
    y1( y1<=0 )=[];
    y1( x1<=0 )=[];
    x1( x1<=0 )=[];
    x1( y1>N )=[];
    y1( y1>N )=[];
    y1( x1>N )=[];
    x1( x1>N )=[];
    XY=[x1' y1'];
    [C c]=unique(XY,'rows');
    if c
        c=C(:,1)+( C(:,2)-1 )*N;
        R1=I(c)';
        R1=flip(R1);
        R1=find(R1==0);
        if R1
            if r(R1(1))*(1+sin(fi2(j)))>r1(i)*(1+sin(fi(i)))
                r1(i)=r(R1(1));
                fi(i)=fi2(j);
            end
        end
    end
end
end

end

maxR_n=max(r1.*(1+sin(fi)));
if abs(maxR_n-maxR_o)<2
    flag=0;
    RR=max(r1);
    FI=fi(find(r1==max(r1)));
else
    maxR_o=maxR_n;
end
end

An=[fi' r1' (r1.*(1+sin(fi)))'];
An=sortrows(An,3,'descend');
pfi=An(:,1);
Rpfi=An(:,2);
pfi(Rpfi<1)=[];
Rpfi(Rpfi<1)=[];

[x1 y1]=pol2cart(pfi,Rpfi*3/4);
X1=X+round(x1);Y1=Y+round(y1);

end

function [X1 Y1] = GoodWay_pole( I,X,Y)
[x y]=meshgrid(1:length(I));
x=x-X;
y=y-Y;
[ Fit Rt]=cart2pol(y,x); %радиус до каждой точки

```

```
Ia=Rt.*I; %потенциал вместе с расстоянием
Ia(Rt<10)=0;
```

```
%% рой
%% по поиску минимума суммы движения
N=length(Ia);
R=sqrt(N^2+(N/2)^2);
I0=Ia;
n=15;%число разведчиков
m=15;%число рабочих
```

```
fi=rand([1 n])*2*pi;
dfi=pi/12;
dr=20;
```

```
M=sum(sum(Ia>0))+1;
flag=1;count=0; minR_=0; co=0;
while flag==1 && count<100
count=count+1;
```

```
    %разведчики
    for i=1:length(fi)
[x1 y1]=pol2cart(fi(i),0:0.33:R);
x1=round(x1)+X;y1=round(y1)+Y;
x1( y1<=0 )=[];
y1( y1<=0 )=[];
y1( x1<=0 )=[];
x1( x1<=0 )=[];
x1( y1>N )=[];
y1( y1>N )=[];
y1( x1>N )=[];
x1( x1>N )=[];
```

```
R1(i)=M;
if x1
    cx=x1+( y1-1 )*N;
    Rr=cumsum(Ia(cx));
    k=find(Rr==min(Rr));
    k=k(end);
```

```
    if i==1
        min(Rr)
    end
    x=x1(k)-X;y=y1(k)-Y;
    R1(i)=sqrt(x^2+y^2);
    K1(i)=sign(min(Rr));
```

```
End
```

```
%рабочие
fi2=rand([1 m])*dfi/2-dfi+fi(i);
R2=[];
for j=1:length(fi2)
    [x1 y1]=pol2cart(fi2(j),0:0.33:R);
    x1=round(x1)+X;y1=round(y1)+Y;
    x1( y1<=0 )=[];
    y1( y1<=0 )=[];
    y1( x1<=0 )=[];
    x1( x1<=0 )=[];
    x1( y1>N )=[];
    y1( y1>N )=[];
    y1( x1>N )=[];
    x1( x1>N )=[];
```

```

R2(j)=M;
if x1
    cx=x1+( y1-1 ) *N;
    Rr=cumsum(Ia(cx));
    k=find(Rr==min(Rr));
    k=k(end);
    x=x1(k)-X;y=y1(k)-Y;
    R2(j)=sqrt(x^2+y^2);
    K2(j)=sign(min(Rr));
end
if R2(j)<M
    if R2(j)>R1(i) && K2(j)<=K1(i)

        fi(i)=fi2(j);
        R1(i)=R2(j);
        K1(i)=K2(j)
    end
end

end

end

if abs(minR_-min(R1.*K1))<2
    co=co+1;
    if co==3
        flag=0;
    end
elseif minR_>min(R1.*K1)
    minR_=min(R1.*K1);
end

end

An=[fi' R1' K1'];
An=sortrows(An,[3 2],{'ascend','descend'});
pfi=An(:,1);
Rpfi=An(:,2);
pfi(Rpfi<1)=[];
Rpfi(Rpfi<1)=[];
[x1 y1]=pol2cart(pfi,Rpfi*7/9);
X1=X+round(x1);Y1=Y+round(y1);

end

```

Приложение Ж Процесс поиска пути при полном переборе

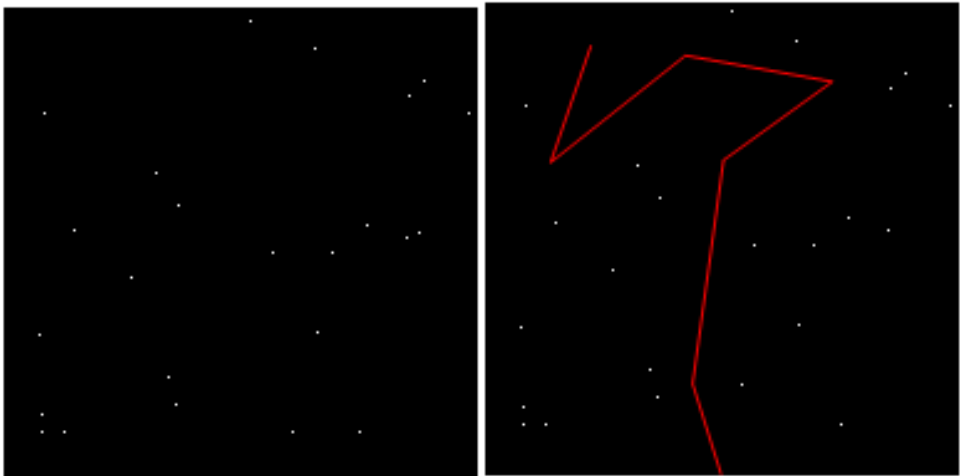
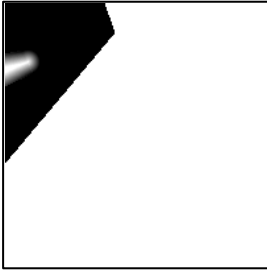
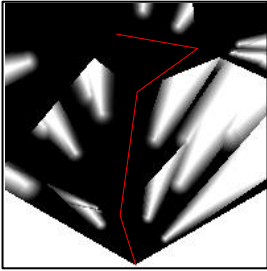
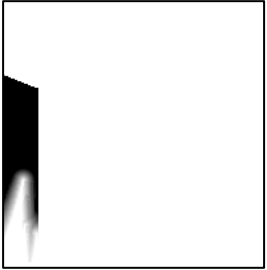
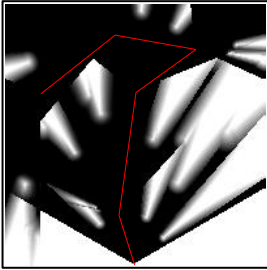
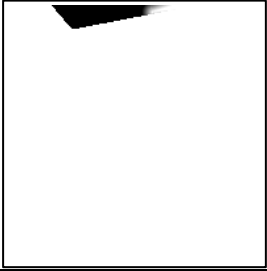
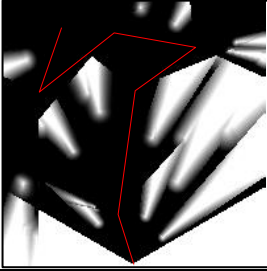


Рисунок Ж.1 – Сцена с препятствиями и путь
Таблица Ж.1 – Область зрения камеры, наложение на предыдущие шаги и путь

Шаг	Область зрения	Сопоставление
i=0		
i=1		
i=2		
i=3		

Шаг	Область зрения	Сопоставление
i=4		
i=5		
i=6		

Приложение 3. Процесс поиска пути при роевом алгоритме

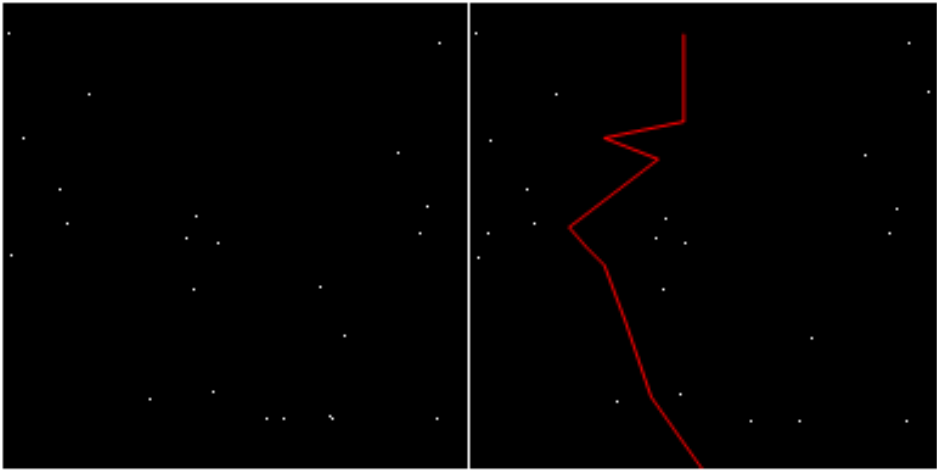
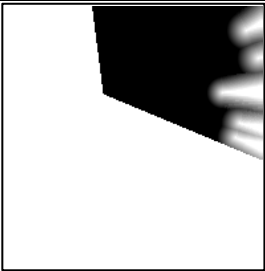
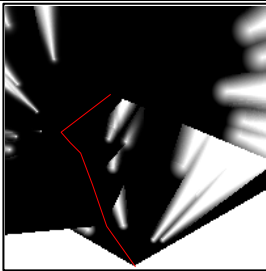
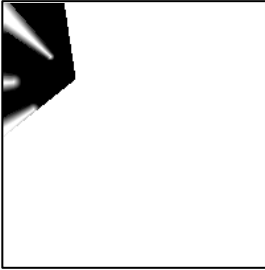
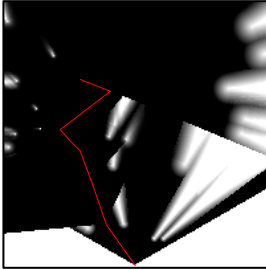
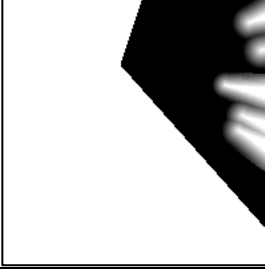
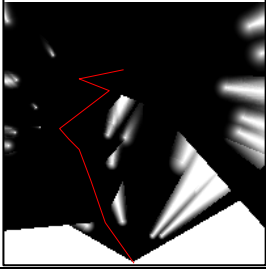
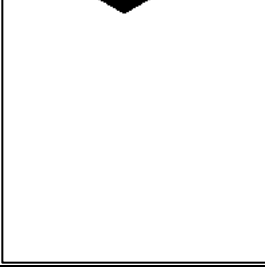
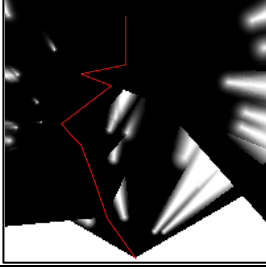


Рисунок 3.1 – Сцена с препятствиями и путь

Таблица 3.1 – Область зрения камеры, наложение на предыдущие шаги и путь

Шаг	Область зрения	Сопоставление
i=0		
i=1		
i=2		
i=3		

Шаг	Область зрения	Сопоставление
i=4		
i=5		
i=6		
i=7		

Приложение И. Процесс поиска пути при методе потенциалов

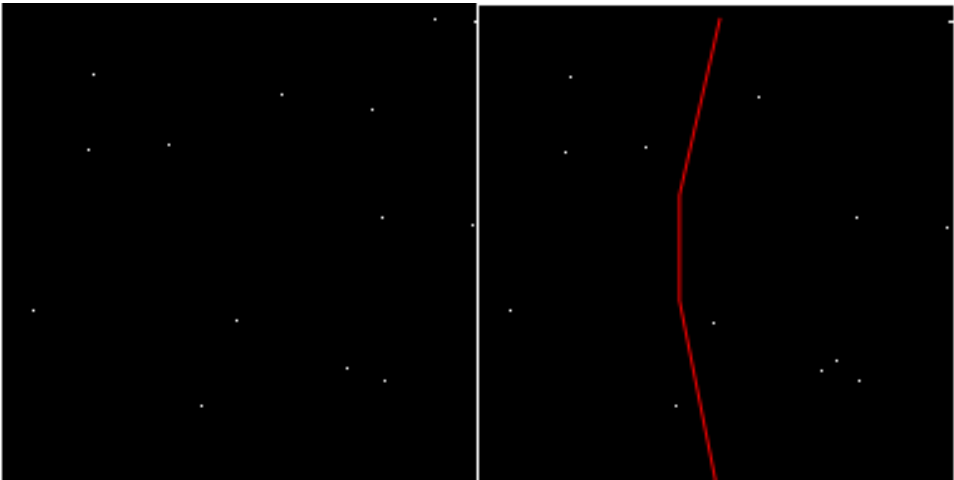


Рисунок И.1 – Сцена с препятствиями и путь

Таблица И.1 – Область зрения камеры, наложение на предыдущие шаги и путь

Шаг	Область зрения	Сопоставление
i=0		
i=1		
i=2		
i=3		