МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт машиностроения
(наименование института полностью)

Кафедра «Промышленная электроника»
(наименование)

11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки / специальности)

Электроника и робототехника
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Система технического зрения для мобильного робота

Обучающийся П.С. Микряков
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н., Е.С. Глибин
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н., доцент, О.В. Лебединская
(ученая степень, звание, И.О. Фамилия)

Тольятти 2023

Аннотация

Название выпускной квалификационной работы: «Система технического зрения для мобильного робота».

Пояснительная записка состоит из введения, подтверждающего актуальность выбранной темы, основной части, состоящей из десяти разделов, в которых подробно описан процесс разработки, заключения, отражающего результаты проделанной работы, и списка используемой литературы, включающего источники информации не только на русском, но и на английском языке.

Объем пояснительной записки составляет 78 страниц. Она содержит 36 рисунков, 7 формул и 3 приложения.

Целью работы является разработка системы для распознавания мобильным роботом направляющей линии с помощью компьютерного зрения.

В ходе выполнения данной выпускной квалификационной работы было решено несколько задач:

– была подтверждена актуальность использования систем технического зрения в разработке автоматически управляемых транспортных средств и автономных мобильных роботов в сфере складской логистики;

– были составлены структурная схема, схема электрическая соединений и электронная компонентная база разрабатываемой системы;

– была разработана и подробно описана программа для распознавания ярко-синей направляющей линии с помощью компьютерного зрения, основанная на принципе поиска объекта по цвету через цветовую модель HSV;

– были предложены возможные модификации основной программы, а также рассмотрены их основные достоинства и недостатки;

– был проведен экономический расчет разрабатываемой системы.

Результатом выполнения данной выпускной квалификационной работы стало внедрение разработанной системы в систему управления движением реального мобильного робота в качестве основного алгоритма поиска пути.

Abstract

The title of the graduation work is «Computer vision system for mobile robot».

Explanatory note consists of an introduction, confirming the relevance of the chosen topic, the main part, consisting of ten sections, which detail the development process, the conclusion, reflecting the results of the work, and a list of references, which includes sources of information not only in Russian, but also in English.

The volume of the explanatory note is 78 pages. It contains 36 figures, 7 formulas and 3 appendices.

The purpose of the work is to develop a system for the recognition of the guiding line by a mobile robot using computer vision.

The graduation work may be divided into five logically connected parts.

In the first part, the relevance of using computer vision systems in the development of automatically controlled vehicles and autonomous mobile robots in the field of warehouse logistics was confirmed.

In the second part, the structural diagram, wiring diagram and electronic component base of the system under development were made.

In the third part, the program for bright blue guiding line recognition using computer vision, based on the principle of object searching by color through the HSV color space, was developed and described in detail.

In the fourth part, possible modifications of the main program were proposed, and their main advantages and disadvantages were analyzed.

In the fifth part, the economic calculation of the system under development was carried out.

The result of this graduation work is the implementation of the developed system into the motion control system of a real mobile robot as the main algorithm of pathfinding.

Содержание

Введение

Развитие компьютерных технологий идет огромными темпами. Особенно это касается сфер искусственного интеллекта и робототехники. Многие вещи, которые всего несколько десятилетий назад можно было встретить лишь в научно-фантастической литературе, сегодня стали неотъемлемой частью нашей жизни. Для современного человека нет ничего удивительного в том, что на передовых автоматизированных заводах вместо рабочих трудятся роботы, по дорогам городов ездят беспилотные автомобили, а искусственные нейронные сети создают шедевры, не уступающие работам профессиональных художников [12].

Одним из наиболее актуальных направлений развития искусственного интеллекта в последние годы стало компьютерное зрение. Способность видеть – важнейшее свойство человека. Именно с помощью зрения мы получаем наибольшее количество информации об окружающем мире. Взглянув на какой-либо предмет, мы, почти не задумываясь, можем сказать, что это такое, каковы его приблизительные размеры, форма, цвет. Благодаря накопленному опыту и знаниям мы легко распознаем объекты и отличаем их друг от друга, ориентируемся в пространстве. Со временем возникла идея наделить зрением и искусственный интеллект. Однако несмотря на то, что современные компьютеры уже давно превзошли человека в сфере вычислений и обработки информации, анализ изображений представлял для них довольно трудную задачу. Развитие машинного обучения и искусственных нейронных сетей значительно ускорило процесс обретения роботами способности видеть. На сегодняшний день искусственный интеллект способен распознавать номерные знаки автомобилей, читать штрих-коды на товарах в супермаркете, анализировать записи с камер видеонаблюдения и производить поиск лиц на фотографиях и видеозаписях. Таким образом, компьютерное зрение представляет собой набор методов, позволяющих компьютерам извлекать информацию из изображений [9].

Области применения компьютерного зрения довольно обширны: от промышленных средств мониторинга технических процессов до автономных управляющих систем, принимающих решения на основе анализа полученной видеоинформации. Одним из наиболее перспективных направлений является использование систем компьютерного зрения в автоматизированных логистических системах. Ярким примером этого служит широкое использование крупными современными компаниями транспортных роботов, обеспечивающих перемещение грузов в промышленной среде без непосредственного участия оператора. Данные машины используются на предприятиях для транспортировки сырья – со склада в цеха, заготовок – между производственными этапами, готовой продукции – с производства на склад и со склада на отгрузку. Применение таких транспортных средств позволяет уменьшить расходы на перевозки, связанные с человеческим фактором и потерей времени, повысить безопасность на предприятии, ускорить производственные процессы. Составление маршрута для данных роботов осуществляется с помощью направляющих линий и маркеров. Их обнаружение и использование в качестве субъектов управления и является главной задачей систем компьютерного зрения [13].

В рамках данной выпускной квалификационной работы предполагается разработка программного и аппаратного обеспечения для распознавания мобильным роботом направляющей линии с помощью компьютерного зрения.

1 Актуальность выбранной темы

В настоящее время робототехника получила применение практически во всех сферах деятельности человека. Потребность в использовании роботизированных систем обусловлена необходимостью повышения эффективности производственных процессов и избавления людей от тяжелого и монотонного ручного труда. Добиться этого позволяет автоматизация производства, т.е. передача функций управления предприятием и контроля за изготовлением продукции от человека к машине. Частным проявлением автоматизации является роботизация. Она представляет собой вытеснение людей из производственного процесса с заменой их на роботов. Автоматизация и роботизация уже несколько десятилетий лежат в основе развития промышленности ведущих стран. Крупнейшие предприятия и компании инвестируют огромные средства в разработку новых технологий и оснащают свои заводы и предприятия современными автоматизированными устройствами и роботизированными комплексами, использование которых позволяет значительно увеличить производительность и эффективность, сократить расходы на персонал, оптимизировать логистические и производственные процессы, а также повысить безопасность труда и устранить ошибки, связанные с человеческим фактором.

Ярким примером этой тенденции является активное внедрение автоматизированных систем для работы на складах и в логистических центрах. Среди них наиболее широкое применение получили автоматически управляемые транспортные средства или AGV (Automated Guided Vehicles) и автономные мобильные роботы или AMR (Autonomous Mobile Robots). Главное отличие между ними заключается в способе навигации. AGV передвигаются с помощью навигационных линий и маркеров и всегда следуют по заранее установленным маршрутам, в то время как AMR обладают большей адаптивностью и свободой передвижения и ориентируется

с помощью датчиков, помогающих им строить карту окружающего пространства, определять кратчайший маршрут и корректировать его в случае возникновения препятствий. Существует три основных вида AGV и AMR – это транспортные тележки, вилочные погрузчики и буксирующие устройства. Транспортные тележки представляют собой роботизированные мобильные платформы, которые осуществляют транспортировку поддонов с деталями на производственных объектах. Вилочные погрузчики также выполняют функции по перемещению объектов, однако при этом они могут быть сконфигурированы для работы с тяжелыми грузами, весом до 4 тонн, или операций по вертикальной погрузке на высоту до 8 метров. Буксировщики также предназначены для транспортировки и способны тащить за собой одну или несколько тележек с товарами или деталями, а их грузоподъемность может достигать 8 тонн [7].

Одной из ведущих компаний в области автоматизации и роботизации логистических центров является Amazon Robotics, дочернее предприятие корпорации Amazon. На сегодняшний день на складах компании работает свыше 100 000 роботизированных систем. Самой известной и распространенной моделью является оранжевый робот Kiva, который предназначен для перемещения больших тележек с грузом в складских помещениях (рисунок 1) [2].

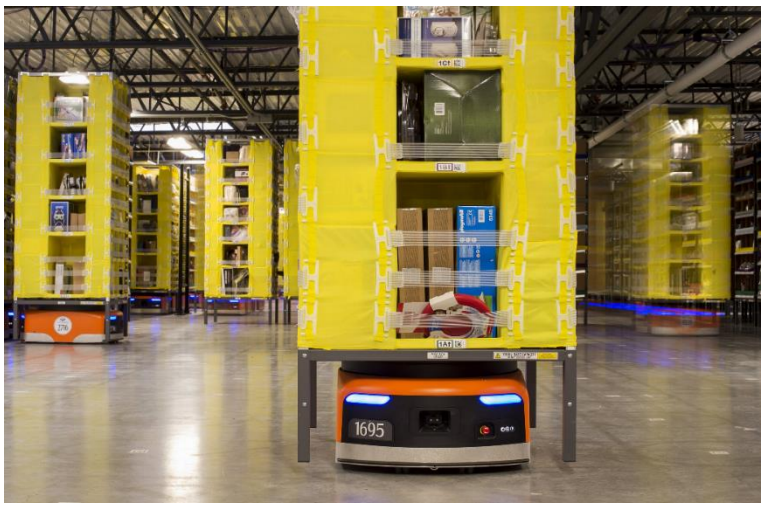

Рисунок 1 – Робот Kiva компании Amazon

Данные AGV оснащены двумя камерами с системами компьютерного зрения: первая расположена на верхней части корпуса и предназначена для считывания штрих-кодов, наклеенных на мобильные стеллажи, для получения информации о хранящихся в них товарах, другая размещена на дне транспортного средства, она позволяет считывать специальные штрих-коды на полу склада, которые отвечают за навигацию робота и формируют его путь из одной точки в другую.

Среди российских разработок стоит отметить универсальные мобильные платформы AGV 1811 от компании INTEC, предназначенные для автоматизации внутрицеховой логистики (рисунок 2). Навигация данных систем основана на использовании компьютерного зрения и контрастных линий, наносимых на напольное покрытие складских помещений и формирующих заранее заданный маршрут движения робота.

Рисунок 2 – AGV 1811 от компании INTEC

Таким образом, в данном разделе была подтверждена актуальность использования систем компьютерного зрения при разработке автоматически

управляемых транспортных средств и автономных мобильных роботов в сфере складской логистики.

2 Выбор компонентов разрабатываемой системы

В качестве основы для реализации подобных систем зачастую используются одноплатные компьютеры (Single-Board Computers), миниатюрные самодостаточные электронно-вычислительные устройства, аппаратная часть которых размещается всего лишь на одной-единственной электронной плате, по размерам примерно сопоставимой с современной банковской пластиковой картой. Таким образом, главным достоинством этих девайсов является компактность и мобильность. Кроме того, подобные микрокомпьютеры обладают неплохой производительностью, если учитывать класс устройства, и могут быть использованы для решения множества самых разных прикладных задач. Также стоит отметить их довольно низкую стоимость и, как следствие, популярность. Еще одноплатные компьютеры обладают возможностью установки на них различных операционных систем, что облегчает их использование.

Для обнаружения направляющей линии используется небольшая веб-камера, осуществляющая захват изображения для последующей его передачи на микрокомпьютер. Далее происходит обработка полученного кадра, и на основе ее результатов вырабатываются управляющие воздействия, передаваемые на микроконтроллер, непосредственно связанный с моторами транспортного средства.

Таким образом, первым этапом разработки системы распознавания мобильным роботом направляющей линии является выбор наиболее подходящей модели одноплатного компьютера.

2.1 Выбор одноплатного компьютера

Самой популярной серией одноплатных компьютеров на сегодняшний день считается Raspberry Pi. Для решения поставленных задач был сделан

выбор в пользу одноплатного компьютера Raspberry Pi 3 Model B. Внешний вид данной модели представлен на рисунке 3.

Рисунок 3 – Raspberry Pi 3 Model B

Среди параметров данного одноплатного компьютера стоит отметить более мощный по сравнению с предыдущей моделью линейки четырехъядерный процессор Cortex-A53, работающий на частоте 1,2 ГГц. Объем оперативной памяти Raspberry Pi 3 Model B составляет 1 Гб. Эта модель содержит четыре порта USB 2.0, порт HDMI для подключения монитора, вывод для Ethernet кабеля, а также встроенные адаптеры Bluetooth и Wi-Fi.

Данный одноплатный компьютер обладает отличной производительностью, что является необходимым условием при работе с алгоритмами компьютерного зрения, содержит достаточное количество USB портов, поддерживает использование беспроводной передачи данных с помощью Bluetooth и Wi-Fi.

2.2 Выбор веб-камеры

Другим важным элементом разрабатываемой системы является веб-камера, которая будет осуществлять захват изображения для последующей

его передачи на Raspberry Pi. Для решения поставленных задач был сделан выбор в пользу небольшой веб-камеры Logitech C270. Внешний вид данной модели представлен на рисунке 4.

Рисунок 4 – Logitech C270

Logitech C270 осуществляет видеозапись в формате высокой четкости (HD) с разрешением 720p (1280×720). Фотографии камера может делать с разрешением 3 Мп. Ее диагональное поле зрения составляет 55º.

Так как камера будет располагаться относительно близко к полу, небольшой угол обзора никак не скажется на работе системы. Кроме того, для обнаружения направляющей линии не требуется большой разрешающей способности устройства, приведенные выше характеристики прекрасно подходят для выполнения поставленной задачи. Также стоит отметить низкую стоимость данной модели, это отличный вариант по соотношению цена-качество.

Таким образом, в данном разделе был произведен выбор одноплатного компьютера и веб-камеры для разрабатываемой системы распознавания мобильным роботом направляющей линии с помощью компьютерного зрения.

3 Составление структурной схемы разрабатываемого устройства

Первым этапом при разработке схемотехники электронного устройства является графическое представление совокупности основных звеньев объекта и связей между ними, т.е. составление его структурной схемы.

Структурная схема разрабатываемой системы для распознавания направляющей линии мобильным роботом с помощью компьютерного зрения представлена на рисунке 5.

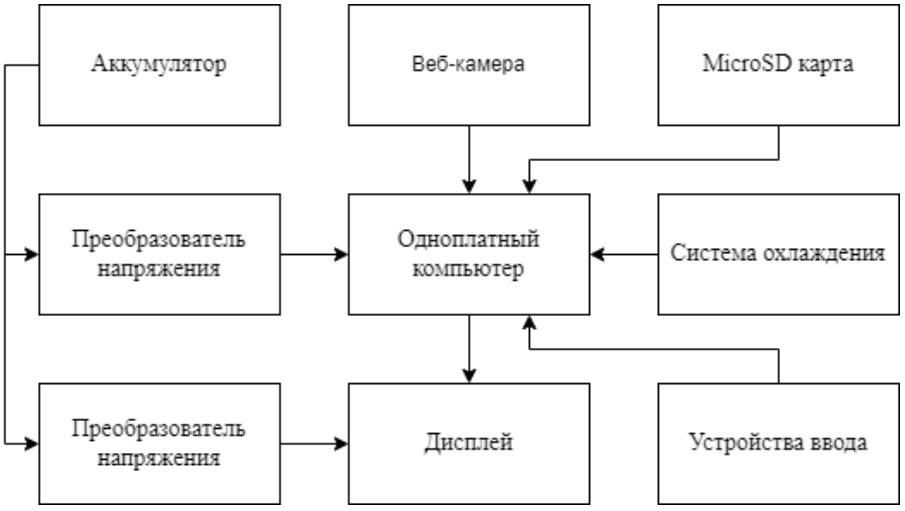

Рисунок 5 – Структурная схема разрабатываемого устройства

Как видно из рисунка, разрабатываемая система состоит из девяти основных элементов:

– одноплатный компьютер Raspberry Pi 3 Model B;

– веб-камера Logitech C270;

– вентилятор, осуществляющий охлаждение микрокомпьютера;

– дисплей;

– microSD карта;

– устройства ввода (клавиатура и компьютерная мышь);

– преобразователь напряжения LM2596S;

– преобразователь напряжения KIS-3R33S;

– аккумулятор.

В качестве источника питания для данной системы используется литиевый аккумулятор с величиной выходного напряжения 11,8 В.

Стоит отметить, что не все элементы разрабатываемой системы могут быть напрямую подключены к данному аккумулятору. Например, для питания одноплатного компьютера и дисплея необходимо напряжение питания 5 В, однако данные устройства не оснащены встроенными преобразователями напряжения.

Чтобы устранить эту проблему, было принято решение использовать для питания от аккумулятора микрокомпьютера регулируемый импульсный понижающий преобразователь напряжения LM2596S (рисунок 6, слева), а для питания дисплея – нерегулируемый преобразователь напряжения KIS-3R33S (рисунок 6, справа).

Рисунок 6 – Преобразователи напряжения LM2596S и KIS-3R33S

В качестве устройства для отображения данных был использован LCD дисплей Waveshare с диагональю 4,3 дюйма и разрешением 480x272 пикселя, разработанный специально для работы с Raspberry Pi (рисунок 7).

Рисунок 7 – Дисплей для вывода информации

Принцип работы разрабатываемой системы состоит в том, что одноплатный компьютер Raspberry Pi будет получать изображение с веб-камеры, обрабатывать полученную информацию и передавать ее на микроконтроллер Arduino, связанный с ним при помощи последовательной связи через USB кабель и управляющий моторами мобильного робота.

Таким образом, в данном разделе была разработана структурная схема системы для распознавания направляющей линии мобильным роботом с помощью компьютерного зрения.

4 Составление электрической схемы разрабатываемого устройства

Для того, чтобы работать с интерфейсом GPIO Raspberry Pi 3 Model B, который предназначен для обеспечения связи одноплатного компьютера с подключаемыми к нему дополнительными компонентами (модулями), необходимо детально разобраться в устройстве и назначении его основных цифровых входов и выходов (пинов). Конфигурация контактов или, иначе говоря, распиновка данного микрокомпьютера представлена на рисунке 8.

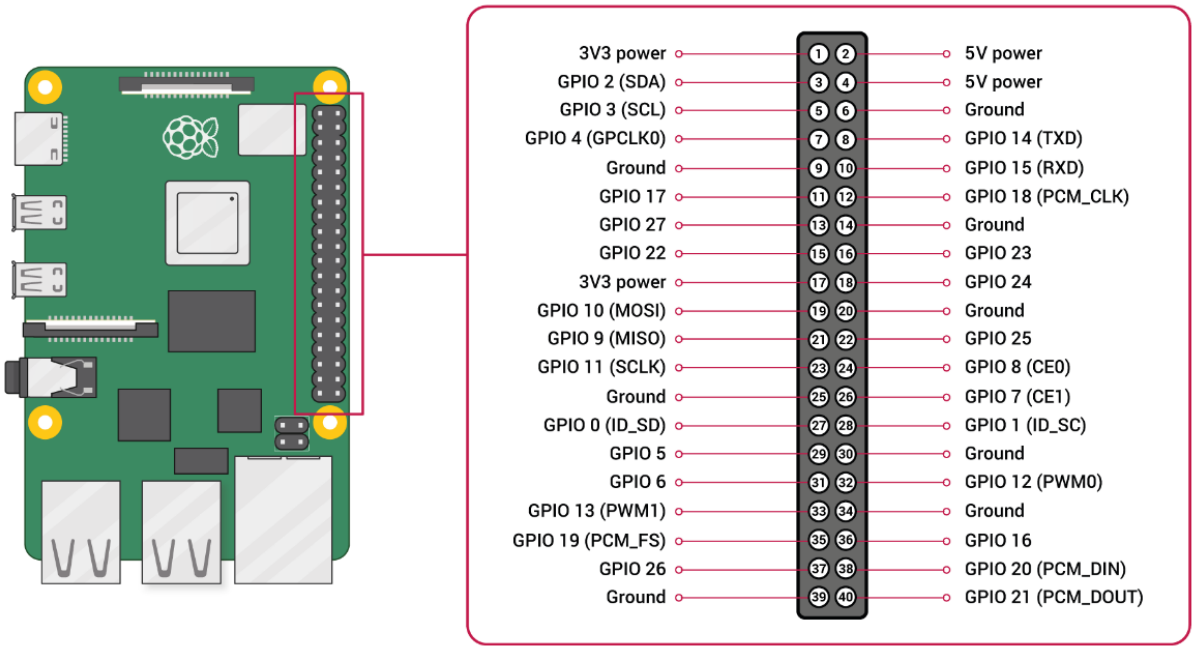

Рисунок 8 – Распиновка Raspberry Pi 3 Model B

Число пинов GPIO Raspberry Pi 3 Model B равняется 40. Они все пронумерованы и делятся на три основные группы:

– питающие (на схемах маркируются Power);

– заземляющие (GND, Ground);

– порты (часто обозначаются как BCM).

Первые необходимы для подачи разных напряжений – 3.3 и 5 В. Вторые обеспечивают безопасность работы платы, отводя электричество. Третьи же

выступают в качестве интерфейсов, способных принимать и отправлять сигналы. Именно к ним пользователь подключает дополнительные модули и периферийные устройства [10].

Также стоит отметить, что заложенная в центральный процессор логическая нумерация пинов отличается от приведенной ранее на схеме физической. Так, не исполняющие функций ввода-вывода «штырьки» одноплатного компьютера номеров не имеют, что следует учитывать при написании кода, поскольку ПО ориентируется именно на логические номера.

4.1 Подключение системы охлаждения

В качестве системы охлаждения для Raspberry Pi будет выступать небольшой вентилятор, расположенный над микрокомпьютером. Его использование в данной системе крайне желательно, так как подобные одноплатные вычислительные устройства имеют свойство довольно быстро нагреваться, что при высоких температурах окружающей среды нередко приводит к выходу их из строя и невозможности их дальнейшего использования. Схема подключения вентилятора к Raspberry Pi 3 Model B представлена на рисунке 9.

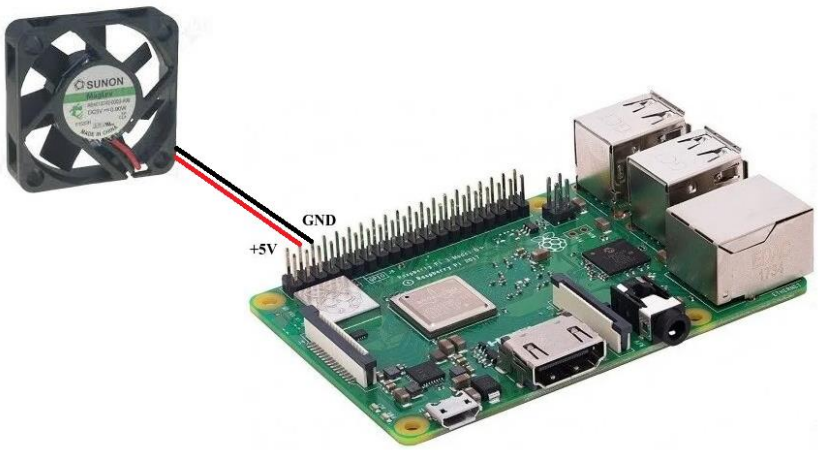

Рисунок 9 – Подключение вентилятора к Raspberry Pi 3 Model B

Как видно из рисунка, вентилятор имеет только два провода: питание и заземление, которые подключаются к пинам Raspberry Pi под номерами 4 и 6 соответственно.

4.2 Подключение Raspberry Pi к Arduino

В качестве устройства, управляющего скоростью и направлением вращения моторов мобильного робота на основании данных, поступивших с веб-камеры и прошедших обработку в одноплатном компьютере Raspberry Pi, выступает микроконтроллер Arduino MEGA 2560. Его связь с микрокомпьютером реализована с помощью UART (Universal Asynchronous Receiver-Transmitter), универсального асинхронного приёмопередатчика, который представляет собой логическую схему, предназначенную для организации обмена информацией с другими цифровыми устройствами. Физическое соединение двух девайсов осуществляется при помощи USB кабеля. Программное же подразумевает написание специального кода для каждого из устройств, а также подключение дополнительных ресурсов, в частности библиотеки WiringPi, обеспечивающей доступ к GPIO-контактам и UART интерфейсу Raspberry Pi для программ, написанных на языке C и C++.

Первым этапом в процессе установления связи между компонентами разрабатываемой системы является подключение библиотеки WiringPi. Однако прежде чем перейти непосредственно к ее установке, нужно скачать все необходимые для этого файлы, воспользовавшись интернет-ресурсами, разместить их на переносном USB-флеш-накопителе, после чего подключить его к одному из USB портов Raspberry Pi и перенести данные на одноплатный компьютер. Так как операционной системой данного устройства является Linux, следующим шагом станет ввод в консоль микрокомпьютера набора определенных команд, перечисленных ниже:

```
cd
tar xfz wiringPi-98bcb20.tar.gz
```

cd wiringPi-98bcb20

./build

После этого автоматически запустится процесс установки скачанной библиотеки. Стоит отметить, что цифры и буквы после записи wiringPi (в данном случае это 98bcb20) являются идентификатором версии библиотеки и на данный момент могут быть неактуальными.

Вторым этапом является определение номера последовательного порта, к которому подключен микроконтроллер Arduino. Для этого можно воспользоваться приведенной консольной командой:

ls /dev/tty*

В результате ее выполнения на экране высветится список всех доступных последовательных устройств (рисунок 10). После этого нужно отключить USB провод, соединяющий Raspberry Pi и Arduino, и вновь ввести данную команду. Отсутствующее в обновленном списке наименование и будет являться искомым номером порта.

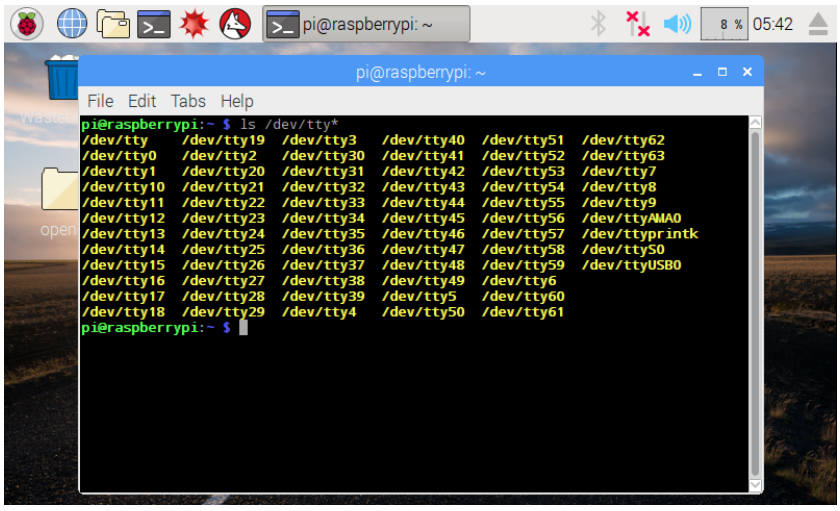

Рисунок 10 – Результат выполнения команды ls /dev/tty*

Третьим и заключительным этапом настройки соединения является написание небольших тестовых программ для каждого из устройств для проверки процесса обмена данными.

Листинг программы для Raspberry Pi на языке C++ приведен ниже:

```
#include <iostream>
#include <wiringPi.h>
#include <wiringSerial.h>
using namespace std;
int main ()
{
	int serialPort = serialOpen("/dev/ttyUSB0", 9600);
	if (serialPort == -1)
	{
		cout << "Failed open port" << endl;
		return 0;
	}
	while (true)
	{
		serialPutchar(serialPort, 'f');
		cout << "f" << endl;
	}
	serialClose(serialPort);
	return 0;
}
```

Разберем данную программу более подробно. В первую очередь в ней происходит подключение заголовочных файлов, необходимых для работы кода. Библиотека iostream является частью стандартной библиотеки C++ и отвечает за ввод информации с клавиатуры и вывод данных в консоль. Библиотека wiringPi дает возможность обращаться к GPIO интерфейсу Raspberry Pi, а библиотека wiringSerial предоставляет инструменты для работы с последовательным портом микрокомпьютера.

Далее внутри основной функции main происходит инициализация последовательного порта, к которому подключен микроконтроллер Arduino, с

помощью команды serialOpen(), принимающей в качестве аргументов номер порта, определенный ранее, и скорость передачи данных. В данной программе было решено использовать стандартную скорость передачи данных в 9600 бод. Это число определяет количество двоичных битов, отправляемых по последовательной линии в секунду. Контроль за состоянием порта будет осуществляться с помощью переменной serialPort типа int [20].

Ниже происходит проверка успешности открытия последовательного порта с помощью конструкции if. Если переменная serialPort содержит значение –1, то это означает, что связь между устройствами не была установлена, вследствие чего в консоль выведется сообщение об ошибке, а программа завершит свою работу.

Если же подключение Raspberry Pi к Arduino прошло успешно, программа попадает в бесконечный цикл и начинает отправлять в микроконтроллер и выводить в консоль управляющий символ «f», означающий, что линия распознана, а мобильный робот должен двигаться прямо. Соответственно обозначены и другие команды: повернуть налево – «l», повернуть направо – «r». Стоит отметить, что использование единственного символа «f» вместо целого слова «forward» повышает быстродействие устройства.

Выход из бесконечного цикла означает остановку мобильного робота, закрытие последовательного порта посредством команды serialClose() и завершение работы всей программы вследствие какого-либо события. Это, например, может быть потеря направляющей линии, нажатие определенной кнопки пользователем или появление в зоне видимости камеры красного «стоп» флажка. В данном тестовом примере такое условие не предусмотрено, т.к. анализ дополнительных переменных усложнил бы и замедлил процесс проверки соединения между Raspberry Pi к Arduino, поэтому выход из цикла осуществляется принудительным закрытием консоли.

Блок-схема алгоритма передачи данных с Raspberry Pi на Arduino представлена на рисунке 11.

Рисунок 11 – Блок-схема алгоритма передачи данных

Далее перейдем к алгоритму работы микроконтроллера. Листинг программы для Arduino приведен ниже:

```
void setup()
{
	Serial.begin(9600);
}
void loop()
{
	if (Serial.available() > 0)
	{
		char getCh = Serial.read();
		Serial.println(getCh);
	}
}
```

Разберем данную программу более подробно. В первую очередь в блоке setup происходит настройка последовательного порта и установление скорости передачи данных в 9600 бод с помощью команды Serial.begin(). Далее в блоке loop осуществляется проверка наличия данных в буфере порта посредством проверки значения, возвращаемого функцией Serial.available(). Если управляющий символ пришел на Arduino, он записывается в переменную getCh типа char с помощью команды Serial.read() для дальнейшего использования. В конце программа выводит полученные данные в монитор последовательного порта посредством функции Serial.println().

Блок-схема алгоритма приема данных на Arduino с Raspberry Pi представлена на рисунке 12.

Рисунок 12 – Блок-схема алгоритма приема данных

После запуска основной программы, представленной в Приложении 1, одноплатный компьютер Raspberry Pi распознает линию, определит направление движения и отправит соответствующий управляющий символ на Arduino. В свою очередь, микроконтроллер подаст напряжение на моторы, и мобильный робот поедет по установленному маршруту.

4.3 Электрическая схема разрабатываемого устройства

На рисунке 13 представлена схема электрическая соединений разрабатываемого устройства.

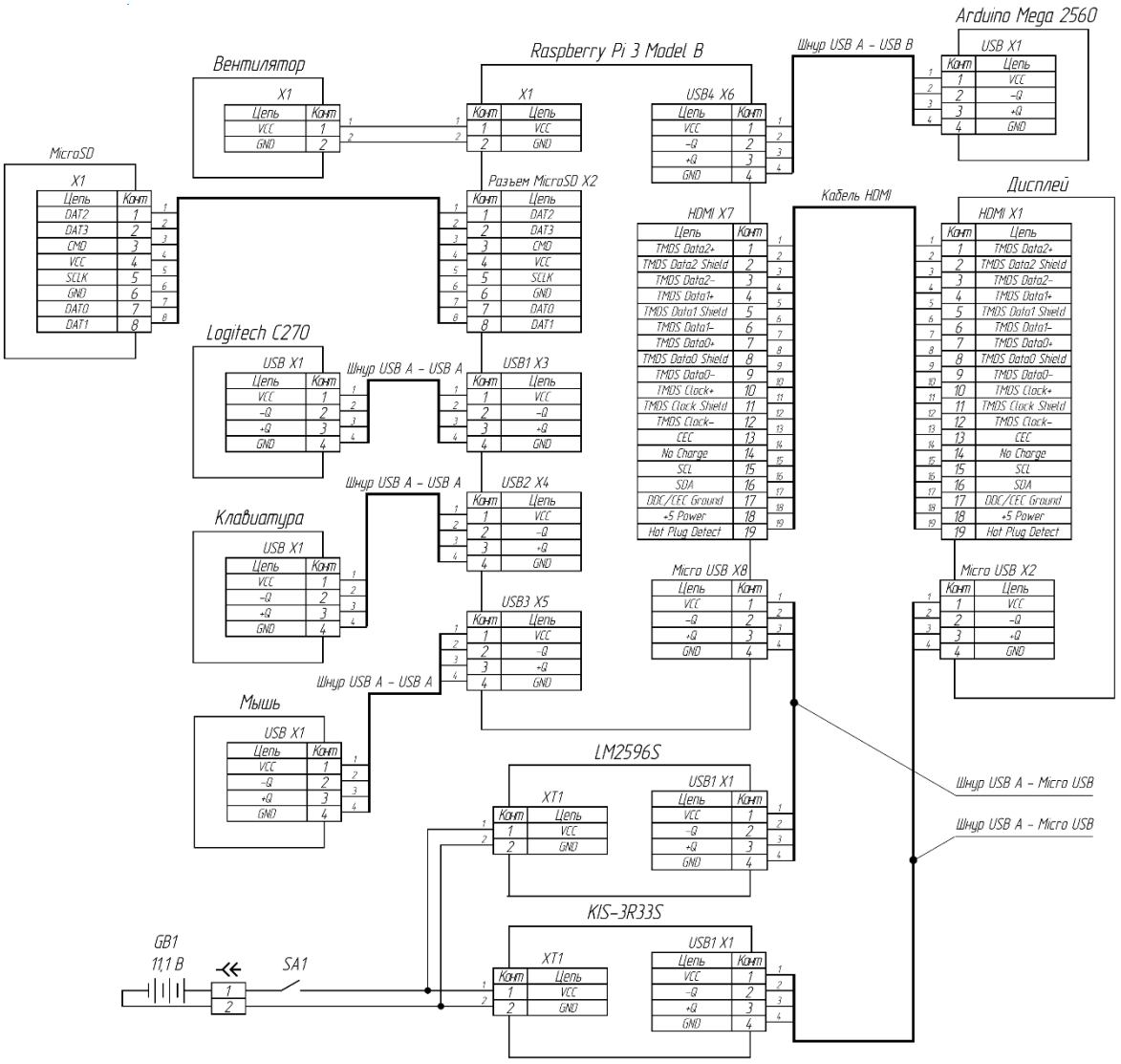

Рисунок 13 – Схема электрическая соединений

В данном разделе была составлена схема электрическая соединений разрабатываемой системы, а также представлены программы и блок-схемы алгоритмов передачи и приема данных по последовательному порту с одноплатного компьютера Raspberry Pi на микроконтроллер Arduino.

5 Выбор инструментов разработки

Для решения поставленных задач данной выпускной квалификационной работы был выбран язык программирования высокого уровня C++, библиотека компьютерного зрения с открытым исходным кодом OpenCV и кроссплатформенная среда разработки Code::Blocks.

5.1 Язык программирования C++

C++ представляет собой высокоуровневый компилируемый язык программирования общего назначения, который подходит для создания самых разных приложений. На сегодняшний день, согласно индексу TIOBE, который оценивает популярность языков программирования, на основе подсчёта результатов поисковых запросов, C++ входит в топ 4 самых распространенных языков в мире вместе с Python, C и Java (рисунок 14).

Apr 2023	Apr 2022	Change	Programming Language	Ratings	Change
1	**1**		**Python**	**14.51%**	**+0.59%**
2	2		C	14.41%	+1.71%
3	3		Java	13.23%	+2.41%
4	4		C++	12.96%	+4.68%
5	5		C#	8.21%	+1.39%
6	6		Visual Basic	4.40%	-1.00%
7	7		JavaScript	2.10%	-0.31%
8	9	^	SQL	1.68%	-0.61%
9	10	^	PHP	1.36%	-0.28%
10	13	^	Go	1.28%	+0.20%

Рисунок 14 – Рейтинг самых популярных языков программирования

Одним из основных преимуществ C++ является его высокая производительность и скорость, которые достигаются благодаря нескольким факторам.

Во-первых, C++ входит в группу компилируемых языков программирования. Программа, написанная на таком языке, сначала преобразуется компилятором в машинный код (компилируется) и записывается в исполняемый файл, который затем уже непосредственно взаимодействует с аппаратной составляющей. Такой подход позволяет добиться более высокой скорости выполнения программы и, следовательно, большей эффективности. Интерпретируемые языки программирования, например Python, работают по другому принципу. Написанные на них программы пропускают этап компиляции и преобразуются в машинный код прямо во время выполнения с помощью другой программы – интерпретатора, что значительно замедляет весь процесс.

Во-вторых, C++ является языком программирования с сильной статической типизацией. Статическая типизация подразумевает, что проверка данных заданным типам выполняется на этапе компиляции. Из этого следует, что для успешного запуска программы программисту нужно еще на этапе написания кода объявлять типы для переменных перед их использованием. Данный подход делает синтаксис более жестким, однако позволяет снизить число ошибок и ускорить выполнение программ за счет того, что компилятор, зная с какими именно типами данных он имеет дело, может проводить дополнительные манипуляции по оптимизации кода. Сильная типизация означает, что язык программирования не позволяет сочетать в выражениях разные типы данных и не осуществляет их автоматические скрытые преобразования. Это вынуждает разработчиков быть более внимательными при написании кода, но в то же время повышает скорость выполнения программ.

В-третьих, C++ унаследовал от языка C богатые возможности по работе с памятью, благодаря которым программисты могут самостоятельно решать, в каких именно ячейках и как долго хранить информацию, и имеют возможность управлять низкоуровневыми ресурсами, такими как регистры процессора, кэш и др. Также в C++ отсутствует автоматическая сборка

мусора. Это позволяет повысить производительность и ускорить выполнение программ, так как на удаление ненужных объектов не тратится время, однако ответственность за эти действия перекладывается с машины на разработчиков, вынужденных лично контролировать выделение и освобождение памяти.

Таким образом, на сегодняшний день C++ является одним из самых производительных и быстрых языков программирования. Доказательством этого служит эксперимент, проведенный португальскими учеными, которые протестировали 27 самых популярных языков программирования путем компиляции/выполнения программ с использованием современных компиляторов, виртуальных машин, интерпретаторов и библиотек и в каждом случае измерили такие параметры, как время выполнения, потребление памяти и энергопотребление. Результаты этого исследования представлены на рисунке 15.

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Рисунок 15 – Результаты исследования энергоэффективности самых популярных языков программирования

Как видно, C++ показывает отличные результаты во всех категориях и входит в топ 5 самых энергоэффективных языков программирования. Это особенно актуально при работе с изображениями и 3D, где требуется высокая скорость работы и производительность. Именно поэтому C++ нередко применяется для создания графических приложений и различных прикладных программ, а также его особенно часто используют для создания компьютерных игр с богатой и насыщенной визуализацией.

Другим значительным преимуществом C++ является его универсальность. Наличие компиляторов, библиотек и инструментов разработки почти под все распространенные операционные системы позволяет с легкостью переносить написанные на этом языке программы с одной платформы на другую. Например, браузер Chrome, написанный преимущественно на языке C++, работает и на Windows, и на Linux, и на macOS.

Также C++ считается отличным фундаментом для изучения программирования. Несмотря на сложный синтаксис, данный язык является эталонным, на его примере были разработаны многие более современные языки программирования, такие как C#, JavaScript, Java, имеющие более простую структуру.

Таким образом, принимая во внимание все вышеперечисленные достоинства языка C++, было принято решение об использовании его в качестве основы для реализации программной части данной выпускной квалификационной работы.

5.2 Библиотека компьютерного зрения OpenCV

На сегодняшний день существует множество приложений, предназначенных для разработчиков и исследователей в области

компьютерного зрения и искусственного интеллекта: AForge.NET, VXL, LTI и др. Но наиболее популярной является библиотека OpenCV [6].

OpenCV (Open Source Computer Vision Library) представляет собой библиотеку компьютерного зрения с открытым исходным кодом. Она реализована на языках программирования C/C++, однако поставляется также и для других языков, таких как Python, Java, Ruby, Matlab, Lua и т.д. OpenCV поддерживает различные операционные системы, к числу которых относятся Windows, Linux, MacOS, iOS, Android и др. Библиотека распространяется бесплатно по лицензии BSD, т.е. ее можно использовать не только в проектах с открытым исходным кодом, но и в закрытых, коммерческих проектах [3].

OpenCV содержит свыше 2500 оптимизированных алгоритмов компьютерного зрения и машинного обучения, позволяющих решать самые разнообразные задачи. Вот некоторые из них:

– обнаружение и распознавание лиц;

– идентификация объектов;

– классификация действий человека на видео;

– отслеживание движущихся объектов;

– распознавание и извлечение 3D-моделей по двумерным изображениям;

– сшивание изображений для получения более высокого разрешения;

– нахождение похожих изображения в базе данных;

– удаление красных глаз с фотографий, сделанных с помощью вспышки.

Благодаря своему простому интерфейсу и обширному функционалу, OpenCV широко используется многими крупными компаниями для разработки своих приложений, а также учеными в научно-исследовательских работах. Например, библиотека применялась для сшивки спутниковых карт, уменьшения шума на медицинских снимках, в беспилотных наземных, воздушных и подводных аппаратах [14].

Общая карта проекта OpenCV представлена на рисунке 16.

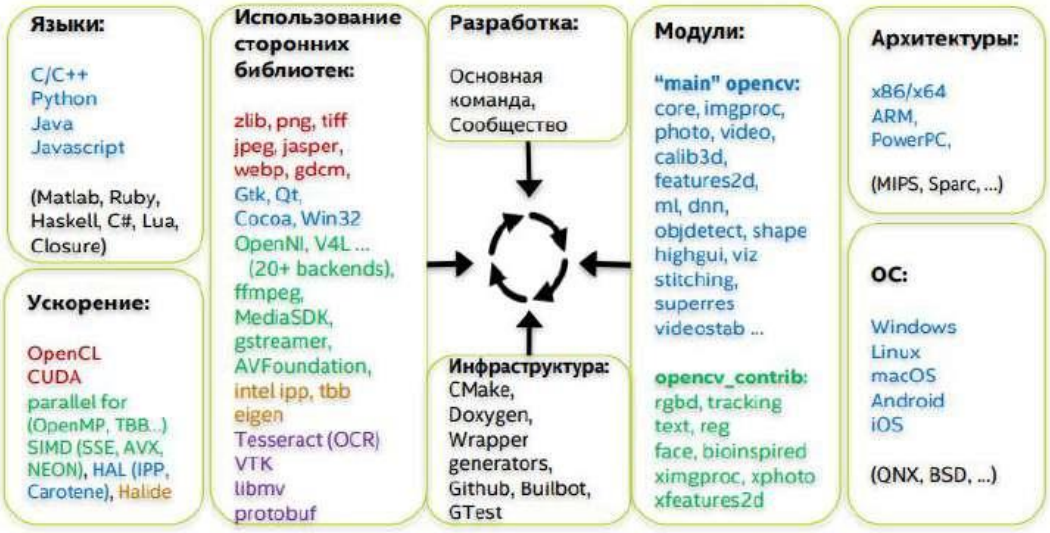

Рисунок 16 – Карта проекта OpenCV

Библиотека OpenCV является структурированной библиотекой. Она разделена на четыре основных компонента:

– Core – модуль, который содержит базовые структуры данных и алгоритмы (базовые операции над многомерными числовыми массивами, матричная алгебра, математические функции, генераторы случайных чисел, базовые функции 2D-графики, запись/восстановление структур данных в/из XML);

– CV – модуль обработки изображений и компьютерного зрения, который позволяет производить базовые операции над изображениями (фильтрация, геометрические преобразования, преобразование цветовых пространств), анализ изображений (выбор отличительных признаков, морфология, поиск контуров), анализ движения, слежение за объектами, обнаружение объектов, в частности лиц;

– HighGUI – модуль для ввода/вывода изображений и видео, создания пользовательского интерфейса (захват видео с камер и из видеофайлов, чтение/запись статических изображений, функции для организации простого UI);

– ML – модуль, реализующий алгоритмы машинного обучения (метод ближайших соседей, наивный байесовский классификатор, деревья решений, бустинг, градиентный бустинг деревьев решений, случайный лес, машина опорных векторов, нейронные сети и др.) [4].

Структура библиотеки OpenCV показана на рисунке 17.

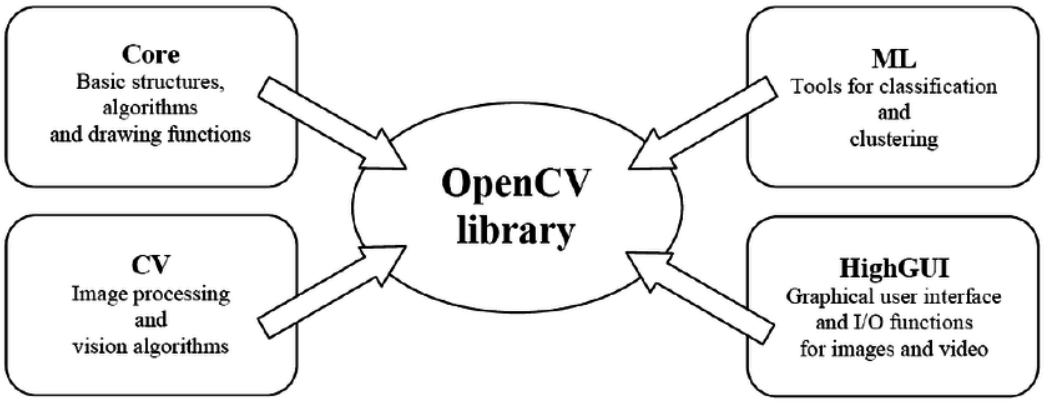

Рисунок 17 – Структура библиотеки OpenCV

Таким образом, простой интерфейс и огромное количеству полезных функций делают OpenCV превосходным инструментом для решения задачи распознавания мобильным роботом направляющей линии с помощью компьютерного зрения.

5.3 Среда разработки Code::Blocks

Code::Blocks представляет собой бесплатную кроссплатформенную интегрированную среду разработки (Integrated Development Environment – IDE) с открытым исходным кодом для программирования на языке C/C++. Данная IDE имеет версии для самых популярных операционных систем, среди которых Windows, Linux, MacOS.

Основным преимуществом Code::Blocks является простой, понятный и удобный пользовательский интерфейс (рисунок 18). Он построен на основе

плавающих и растягивающихся док-панелей, которые можно пристыковывать к любым сторонам главного окна программы, просто перетаскивая их мышкой. Благодаря этой функции можно очень гибко настраивать разные компоновки интерфейса для различных размеров экрана. В случае если мониторов несколько, данная IDE позволяет отделить некоторые панели от основного окна и разместить их на соседних мониторах.

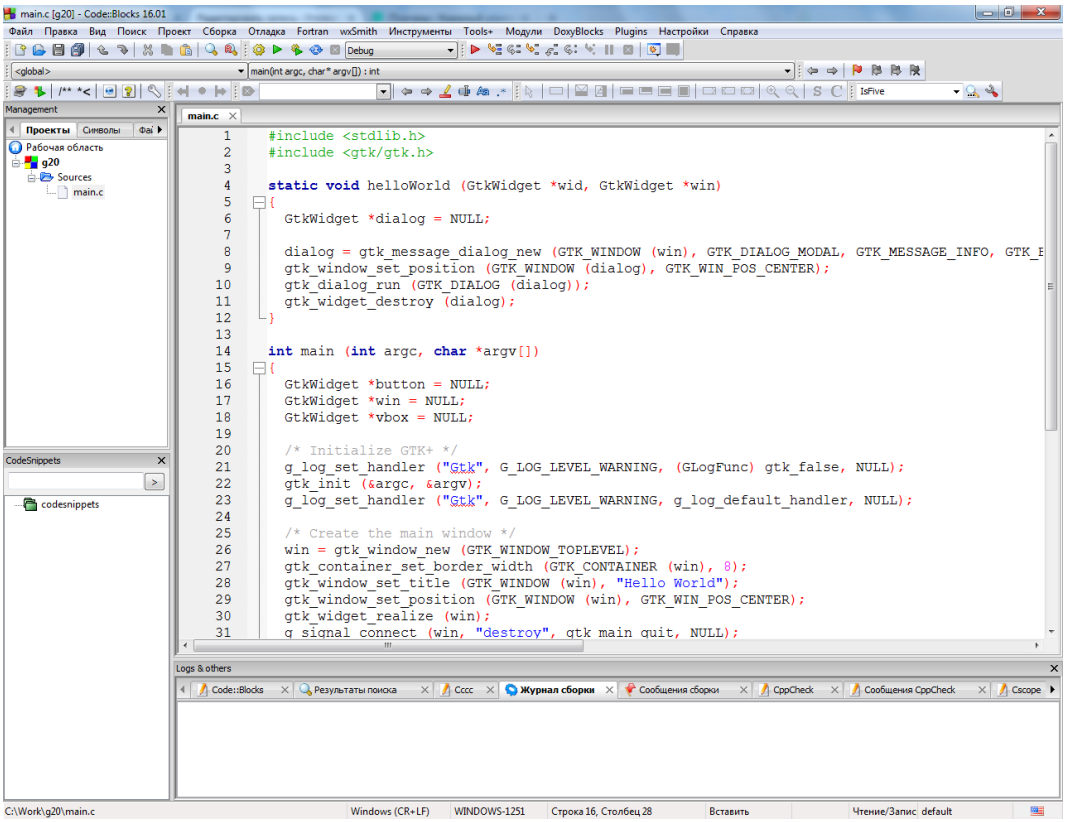

Рисунок 18 – Интерфейс Code::Blocks

Еще одним плюсом Code::Blocks является наличие множества встроенных готовых шаблонов проектов, что очень удобно, т.к. это экономит время пользователя, позволяя ему пропустить этап настройки программы и сразу приступить к написанию кода (рисунок 19) [16].

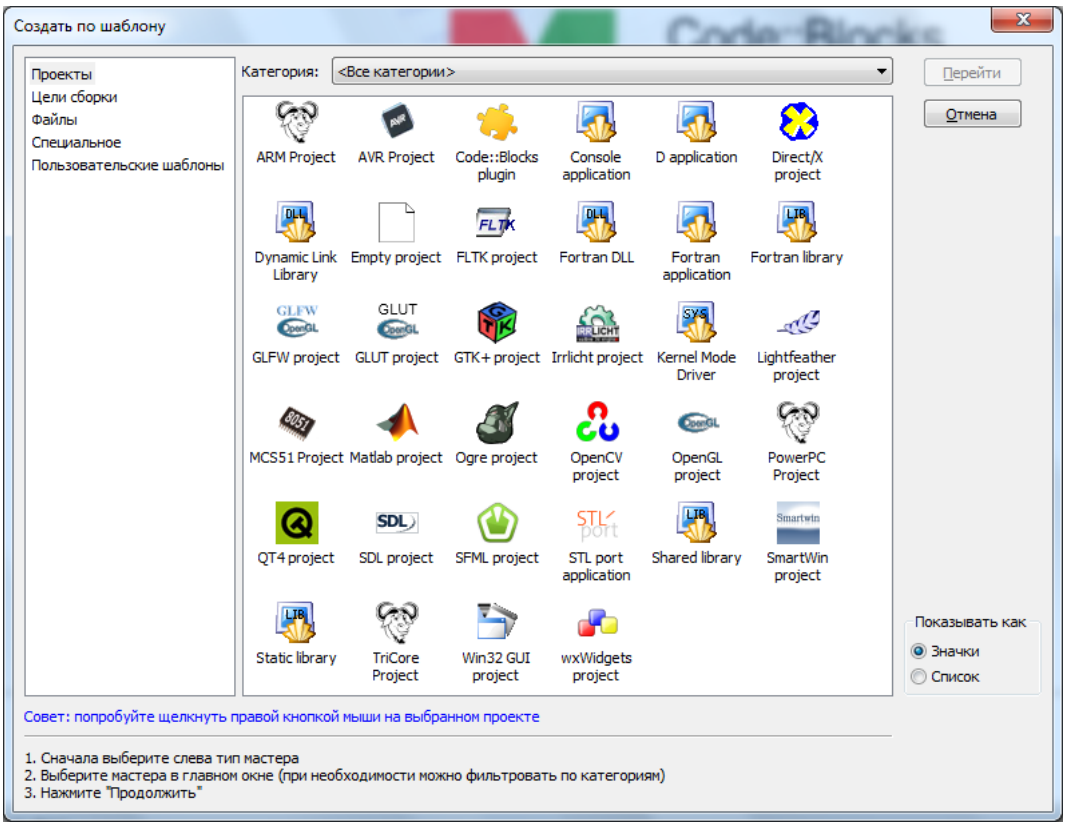

Рисунок 19 – Шаблоны в Code::Blocks

Редактор Code::Blocks обладает множеством функций, упрощающих и ускоряющих процесс написания кода, чтобы создать все необходимые условия для комфортной работы программиста. Вот некоторые из них:

– выделение синтаксиса;

– автодополнение кода;

– браузер классов;

– сворачивание участков кода;

– умный отступ;

– быстрое переключение между .c и .h файлами;

– пользовательские сочетания клавиш.

Среди достоинств Code::Blocks также можно выделить поддержку большого числа компиляторов и возможность легко переключаться между ними в зависимости от ситуации; способность реализовывать параллельную сборку, задействуя при этом несколько ядер процессора; наличие развитых

средств поддержки проектов, включая workspace для объединения нескольких проектов в одном пространстве; оснащение механизмом импорта проектов из других сред разработки, а также экспорта файлов в разнообразные форматы.

Принимая во внимание все вышеперечисленные преимущества интегрированной среды разработки Code::Blocks, было принято решение об использовании ее в качестве основного инструмента для написания кода программы распознавания направляющей линии мобильным роботом с помощью компьютерного зрения.

Таким образом, в данном разделе был произведен выбор языка программирования, библиотеки компьютерного зрения и кроссплатформенной среды разработки для реализации программной части выпускной квалификационной работы.

6 Алгоритм распознавания направляющей линии

В основе алгоритма распознавания направляющей линии мобильным роботом с помощью компьютерного зрения лежит анализ видеоряда, поступающего с веб-камеры на одноплатный компьютер и представляющего собой набор изображений, называемых кадрами. Для извлечения информации о положении робота относительно линии, которая в дальнейшем будет применена для коррекции траектории его движения, используются разнообразные методы обработки изображений. Они позволяют накладывать на кадры фильтры, выделять на них зоны интереса, проводить геометрические преобразования, манипулировать пикселями и т.д. Именно комбинация этих алгоритмов и функций позволяет мобильному роботу «видеть» направляющую линию и следовать ей.

6.1 Цветовая сегментация кадра

Сегментация – это процесс разделения цифрового изображения на несколько сегментов с целью упрощения процесса его анализа. Для реализации цветовой сегментации кадров, поступающих с веб-камеры на одноплатный компьютер, воспользуемся инструментами библиотеки OpenCV для манипуляции цветовыми пространствами.

Цветовое пространство – это модель представления цвета, согласно которой любой оттенок может быть представлен в виде точки, имеющей определенные цветовые координаты. Самым популярным и широко используемым цветовым пространством является модель RGB, в основе которой лежит использование комбинаций трех основных цветов – красного (Red), зеленого (Green) и синего (Blue). Данная модель является аддитивной, т.е. оттенки получаются посредством сложения основных цветов друг с другом. Например, при смешении синего и красного получается пурпурный, зелёного и красного – жёлтый, зелёного и синего – голубой. Комбинация всех

трёх основных излучений даст белый цвет, а отсутствие излучения – черный. Для удобства и наглядности модель RGB часто представляют в виде цветового куба (рисунок 20).

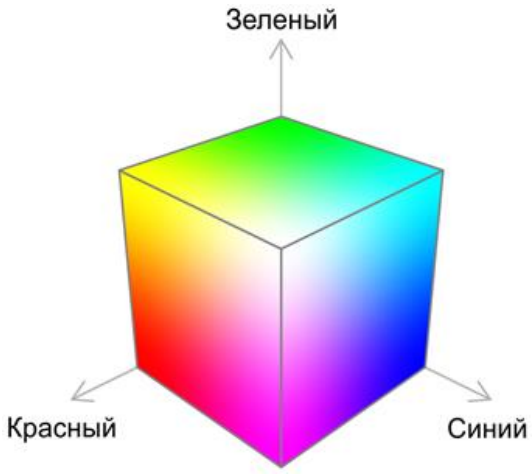

Рисунок 20 – RGB модель в виде цветового куба

Изображения в модели RGB можно интерпретировать, как двумерные массивы или матрицы с тремя слоями (каналами), представляющими красный, зеленый и синий цвета. Каждый элемент двумерного массива каждого канала хранит в себе интенсивность одного из цветов определенного пикселя изображения. Комбинация соответствующих пикселей из каждого канала дает уникальный общий цвет пикселя (рисунок 21) [18].

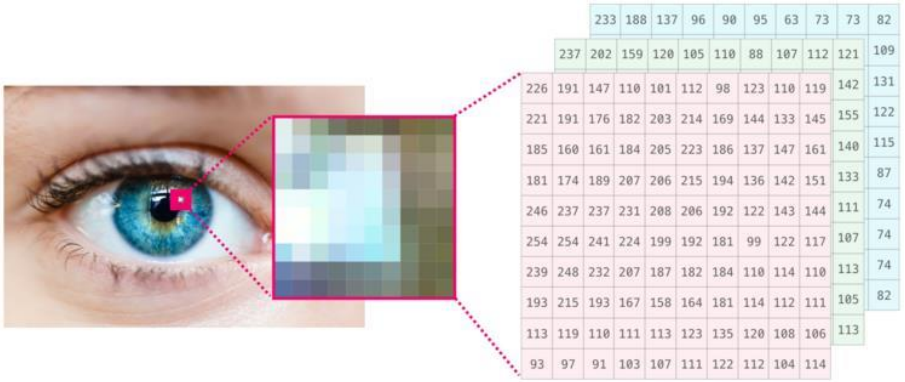

Рисунок 21 – Изображение в виде матрицы

В компьютере каждая из цветовых координат пикселя записывается с помощью одного байта, т.е. в диапазоне чисел от 0 до 255 включительно, где 0 – минимальная, а 255 – максимальная интенсивность. Так, например, красный цвет будет закодирован в виде (255, 0, 0), желтый – (255, 255, 0), оранжевый – (255, 128, 0) и т.д.

Другим часто используемым цветовым пространством является модель HSV, координатами в которой выступают оттенок (Hue), насыщенность (Saturation) и яркость (Value). Для удобства и наглядности данную модель часто представляют в виде цветового конуса (рисунок 22).

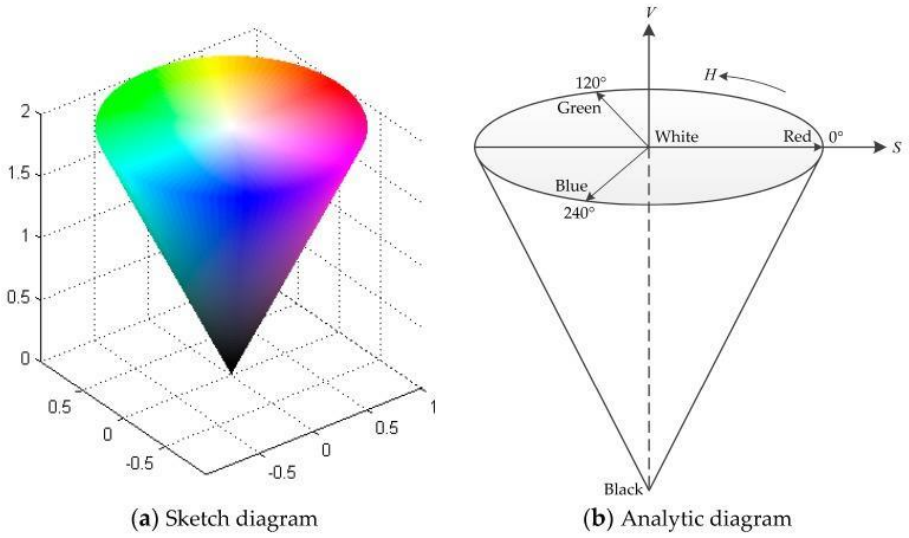

Рисунок 22 – HSV модель в виде цветового конуса

Оттенки в пространстве HSV представлены углами на окружности, которые лежат в диапазоне от 0º до 360º. Красный, зеленый и синий цвета, например, делят основание конуса на три равные части, образуя углы в 0º, 120º и 240º соответственно. Насыщенность варьируется в пределах от 0 до 100 или от 0 до 1 и соответствует диаметру окружности. Чем больше этот параметр, тем «чище» цвет, поэтому его еще иногда называют чистотой цвета. Яркость соответствует высоте конуса и также задается диапазонами от 0 до 100 или от 0 до 1, где 0 представляет черный цвет, а 1 или 100 – белый.

Благодаря тому, что все возможные оттенки представляют собой набор фиксированных значений, а параметры, отвечающие за насыщенность и яркость, независимы друг от друга, цветовое пространство HSV лучше подходит для обнаружения ярко-синей направляющей линии чем RGB, поскольку позволяет более гибко настраивать параметры сегментации изображения и меньше подвержено влиянию освещения [19].

В библиотеке OpenCV за сегментацию изображения по цвету отвечает функция inRange(src, lowerb, upperb, dst), где:

– src – входной кадр, на который накладывается фильтр;

– lowerb – нижняя граница цветового диапазона;

– upperb – верхняя граница цветового диапазона;

– dst – выходной кадр, с примененным фильтром.

Данная функция превращает поступающее ей на вход цветное изображение в бинарное, делая все пиксели, попадающие в указанный цветовой диапазон, белыми, а остальные – черными. Демонстрация работы функции представлена на рисунке 23 [17].

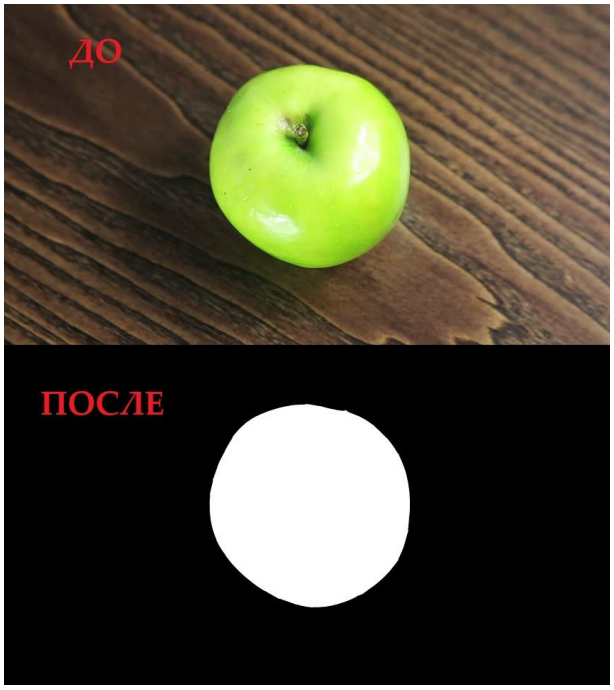

Рисунок 23 – Демонстрация работы функции inRange()

6.2 Выделение контуров

Ещё одним важным инструментом библиотеки компьютерного зрения OpenCV является функция выделения контуров на изображении. Данный алгоритм позволяет анализировать положение объектов в кадре и может быть использован для обнаружения мобильным роботом направляющей линии по ее внешним границам.

В библиотеке OpenCV за нахождение контуров объекта на изображении отвечает функция findContours(image, contours, hierarchy, mode, method), где:

– image – исследуемое изображение;

– contours – список всех найденных контуров, представленных в виде векторов их точек;

– hierarchy – структура, устанавливающая родительско-дочерние отношения между контурами. Например, при обнаружении двух контуров, один из которых расположен внутри другого, внешний контур будет являться родительским, а внутренний – дочерним. Иерархия представлена в виде массива, каждый элемент которого представляет собой еще один массив из четырех индексов (Next, Previous, First_Child, Parent), соответствующий одному из контуров. Индекс Next обозначает следующий контур на том же иерархическом уровне для рассматриваемого контура. Индекс Previous обозначает предыдущий контур на том же иерархическом уровне для рассматриваемого контура. Индекс First_Child обозначает первый дочерний контур для рассматриваемого контура. Индекс Parent обозначает родительский контур для рассматриваемого контура. Если контур, на который ссылается какой-либо из индексов, отсутствует, то соответствующий индекс возвращает значение -1 [8].

Параметр mode содержит один из четырех режимов извлечения найденных контуров:

– CV_RETR_LIST – извлекает все контуры без установления каких-либо иерархических отношений;

– CV_RETR_EXTERNAL – извлекает только внешние контуры;

– CV_RETR_CCOMP – извлекает все контуры и организует их в двухуровневую иерархию. На верхнем уровне находятся внешние границы объектов. На нижнем уровне – границы отверстий. Если внутри отверстия присутствует другой контур, он все равно помещается на верхний уровень;

– CV_RETR_TREE – извлекает все контуры и группирует их в многоуровневую иерархию.

Параметр method содержит один из четырех методов аппроксимации контуров:

– CV_CHAIN_APPROX_NONE – упаковка отсутствует и все контуры хранятся в виде множества отрезков;

– CV_CHAIN_APPROX_SIMPLE – склеивает все горизонтальные, вертикальные и диагональные сегменты, оставляя только их конечные точки;

– CV_CHAIN_APPROX_TC89_L1 – применяет к контурам один из вариантов алгоритма аппроксимации Те-Чина;

– CV_CHAIN_APPROX_TC89_KCOS – также применяет к контурам один из вариантов алгоритма аппроксимации Те-Чина.

Для визуализации контуров на кадре в библиотеке OpenCV предусмотрена функция drawContours(image, contours, contourIdx, color, thickness), где:

– image – изображение, на котором будут нарисованы контуры;

– contours – список контуров для визуализации, представленных в виде векторов их точек;

– contourIdx – параметр, указывающий конкретный контур для отрисовки. Если он равен -1, то будут визуализированы все контуры;

– color – цвет визуализируемых контуров;

– thickness – толщина линий визуализируемых контуров.

Пример нахождения и визуализации контуров на изображении представлен на рисунке 24.

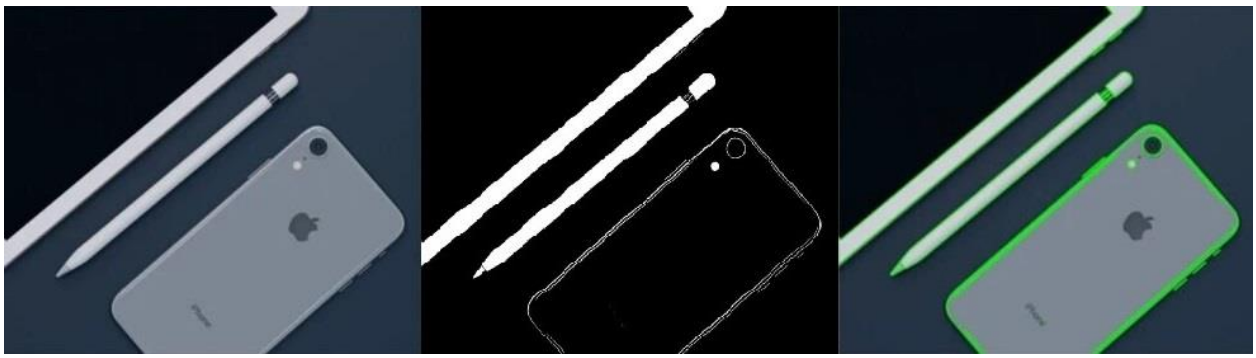

Рисунок 24 – Демонстрация работы функций findContours() и drawContours()

6.3 Моменты изображения

Моменты изображения – это определённые средневзвешенные значения интенсивности пикселей изображения. Они определяются следующей формулой:

$$M_{ij} = \sum_{x}\sum_{y} x^i\, y^j I(x,y), \#(1)$$

где x, y – координаты пикселя;

i, j – мощность, на которой соответствующий пиксель взят в сумме с другими отображенными;

I(x,y) – интенсивность пикселя.

В приведенном выше выражении момент M_{ij} определяется как сумма по всем пикселям объекта значений пикселей в точке x, y, умноженных на коэффициент $x^i y^j$ [5].

В бинарных изображениях интенсивность пикселя $I(x,y)$ может принимать только два значения: ноль для черных сегментов и единица для белых. Таким образом, момент нулевого порядка M_{00} такого изображения будет представлять собой площадь белого объекта (сумму всех его пикселей)[1]. Вычисляется он по следующей формуле:

$$M_{00} = \sum_{x}\sum_{y} I(x,y)\,.\#(2)$$

Моменты M_{10} и M_{01} первого порядка бинарного изображения являются суммами x и y составляющих всех пикселей белого объекта и находятся по формулам (3) и (4) соответственно.

$$M_{10} = \sum_x \sum_y xI(x,y) \,.\#(3)$$

$$M_{01} = \sum_x \sum_y yI(x,y) \,.\#(4)$$

С помощью моментов M_{00}, M_{10} и M_{01} можно найти центроид белого объекта в кадре, которым будет искомая направляющая линия. Его координаты могут быть вычислены с помощью формул (5) и (6) [21].

$$x = \frac{M_{10}}{M_{00}} .\#(5)$$

$$y = \frac{M_{01}}{M_{00}} .\#(6)$$

В библиотеке OpenCV за нахождение моментов изображения отвечает функция moments(array, binaryImage), где:

– array – исследуемое изображение;

– binaryImage – параметр бинарности изображения типа bool. Если он принимает значение true, то все ненулевые пиксели кадра рассматриваются как единицы.

Таким образом, в данном разделе были подробно описаны алгоритмы и функции библиотеки компьютерного зрения OpenCV, необходимые для поэтапного преобразования получаемой с веб-камеры видеоинформации к виду, позволяющему мобильному роботу обнаружить направляющую линию.

7 Обзор существующих решений

Помимо представленного алгоритма распознавания направляющей линии, основанного на принципе поиска объекта по цвету через цветовое пространство HSV с дальнейшим вычислением моментов бинарного изображения, существует множество других способов определения положения мобильного робота в пространстве, использующих иные инструменты компьютерного зрения.

7.1 Решение 1

Ярким примером является статья индийских исследователей из Технологического института Вишвакармы, в которой описывается разработка системы навигации мобильного робота по двум направляющим линиям, имитирующим дорожную разметку [22].

Кратко рассмотрим основные этапы работы их алгоритма.

В основе данного метода распознавания двух направляющих линий лежит преобразование цветного кадра с веб-камеры в оттенки серого. Добиться этого позволяет функция cvtColor(), принимающая в качестве кода преобразования запись COLOR_BGR2GRAY.

Далее к получившемуся кадру применяется размытие по Гауссу, которое призвано сгладить изображение и уменьшить шумы. Принцип действия данного метода заключается в том, что он пересчитывает значение яркости каждого пикселя преобразуемого кадра и устанавливает его равным средневзвешенному значению его окрестности, причем соседние пиксели оказывают все меньшее влияние на исходный пиксель по мере увеличения их расстояния до него. В библиотеке OpenCV за реализацию размытия по Гауссу отвечает функция GaussianBlur(src, dst, ksize, sigma), где:

– src – входной кадр, на который накладывается фильтр;

– dst – выходной кадр, с примененным фильтром;

– ksize – размер ядра, области вокруг пикселя, которая учитывается при вычислении его нового значения;

– sigma – среднеквадратическое отклонение нормального распределения в функции Гаусса, увеличение которого приводит к более сильному размытию изображения и сглаживанию острых краев.

Следующим шагом является выделение границ объектов на изображении с помощью оператора Кэнни. Работа данного метода состоит из 5 этапов:

1. Сглаживание изображения. Прежде чем кадр будет обработан алгоритмом обнаружения границ, к нему автоматически применяется размытие по Гауссу для уменьшения шумов. В данной научной работе этот этап был ранее пройден посредством отдельного вызова функции GaussianBlur().

2. Поиск градиентов изображения. Градиент показывает изменение интенсивности или яркости пикселей кадра в разных направлениях. Для его нахождения используется оператор Собеля, результатом применения которого в каждой точке изображения является вектор градиента в этой точке.

3. Подавление не-максимумов. На данном этапе подавляются все точки, которые не являются локальными максимумами градиента в определенном направлении.

4. Двойная пороговая фильтрация изображения. Здесь применяются специальные пороговые значения для определения того, является ли данный пиксель кадра частью границы объекта или нет. Оператор Кэнни использует два порога фильтрации: если значение пикселя выше верхней границы – он принимает максимальное значение (граница считается достоверной), если ниже – пиксель подавляется. Точки же со значением, попадающим в диапазон между границами, принимают фиксированное среднее значение (они будут уточнены на следующем этапе). Таким образом, чем меньше порог, тем больше границ будет найдено, но тем более восприимчивым к шуму станет результат, выделяя лишние объекты на изображении. Высокий же порог,

наоборот, может проигнорировать некоторые края или выдать неполную границу в виде фрагментов.

5. Трассировка области неоднозначности. Упрощённо, задача сводится к выделению групп пикселей, получивших на предыдущем этапе промежуточное значение, и отнесению их к границе (если они соединены с одной из установленных границ) или их подавлению (в противном случае). Пиксель добавляется к группе, если он соприкасается с ней по одному из 8-ми направлений.

В библиотеке OpenCV за применение оператора Кэнни для нахождения границ объектов на изображении отвечает функция Canny(image, edges, threshold1, threshold2), где:

– image – исследуемое изображение;

– edges – бинарное изображение, на котором останутся только границы объектов;

– threshold1 – нижняя граница фильтрации;

– threshold2 – верхняя граница фильтрации.

После этого на полученном кадре выделяется область интереса, содержащая только дорожное покрытие с направляющими линиями, а все остальное отбрасывается.

Последним шагом в процессе обработки изображения, поступающего с веб-камеры, является использование преобразования Хафа для нахождения направляющих линий. Данный алгоритм основан на представлении прямой в полярных координатах и использует следующее уравнение:

$$r = xcos\theta + ysin\theta, \#(7)$$

где r – длина нормали к прямой, проведенной из начала координат;

θ – угол между этой нормалью и осью абсцисс;

x и y – координаты точки на прямой.

Любой прямой, проходящей через произвольную точку (x, y) в прямоугольной системе координат, соответствует точка (r, θ) в полярной системе координат. Следовательно, множеству прямых, проходящих через

произвольную точку (x, y) в прямоугольной системе координат, соответствует набор точек (r, θ) в полярной системе координат, который, согласно уравнению 7, представляет собой синусоиду, уникальную для данной точки (x, y) и однозначно её определяющую. Таким образом, все возможные прямые, проходящие через все точки изображения, могут быть представлены в виде набора синусоид в полярных координатах. Пересечение же синусоид друг с другом означает, что соответствующие им точки в прямоугольных координатах находятся на одной прямой. Каждая такая прямая в свою очередь записывается в специальный двумерный массив, называемый аккумулятором, строки/столбцы которого соответствуют показателям r и θ. Размер данного массива зависит от выбранного шага для угла θ и максимально возможной длины нормали r, ограниченной диагональю кадра. Из этого всего следует, что чем больше пересечений синусоид, тем больше соответствующих им точек лежат на одних и тех же прямых и тем больше записей в аккумуляторе принадлежат этим прямым. Только при достижении определенного количества записей прямая считается обнаруженной.

В библиотеке OpenCV за использование преобразования Хафа для нахождения прямых на изображении отвечает функция HoughLinesP(image, lines, rho, theta, threshold, minLineLength, maxLineGap), где:

– image – исследуемое изображение;

– lines – массив типа вектор для найденных линий, хранящий координаты конечных точек каждого обнаруженного сегмента линии;

– rho – показатель r, определяющий размер аккумулятора;

– theta – показатель θ, определяющий размер аккумулятора;

– threshold – пороговое значение количества записей в аккумуляторе, необходимое для обнаружения линии;

– minLineLength – минимальная длина линии. Сегменты линии короче этого значения отклоняются;

– maxLineGap – максимально допустимый промежуток между сегментами одной линии для их соединения.

Далее обнаруженные направляющие линии используются в качестве основы для построения вокруг них вращающегося прямоугольника, ориентация которого и формирует управляющее воздействие для корректировки траектории движения мобильного робота.

Результаты всех преобразований и работы алгоритма в целом представлены на рисунке 25.

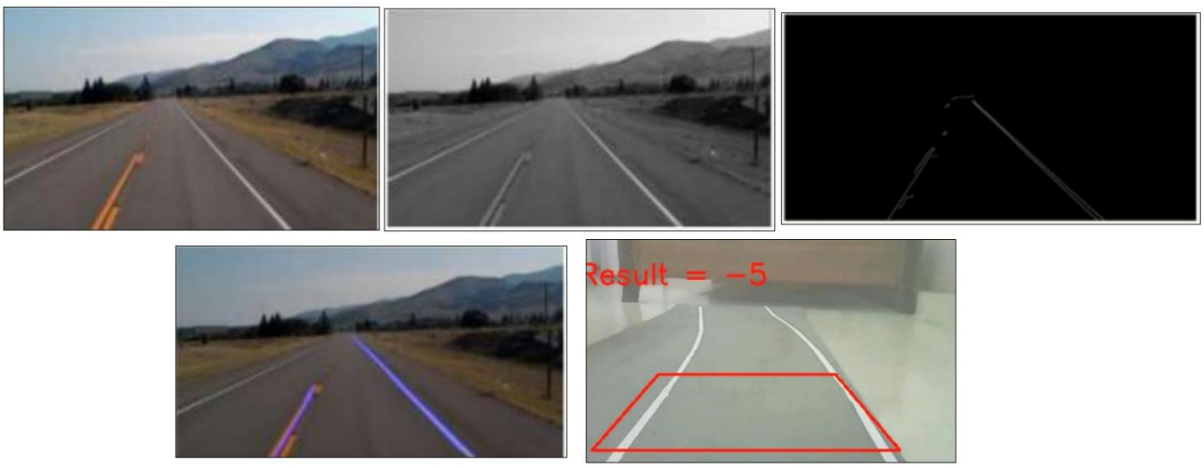

Рисунок 25 – Результаты работы алгоритма индийских ученых

Подобные методы позиционирования нашли широкое применение в современных беспилотных автомобилях.

7.2 Решение 2

Другой интересный способ навигации мобильного робота представлен в работе корейских исследователей из Университета Конджу, который, в отличие от рассмотренного ранее метода индийских ученых, основан на использовании маркеров вместо направляющих линий [15].

Кратко рассмотрим основные этапы работы данного алгоритма.

Для более удобного анализа кадров авторы данной научной работы первым делом программно изменяют перспективу изображений,

поступающих с веб-камеры, на вид с высоты птичьего полета. За реализацию такого преобразования в библиотеке OpenCV отвечают две функции. Первая называется getPerspectiveTransform(src, dst), где:

– src – массив, хранящий координаты 4 вершин области на исходном изображении, перспективу которой требуется изменить;

– dst – массив, хранящий координаты 4 вершин прямоугольной области на выходном изображении, в которую будет помещен преобразованный фрагмент.

Данная функция возвращает матрицу преобразования, необходимую для дальнейшей работы. За непосредственное изменение перспективы изображения отвечает функция warpPerspective(src, dst, M, dsize), где:

– src – входное изображение, область которого требуется преобразовать;

– dst – выходное изображение, на котором будет размещен измененный фрагмент;

– M – матрица преобразования, представляющая собой результат работы функции getPerspectiveTransform();

– dsize – размер выходного изображения.

Результат преобразования перспективы изображения представлен на рисунке 26.

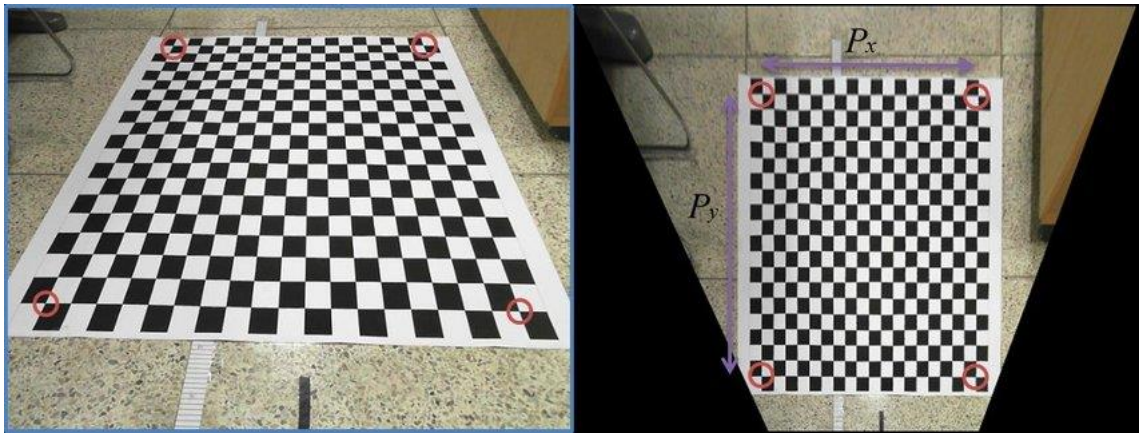

Рисунок 26 – результат преобразования перспективы изображения

Далее полученный кадр переводится в пространство HSV и проходит цветовую сегментацию для обнаружения направляющих маркеров. После этого к бинарному изображению применяется преобразование Хафа. Найденные линии используются для определения положения мобильного робота в пространстве. Так, например, с их помощью вычисляется ориентация указательных символов внутри маркеров, в частности треугольников, а также расстояние от AGV до ближайшего маркера. Все эти данные затем используются для формирования управляющего воздействия и корректировки траектории движения мобильного робота.

Основной принцип работы алгоритма представлен на рисунке 27.

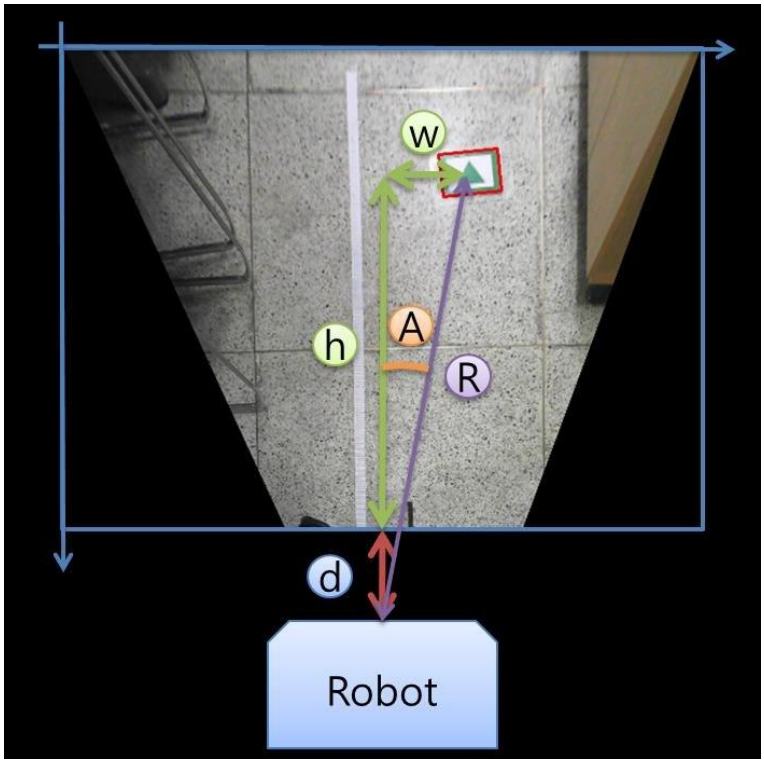

Рисунок 27 – Принцип работы алгоритма корейских ученых

Таким образом, в данном разделе был произведен обзор существующих методов навигации мобильных роботов с помощью компьютерного зрения, а также осуществлен разбор использовавшихся в них функций библиотеки OpenCV.

8 Программная реализация

Программа для распознавания ярко-синей направляющей линии с помощью компьютерного зрения основана на принципе поиска объекта по цвету через цветовое пространство HSV с дальнейшим вычислением положения мобильного робота в пространстве и формированием управляющего воздействия для коррекции траектории его движения. Листинг программы представлен в Приложении 1.

8.1 Алгоритм работы программы

Алгоритм работы программы состоит из 22 последовательных этапов:

– подключение всех необходимых для работы программы заголовочных файлов и библиотек;

– объявление переменных;

– задание цветового диапазона – определение верхней и нижней границ;

– инициализация камеры;

– открытие последовательного порта;

– проверка успешности открытия последовательного порта. В случае возникновения какой-либо ошибки в консоль одноплатного компьютера Raspberry Pi выведется сообщение «Failed open port» и программа завершит свою работу;

– захват изображения, поступающего с веб-камеры;

– проверка успешности захвата кадра. В случае возникновения какой-либо ошибки в консоль одноплатного компьютера Raspberry Pi выведется сообщение «Camera error» и программа завершит свою работу;

– преобразование полученного кадра из цветового пространства BGR в цветовую модель HSV;

– цветовая сегментация с помощью подобранных ранее границ и извлечение бинарного изображения с выделением области заданного цвета, идентификация обнаруженного синего объекта в качестве направляющей линии;

– нахождение моментов изображения;

– сравнение площади распознанной линии с установленным значением. При превышении порога программа продолжит свою работу и перейдет к дальнейшим математическим преобразованиям. В противном случае в консоль одноплатного компьютера Raspberry Pi выведется сообщение «Line was not detected», произойдет пропуск всех вычислений и вывод найденного кадра с данными из предыдущего цикла;

– определение координат середины направляющей линии;

– нахождение координат центра кадра;

– вычисление разности x-составляющих координат середины линии и центра кадра и запись полученного значения в переменную delta с целью ее дальнейшего использования в качестве управляющего воздействия;

– визуализация полученной информации о положении мобильного робота;

– запись значения переменной delta в виде отдельных цифр в специальный массив-буфер для последующей отправки;

– отправка элементов, хранящихся в буфере, через последовательный порт на микроконтроллер Arduino;

– принудительное обнуление массива-буфера;

– вывод изображения с веб-камеры на дисплей, подключенный к одноплатному компьютеру;

– проверка нажатия пользователем какой-либо клавиши на клавиатуре в течение 10 мс, которое означало бы завершение работы программы. При отсутствии действий со стороны человека происходит автоматический переход к началу нового цикла – захват нового кадра;

– закрытие последовательного порта и завершение программы при условии, что кнопка была нажата.

Блок-схема описанного выше алгоритма работы программы представлена на рисунке 28.

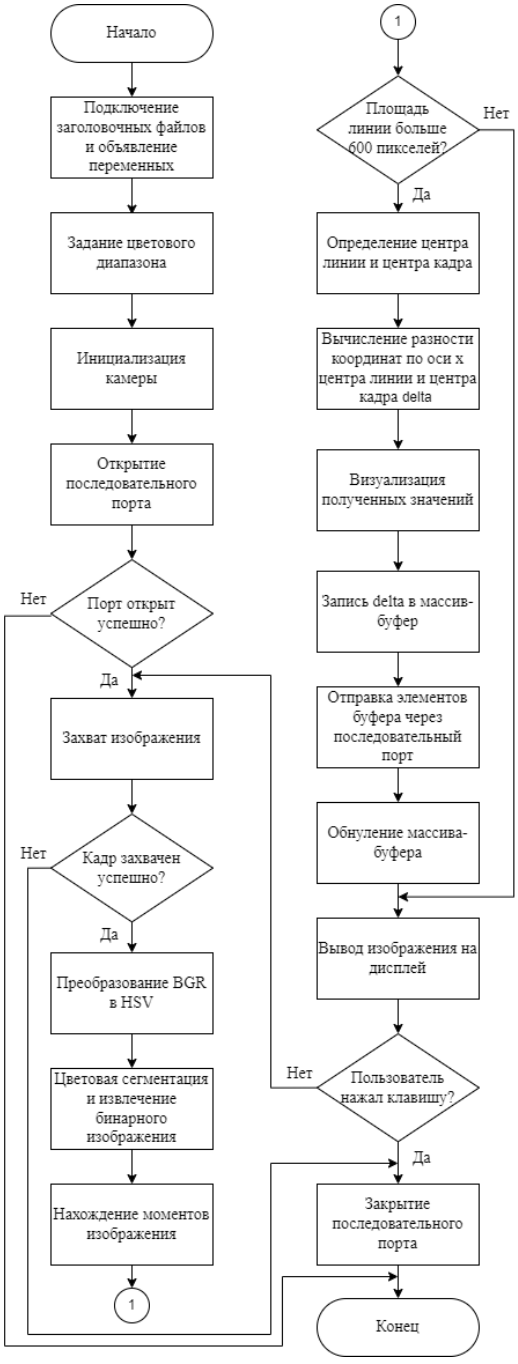

Рисунок 28 – Блок-схема алгоритма работы программы

Подробный разбор функций, отвечающих за реализацию приведенного алгоритма, представлен в следующем разделе.

8.2 Подробный разбор кода программы

Проведем детальный разбор кода программы.

Первым делом производится подключение всех необходимых для работы заголовочных файлов:

– opencv2/core.hpp содержит классы и функции, которые используются для работы с базовыми структурами данных библиотеки OpenCV: матрицами, векторами, точками и т.д. Эти объекты являются основными строительными блоками для создания приложений компьютерного зрения;

– opencv2/imgproc.hpp содержит функции, которые используются для обработки изображений. Данный модуль дает доступ к инструментам, позволяющим, например, находить границы и контуры объектов, настраивать яркость и контрастность, изменять цветовые пространства и многое другое;

– opencv2/highgui.hpp содержит функции, которые используются для работы с графическим интерфейсом пользователя. Данный модуль отвечает за создание и отображение окон, чтение и запись изображений и т.д.;

– opencv2/videoio.hpp содержит функции, которые используются для захвата видео с камеры, а также чтения и записи видеофайлов;

– iostream отвечает за ввод пользователем информации с клавиатуры и вывод данных в консоль;

– vector дает возможность работать с одноименной динамической структурой данных, которая обеспечивает быстрое добавление новых элементов, автоматически меняет свой размер при необходимости и гарантирует отсутствие утечек памяти;

– wiringPi позволяет обращаться к GPIO интерфейсу Raspberry Pi;

– wiringSerial открывает доступ к инструментам для работы с последовательным портом Raspberry Pi;

– unistd.h содержит функцию delay(), которая позволяет приостанавливать выполнение программы на определенное время, указанное внутри скобок и измеряемое в миллисекундах.

Следующим шагом является объявление внутри основной функции main() всех переменных и структур, которые будут использоваться в программе в дальнейшем.

Объекты типа Mat представляют собой n-мерные массивы или матрицы, используемые для хранения изображений. Так, frame будет содержать исходный кадр с веб-камеры в цветовом пространстве BGR, hsv – тот же кадр, но уже в формате HSV, а mask – бинарное изображение после сегментации кадра hsv по цвету и выделения направляющей линии.

Объект m типа Moments будет хранить моменты изображения двоичного кадра mask.

Ниже находятся целочисленные переменные типа int, которые будут участвовать в математических преобразованиях. Width будет содержать в себе ширину всех кадров в пикселях, x и y – соответствующие координаты середины направляющей линии, center – положение центра кадра на оси абсцисс, а delta – уже упомянутую ранее разность x-составляющих координат середины линии и центра кадра, являющуюся управляющем воздействием для мобильного робота.

Далее идут нижняя (lowerBlue) и верхняя (upperBlue) границы диапазона оттенков синего для сегментации изображения в цветовом пространстве HSV, представляющие собой два массива типа vector, содержащие по 3 элемента типа int:

vector<int>lowerBlue = { 75, 60, 50 };

vector<int>upperBlue = { 130, 255, 255 };

Стоит отметить, что в библиотеке OpenCV значения цвета кодируются числами от 0 до 180, насыщенности – от 0 до 255, яркости – также от 0 до 255.

Массив-буфер buffer типа char, способный хранить в себе 5 элементов, будет использоваться в процессах визуализации управляющего воздействия delta и ее отправки через последовательный порт на микроконтроллер Arduino.

Далее происходит инициализация камеры с помощью экземпляра cap класса VideoCapture библиотеки OpenCV, предназначенного для захвата видео и чтения видеофайлов. К данному объекту применяется метод open(), принимающий индекс устройства, с которого необходимо произвести захват видеопотока. Так как разрабатываемая система предусматривает наличие лишь одной камеры, то аргументом для метода open() будет являться число 0. Запись, отвечающая за это, в коде выглядит следующим образом:

```
VideoCapture cap;
cap.open(0);
```

Следующим шагом является инициализация последовательного порта, к которому подключен микроконтроллер Arduino, с помощью команды serialOpen(), принимающей в качестве аргументов номер порта и скорость передачи данных. Контроль за состоянием порта будет осуществляться с помощью переменной serialPort типа int:

```
int serialPort = serialOpen("/dev/ttyUSB0", 9600);
```

Следом идет проверка успешности открытия последовательного порта с помощью конструкции if. Если переменная serialPort содержит значение -1, то это означает, что связь между Raspberry Pi и Arduino не была установлена, вследствие чего в консоль одноплатного компьютера выведется сообщение об ошибке «Failed open port», а программа завершит свою работу.

Далее искусственно создается небольшая задержка длительностью в 1 секунду с помощью функции delay() для обеспечения синхронизации устройств и стабильности соединения.

После этого программа попадает в бесконечный цикл, в котором осуществляется поиск направляющей линии и формирование управляющего воздействия.

Данный процесс начинается с захвата изображения, поступающего в одноплатный компьютер с веб-камеры, при помощи упомянуто ранее объекта cap и записи его в матрицу frame:

cap >> frame;

Следом идет проверка успешности захвата кадра с помощью конструкции if. Если метод empty, примененный к объекту frame, возвращает логическое значение true, то это означает, что во время захвата видеопотока произошла ошибка и матрица для хранения изображения осталась пустой, вследствие чего в консоль одноплатного компьютера Raspberry Pi выведется сообщение об ошибке «Camera error», а программа завершит свою работу.

Пример кадра, полученного с веб-камеры, мобильным роботом представлен на рисунке 29.

Рисунок 29 – Кадр с веб-камеры

Далее происходит вычисление ширины захваченного кадра в пикселях с помощью метода size().width, применяемого к матрице frame, и запись полученного значения в переменную width.

Следующим шагом является преобразование полученного изображения в цветовое пространство HSV с помощью функции библиотеки OpenCV cvtColor(src, dst. code), где:

– src – входной кадр, который необходимо преобразовать;

– dst – выходной преобразованный кадр;

– code – код требуемого преобразования.

Запись, отвечающая за это, в коде выглядит следующим образом:

cvtColor(frame, hsv, COLOR_BGR2HSV);

В данном случае на месте параметра code находится запись COLOR_BGR2HSV, сигнализирующая программе о том, что необходимо провести преобразование изображения, хранящегося в матрице frame, из цветового пространства BGR в HSV и записать его в матрицу под названием hsv. Стоит отметить, что в библиотеке OpenCV в качестве стандартной цветовой модели используется пространство BGR вместо более распространенного на сегодняшний день RGB. Причиной этого стала популярность данного цветового формата среди производителей камер и программного обеспечения в прошлом, когда библиотека еще только разрабатывалась.

Пример кадра в цветовой модели HSV представлен на рисунке 30.

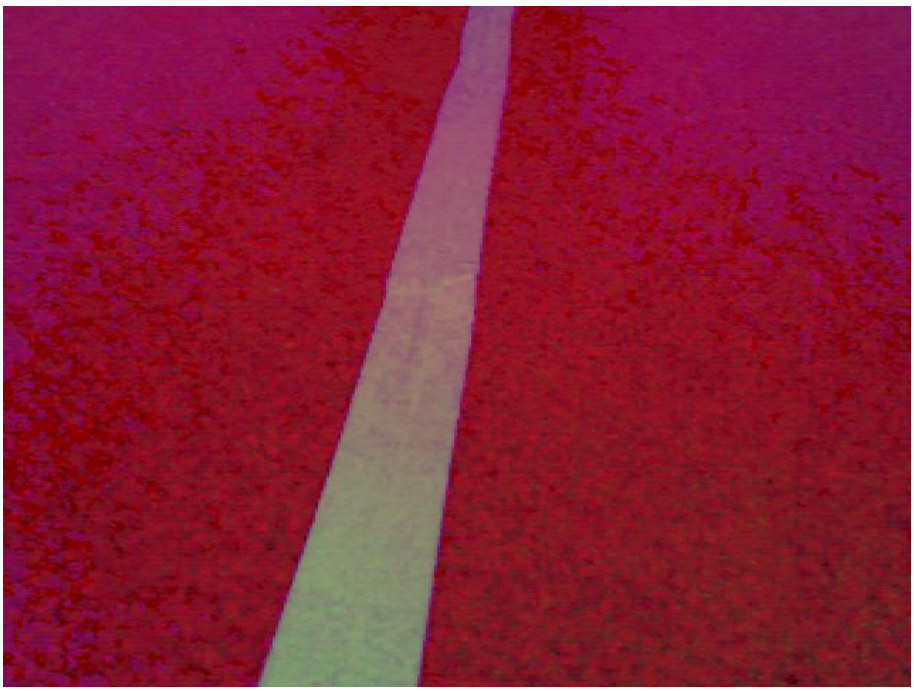

Рисунок 30 – Кадр в цветовом пространстве HSV

Далее переходим к цветовой сегментации кадра с помощью функции inRange(). Данный алгоритм проверяет попадание пикселей входного изображения hsv в диапазон синего цвета, заданный при помощи ранее определенных границ lowerBlue и upperBlue. Если пиксель находится в установленных пределах, то он принимает значение 1, т.е. окрашивается в белый цвет. В противном случае пикселю присваивается значение 0 и он становится черным. Полученное бинарное изображение записывается в матрицу mask. Запись, отвечающая за это, в коде выглядит следующим образом:

inRange(hsv, lowerBlue, upperBlue, mask);

Пример кадра после цветовой сегментации представлен на рисунке 31.

Рисунок 31 – Кадр после цветовой сегментации

На этом этапе мы предполагаем, что единственным ярко-синим объектом в поле зрения веб-камеры является направляющая линия. Из этого следует, что все белые области получившегося двоичного кадра будут интерпретироваться как направляющая линия.

Ниже происходит вычисление моментов бинарного изображения mask и запись их в специальный объект m типа Moments:

m = moments(mask, 1);

Следом идет сравнение площади распознанной линии с установленным значением с помощью момента первого порядка M_{00}. При превышении порога программа продолжит свою работу и перейдет к дальнейшим математическим операциям. В противном случае в консоль Raspberry Pi выведется сообщение «Line was not detected» и начнется повторный поиск – одноплатный компьютер пропустит текущую итерацию бесконечного цикла и перейдет к следующей. Данная проверка позволяет мобильному роботу в случае потери направляющей линии на повороте игнорировать незначительные объекты синего цвета, которые могут попасть в поле зрения

его камеры, и продолжать двигаться согласно управляющим значениям, вычисленным ранее, когда линия еще присутствовала в кадре. Этот прием значительно повышает шанс для транспортного средства самостоятельно вернуться на «трассу».

При выполнении описанного выше условия программа переходит к нахождению координат x и y середины направляющей линии с помощью моментов второго порядка M_{10} и M_{01} по формулам (5) и (6). Далее вычисляется положение центра кадра на оси абсцисс, равное половине ширины изображения width, которое записывается в переменную center. После этого происходит расчет управляющего воздействия delta, представляющего собой разность x-составляющих координат середины линии и центра кадра.

Следующим шагом является визуализация полученных данных на дисплее, которая поможет пользователю легко отследить то, как система распознает направляющую линию, и оценить, насколько корректными являются вычисленные микрокомпьютером значения управляющего воздействия delta при различных положениях мобильного робота, для последующей корректировки кода.

Первым делом изобразим центры линии и кадра в виде кружков красного и зеленого цвета соответственно. Для этого воспользуемся функцией для рисования окружностей библиотеки OpenCV circle(img, center, radius, color, thickness), где:

– img – изображение, на котором будет нарисован окружность;

– center – координаты центра окружности;

– radius – радиус окружности;

– color – цвет окружности;

– thickness – толщина контура окружности. При отрицательном значении данного параметра будет нарисован сплошной круг.

Запись, отвечающая за это, в коде выглядит следующим образом:

circle(frame, Point(x, y), 5, Scalar(0, 0, 255), -1);

circle(frame, Point(center, y), 5, Scalar(0, 255, 0), -1);

Далее изобразим желтый отрезок, соединяющий данные кружки, для визуальной оценки расстояния между серединой линии и центром кадра. Для этого воспользуемся функцией библиотеки OpenCV line(img, pt1, pt2, color, thickness), где:

– img – изображение, на котором будет нарисована линия;

– pt1 – координаты начала отрезка;

– pt1 – координаты конца отрезка;

– color – цвет линии;

– thickness – толщина линии.

Запись, отвечающая за это, в коде выглядит следующим образом:

line(frame, Point(center, y), Point(x, y), Scalar(0, 255, 255), 2);

Последним этапом визуализации является нанесение найденного значения управляющего воздействия на изображение в виде текста. Для этого воспользуемся созданным ранее массивом-буфером buffer и занесем в него число, хранящееся в переменной delta, в виде отдельных символов типа char с помощью команды sprintf():

sprintf(buffer, "%d", delta);

Затем обратимся к функции библиотеки OpenCV putText(img, text, org, fontFace, fontScale, color, thickness), где:

– img – изображение, на которое будет выведена надпись;

– text – текст, который необходимо вывести;

– org – координаты, в которых будет выведен первый символ надписи;

– fontFace – тип шрифта;

– fontScale – коэффициент масштабирования, который умножается на базовый размер конкретного шрифта;

– color – цвет текста;

– thickness – толщина символов.

Запись, отвечающая за это, в коде выглядит следующим образом:

putText(frame, buffer, Point(width/2, y - 20), FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255), 3);

Результат вышеперечисленных действий представлен на рисунке 32.

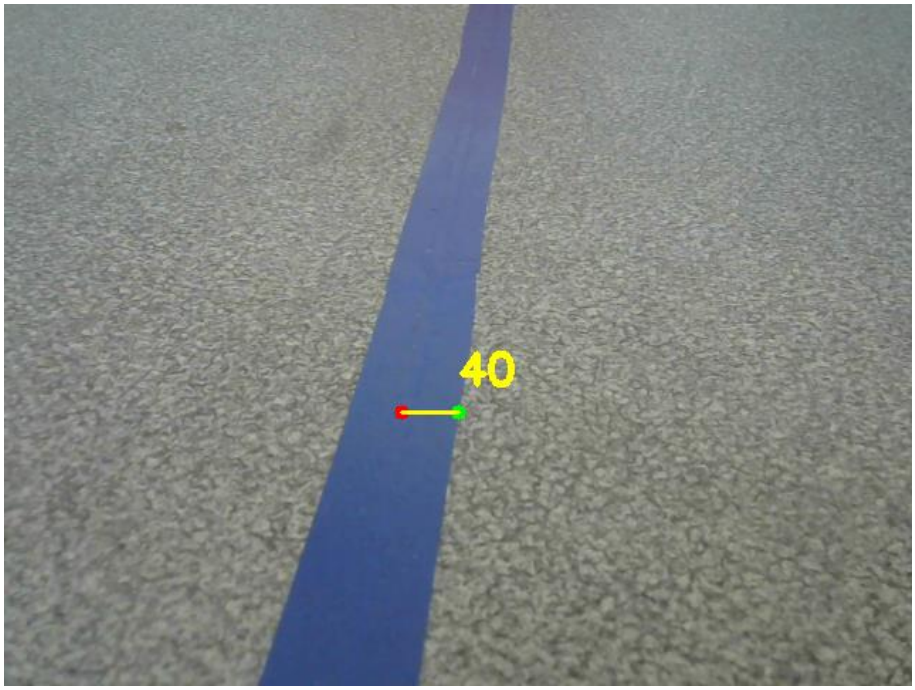

Рисунок 32 – Кадр с визуализацией данных

Следующим шагом является отправка управляющего воздействия delta на микроконтроллер Arduino. Для этого используется команда serialPrintf(), которая передает через последовательный порт набор символов из массива-буфера buffer, хранящего в себе интересующее нас значение:

serialPrintf(serialPort, "%s\n", buffer);

После этого происходит принудительное обнуление всех элементов буфера во избежание каких-либо ошибок при дальнейших вычислениях с помощью функции memset(), принимающей в качестве аргументов указатель на массив, значение, к которому будут приравнены его элементы, а также количество этих элементов:

memset(buffer, 0, sizeof buffer);

Далее идут конструкции, отвечающие за вывод изображения на дисплей.

Окно, через которое происходит трансляция кадров с веб-камеры, создается с помощью функции библиотеки OpenCV namedWindow(winname, flag), где:

– winname – название окна;

– flag – тип отображения окна. Например, WINDOW_NORMAL позволяет пользователю изменять размеры окна, тогда как WINDOW_AUTOSIZE автоматически подгоняет ширину и высоту окна под отображаемое изображение без возможности регулировки.

Запись, отвечающая за это, в коде выглядит следующим образом:

namedWindow("Window", WINDOW_AUTOSIZE);

Для лучшего позиционирования окна на экране дисплея используется функция библиотеки OpenCV moveWindow(winname, x, y), где:

– winname – название окна, которое будет передвинуто;

– x – координата точки по оси абсцисс, в которую будет перемещен левый верхний угол передвигаемого окна;

– y – координата точки по оси ординат, в которую будет перемещен левый верхний угол передвигаемого окна.

Запись, отвечающая за это, в коде выглядит следующим образом:

moveWindow("Window", 70, 70);

За непосредственную демонстрацию изображения в созданном окне отвечает функция библиотеки OpenCV imshow(winname, mat), где:

– winname – название окна, в котором будет показываться кадр;

– mat – демонстрируемое изображение.

Запись, отвечающая за это, в коде выглядит следующим образом:

imshow("Window", frame);

Завершает данный блок кода условие, при котором система в течение 10 мс ждет нажатия пользователем какой-либо клавиши на клавиатуре для выхода из бесконечного цикла и завершения своей работы. Если действий со

стороны человека по истечении этого времени не будет зарегистрировано, программа перейдет к следующей итерации. Реализация данной проверки основана на функции библиотеки OpenCV waitKey(delay), принимающей в качестве аргумента количество миллисекунд, в течение которых система будет ожидать нажатия какой-либо клавиши на клавиатуре. Помимо этого данная функция также возвращает код используемой кнопки. Таким образом, условие if(waitKey(10) >= 0) срабатывает при нажатии любой клавиши, а if(waitKey(10) == 27) – только при взаимодействии с esc.

При выходе из бесконечного цикла происходит разрушение окна, посредством которого осуществлялась трансляция кадров с веб-камеры, при помощи функции библиотеки OpenCV destroyWindow(), принимающей в качестве аргумента имя данного окна, а также закрытие последовательного порта с помощью команды serialClose(), аргументом к которой является переменная serialPort, отвечающая за связь между Raspberry Pi и Arduino.

После применения оператора return 0 работа программы завершается.

Таким образом, в данном разделе была разработана и подробно описана программа для распознавания ярко-синей направляющей линии с помощью компьютерного зрения, основанная на принципе поиска объекта по цвету через цветовое пространство HSV с дальнейшим вычислением положения мобильного робота в пространстве и формированием управляющего воздействия для коррекции траектории его движения.

9 Возможные варианты модернизации программы

Представленная выше программа прекрасно справляется с выполнением поставленной задачи – обнаружением направляющей линии. Однако, несмотря на это, она имеет огромный потенциал для дальнейшего совершенствования.

9.1 Модификация №1

Одним из путей улучшения разработанной программы является добавление дополнительных управляющих воздействий, отвечающих за детектирование отклонений мобильного робота от середины направляющей линии в разных частях кадра. Листинг модифицированной программы, реализующей данный подход, представлен в Приложении 2, а результат ее работы – на рисунке 33.

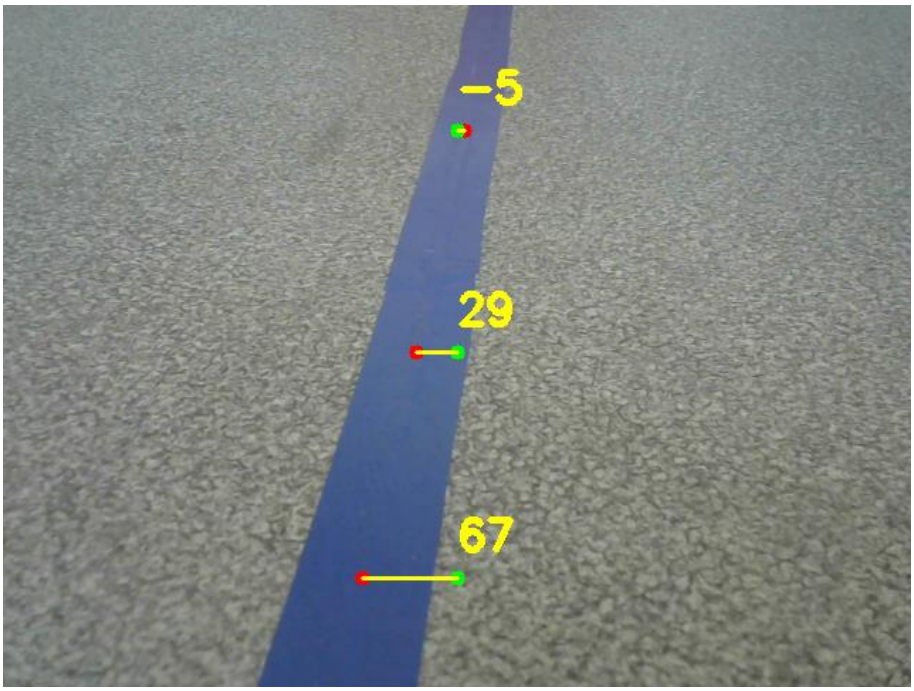

Рисунок 33 – Результат работы модификации программы №1

Алгоритм работы данной модификации идентичен алгоритму работы основной программы за исключением того, что теперь пункты 9-16 выполняются по одному разу для каждой из трех разных частей кадра: верхней, средней и нижней.

Подобное разделение поля зрения камеры реализовано с помощью метода для обрезки изображений Range(). Его запись происходит следующим образом:

```
img(Range(start_row, end_row), Range(start_col, end_col));
```

Данная функция имеет несколько параметров:

– img – изображение, которое необходимо обрезать;

– start_row – y-координата начала сохраняемого участка;

– end_row – y-координата конца сохраняемого участка;

– start_col – x-координата начала сохраняемого участка;

– end_col – x-координата конца сохраняемого участка.

Главным преимуществом данного подхода является получение большего количества данных о положении мобильного робота относительно направляющей линии, что позволяет более точно настроить ПИД-регулятор транспортного средства и, как следствие, добиться увеличения плавности его движения и улучшения прохождения им поворотов.

Недостатком данной модификации является значительное увеличение количества математических преобразований, совершаемых в процессе выполнения программы, что увеличивает потребление вычислительных ресурсов микрокомпьютера и негативно сказывается на его быстродействии.

9.2 Модификация №2

Другим возможным способом улучшения разработанной программы может быть использование контуров для нахождения середины направляющей линии вместо моментов изображения.

Основным недостатком поиска центроида объекта с помощью моментов изображения является негативное влияние посторонних объектов в кадре того же цвета, что и искомая линия. Причина этого заключается в том, что элементы M_{00}, M_{10} и M_{01} высчитываются с учетом абсолютно всех белых пикселей бинарного изображения вне зависимости от их положения и принадлежности тому или иному предмету. Из этого следует, что появление в поле зрения камеры крупного объекта ярко-синего цвета способно значительно повлиять на процесс вычисления управляющего воздействия и изменить траекторию движения мобильного робота.

Одним из возможных решений данной проблемы является нахождение контуров всех синих объектов в кадре, определение наибольшего из них и интерпретация его в качестве направляющей линии. Листинг модифицированной программы, реализующей упомянутый выше подход, представлен в Приложении 2. Блок-схема алгоритма работы данной модификации представлена на рисунке 34.

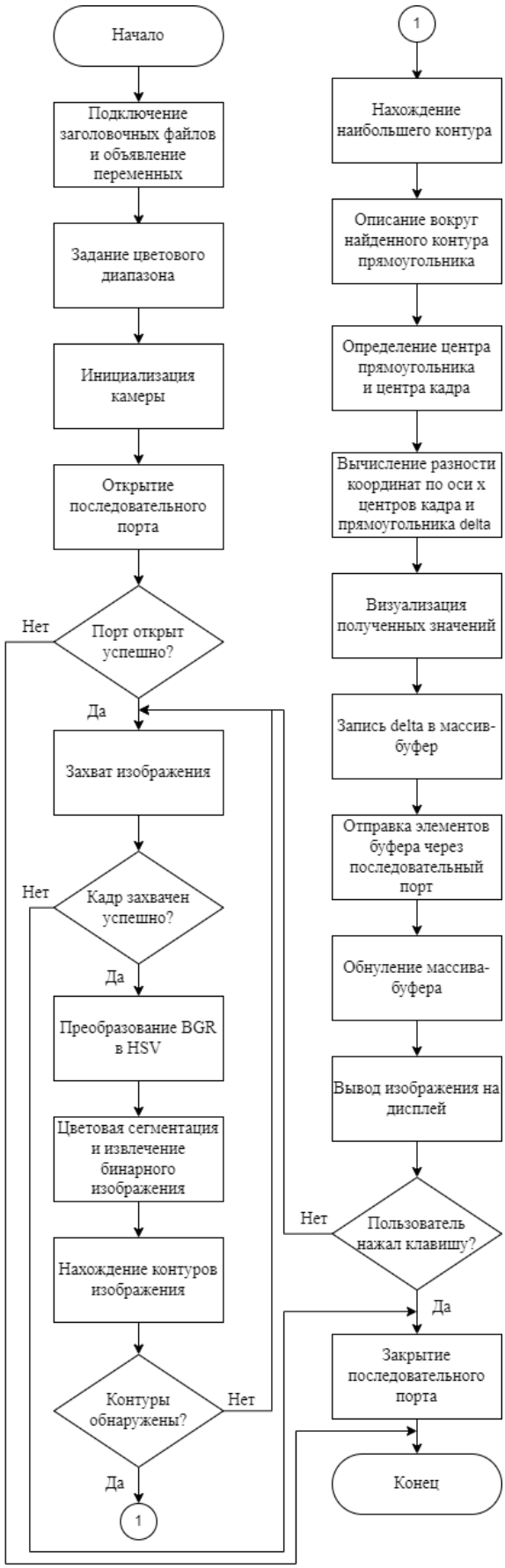

Рисунок 34 – Блок-схема алгоритма работы модификации программы №2

Проведем разбор отдельных ключевых функций модифицированной программы.

Нахождение границ всех синих объектов в кадре осуществляется с помощью функции библиотеки OpenCV findContours(). Запись, отвечающая за это, в коде выглядит следующим образом:

findContours(mask, contours, hierarchy, RETR_TREE, CHAIN_APPROX_NONE);

В данном случае функция принимает бинарное изображение mask, прошедшее цветовую сегментацию, записывает все найденные контуры в массив под названием contours типа vector<vector<Point>> в виде множества отрезков, а также организует их многоуровневую иерархию, хранимую в массиве hierarchy типа vector<Vec4i>.

Следом идет проверка успешности обнаружения границ объектов с помощью конструкции if. Если метод empty, примененный к объекту contours, возвращает логическое значение true, то это означает, что во время обработки изображения не было найдено ни одного контура и массив для их хранения остался пустым, вследствие чего в консоль Raspberry Pi выведется сообщение об ошибке «Line was not detected» и начнется повторный поиск – одноплатный компьютер пропустит текущую итерацию бесконечного цикла и перейдет к следующей.

Далее в цикле for происходит перебор всех обнаруженных контуров, вычисляются их площади с помощью функции contourArea(), принимающей в качестве аргумента элементы массива contours, которые затем сравниваются с заранее установленным значением. При превышении порога текущий контур визуализируется с помощью функции drawContours(), а вокруг него описывается вращающийся прямоугольник минимально возможных размеров с помощью объекта rect типа RotatedRect и функции библиотеки OpenCV minAreaRect(). Данное пороговое значение подбирается таким образом, чтобы выявлять границы наибольшего синего объекта в кадре, т.е.

направляющей линии. Все остальные более мелкие контуры просто игнорируются [11].

Ниже происходит вычисление управляющего воздействия delta, которое в данном случае представляет собой разность x-составляющих координат центра описываемого прямоугольника и центра кадра. В качестве дополнительного параметра может также использоваться угол поворота прямоугольника angle между его длиной и осью абсцисс. Запись, отвечающая за это, в коде выглядит следующим образом:

```
delta = width / 2 - rect.center.x;
angle = rect.angle;
```

Следующим шагом является визуализация полученных данных с помощью уже разобранных ранее функций для рисования линий и окружностей, а также вывода текста на экран.

Результат работы данной модификации представлен на рисунке 35.

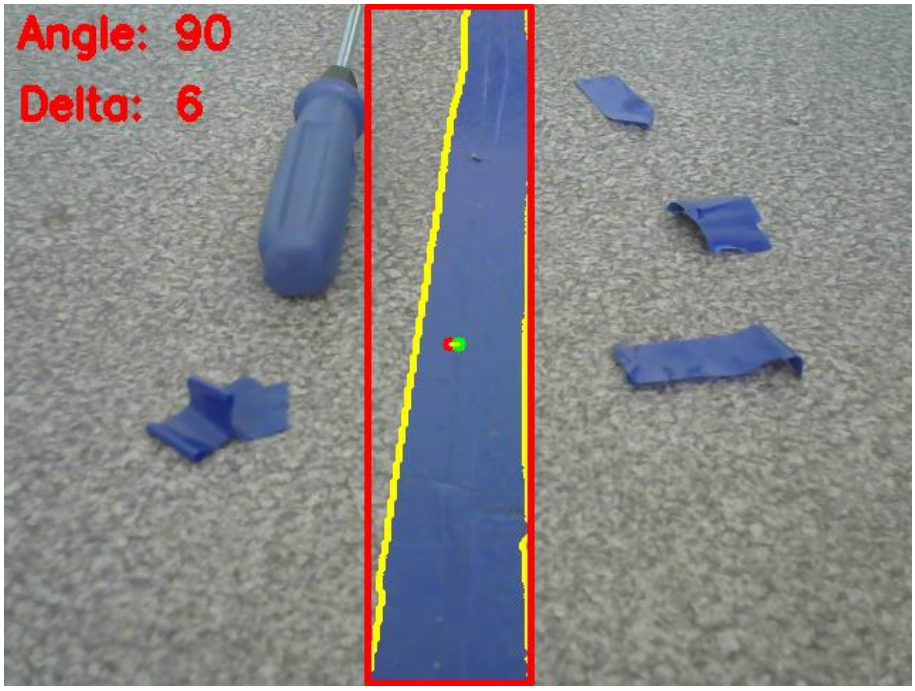

Рисунок 35 – Результат работы модификации программы №2

Главным преимуществом описанного выше подхода с использованием контуров вместо моментов изображения является возможность устранения негативного воздействия со стороны случайных помех и посторонних объектов синего цвета в кадре на движение мобильного робота.

Недостатками данной модификации является увеличение количества математических преобразований, совершаемых в процессе выполнения программы, а также большее использование графических функций для визуализации полученных данных, что увеличивает потребление вычислительных ресурсов микрокомпьютера и негативно сказывается на его быстродействии.

Таким образом, в данном разделе были разработаны возможные модификации основной программы для распознавания мобильным роботом направляющей линии с помощью компьютерного зрения, а также рассмотрены их основные достоинства и недостатки.

10 Экономический расчет

Стоимость каждого отдельно взятого компонента, входящего в состав разрабатываемой системы, а также итоговая стоимость устройства приведена на рисунке 36.

№	Наименование	Кол.	Цена, руб.
1	Raspberry Pi 3 Model B	1	8400
2	Веб-камера Logitech C270	1	1600
3	Вентилятор	1	100
4	LCD дисплей Waveshare 4,3"	1	6000
5	Карта microSD 8 GB	1	300
6	Клавиатура	1	400
7	Компьютерная мышь	1	300
8	LM2596S	1	150
9	KIS-3R33S	1	120
10	Аккумулятор 11,8V 5000mAh	1	6000
11	Шнур USB A – USB A	3	200
12	Шнур USB A – Micro USB	2	200
13	Шнур USB A – USB B	1	200
14	Кабель HDMI	1	300
		Итого	24870

Рисунок 36 – Экономический расчет

Таким образом, в данном разделе был проведен экономический расчет разрабатываемой системы на основе тщательного анализа рынка.

Заключение

В рамках выполнения данной выпускной квалификационной работы была разработана система распознавания направляющей линии мобильным роботом с помощью компьютерного зрения.

В первом разделе была подтверждена актуальность использования систем компьютерного зрения при разработке автоматически управляемых транспортных средств и автономных мобильных роботов в сфере складской логистики.

Во втором разделе был произведен выбор одноплатного компьютера и веб-камеры для разрабатываемой системы.

В третьем разделе была составлена структурная схема разрабатываемой системы и представлен перечень всех используемых в ней элементов.

В четвертом разделе была составлена схема электрическая соединений разрабатываемой системы, а также представлены программы и блок-схемы алгоритмов передачи и приема данных по последовательному порту с одноплатного компьютера Raspberry Pi на микроконтроллер Arduino.

В пятом разделе был произведен выбор языка программирования, библиотеки компьютерного зрения и среды разработки для реализации программной части выпускной квалификационной работы

В шестом разделе были подробно описаны алгоритмы и функции библиотеки компьютерного зрения OpenCV, необходимые для поэтапного преобразования получаемой с веб-камеры видеоинформации к виду, позволяющему мобильному роботу обнаружить направляющую линию.

В седьмом разделе был произведен обзор существующих методов навигации мобильных роботов с помощью компьютерного зрения, а также осуществлен разбор использовавшихся в них функций библиотеки OpenCV.

В восьмом разделе была разработана и подробно описана программа для распознавания ярко-синей направляющей линии с помощью компьютерного зрения, основанная на принципе поиска объекта по цвету

через цветовое пространство HSV с дальнейшим вычислением положения мобильного робота в пространстве и формированием управляющего воздействия для коррекции траектории его движения.

В девятом разделе были разработаны возможные модификации основной программы для распознавания мобильным роботом направляющей линии с помощью компьютерного зрения, а также рассмотрены их основные достоинства и недостатки.

В десятом разделе был проведен экономический расчет разрабатываемой системы на основе тщательного анализа рынка.

Результатом выполнения данной выпускной квалификационной работы стало внедрение разработанной системы для распознавания направляющей линии с помощью компьютерного зрения в систему управления движением реального мобильного робота в качестве основного алгоритма поиска пути.

Список используемой литературы

1. 33. OpenCV шаг за шагом. Сравнение контуров через суммарные характеристики – моменты [Электронный ресурс]. URL: https://robocraft.ru/computervision/867 (дата обращения: 23.04.2023).

2. Автоматизация складов с помощью роботов [Электронный ресурс]. URL: https://top3dshop.ru/blog/warehouse-and-logistics-robots-review.html (дата обращения: 21.03.2023).

3. Булатников, Е. В. Сравнение библиотек компьютерного зрения для применения в приложении, использующем технологию распознавания плоских изображений / Е. В. Булатников, А. А. Гоева. // Вестник МГУП имени Ивана Федорова – 2015. – №6 – С. 85-91. – ISSN 2409-6652.

4. Волков, К. А. Алгоритмы и программные библиотеки обработки мультимедийной информации : учеб.-метод. пособие / К. А. Волков, В. Ю. Цветков, В. К. Конопелько. – Минск : БГУИР, 2017. – 99 с. : ил. – ISBN 978-985-543-246-4.

5. Кэлер, А. Изучаем OpenCV 3. Разработка программ компьютерного зрения на C++ с применением библиотеки OpenCV / А. Кэлер, Г. Брэдски ; пер. с англ. – Москва : ДМК Пресс, 2017. – 826 с. : ил. – ISBN 978-5-97060-471-7.

6. Лекция 4: Начало работы с библиотекой OpenCV [Электронный ресурс]. URL: https://intuit.ru/studies/courses/10621/1105/lecture/17985 (дата обращения: 11.04.2023).

7. Обзор роботов AGV и AMR [Электронный ресурс]. URL: https://top3dshop.ru/blog/agv-amr-robots-review.html (дата обращения: 14.03.2023).

8. Обнаружение контуров с использованием OpenCV [Электронный ресурс]. URL: https://waksoft.susu.ru/2021/11/30/obnaruzhenie-kongurov-s-ispolzovaniem-opencv/ (дата обращения: 26.04.2023).

9. Потапов, А. С. Системы компьютерного зрения : учебное пособие / А. С. Потапов. – Санкт-Петербург : Университет ИТМО, 2016. – 161 с.

10. Работа через GPIO в Raspberry Pi 3 [Электронный ресурс]. URL: https://vashumnyidom.ru/upravlenie/ustrojstva/gpio-raspberry-pi-3.html (дата обращения: 26.03.2023).

11. Следование линии на основе OpenCV [Электронный ресурс]. URL: https://habr.com/ru/articles/426675/ (дата обращения: 08.03.2023).

12. Форсайт, Д. Компьютерное зрение. Современный подход / Д. Форсайт, Ж. Понс ; пер. с англ. – Москва : Издательский дом “Вильямс”, 2004. – 928 с. : ил. – ISBN 5-8459-0542-7.

13. Шапиро, Л. Компьютерное зрение / Л. Шапиро, Д. Стокман ; пер. с англ. – Москва : БИНОМ. Лаборатория знаний, 2006. – 752 с. – ISBN 5-94774-384-1.

14. Bradski, G. Learning OpenCV / G. Bradski, A. Kaehler – Sebastopol : O'Reilly Media, 2008. – ISBN: 9780596516130.

15. Chang-Ho Hyun, A Vision-Based Automated Guided Vehicle System with Marker Recognition for Indoor Use / Hyun Chang-Ho, Lee Jeisung, Park Mignon // Sensors – 2013. – №13 – ISSN 1424-8220.

16. CodeBlocks – среда программирования на языке C/C++ [Электронный ресурс]. URL: https://progtips.ru/instrumenty-programmista/codeblocks.html (дата обращения: 16.04.2023).

17. Color Detection & Object Tracking [Электронный ресурс]. URL: https://www.opencv-srf.com/2010/09/object-detection-using-color-seperation.html (дата обращения: 20.04.2023).

18. Davies, E. R. Computer and Machine Vision: Theory, Algorithms, Practicalities / E. R. Davies. – Oxford : Academic Press is an imprint of Elsevier, 2012. – ISBN: 978-0-12-386908-1.

19. DeepPiCar – Часть 4. Автономная навигация по полосам через OpenCV [Электронный ресурс]. URL:

https://machinelearningmastery.ru/deeppicar-part-4-lane-following-via-opencv-737dd9e47c96/ (дата обращения: 01.03.2023).

20. Serial Communication [Электронный ресурс]. URL: https://learn.sparkfun.com/tutorials/serial-communication/all (дата обращения: 07.04.2023).

21. Shape Matching using Hu Moments (C++/Python) [Электронный ресурс]. URL: https://learnopencv.com/shape-matching-using-hu-moments-c-python/ (дата обращения: 22.04.2023).

22. Shrishailappa Patil, Autonomous cars in rural area roads / Patil Shrishailappa, Patil Ashwini, Pophale Sayali Jitendra, Prajapati Ritesh Jogendra, Puranik Prateek Sunil, Potarwar Rushabh // World Journal of Advanced Engineering Technology and Sciences – 2022. – №07(02) – С. 180-188. – ISSN 2582-8266.

Приложение А

Программный код основного алгоритма работы

```
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/videoio.hpp>
#include <iostream>
#include <vector>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <unistd.h>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
	Mat frame, hsv, mask;
	Moments m;
	int width, x, y, center, delta;
	vector<int>lowerBlue = { 75, 60, 50 };
	vector<int>upperBlue = { 130, 255, 255 };
	char buffer[5] = { 0 };

	VideoCapture cap;
	cap.open(0);

	int serialPort = serialOpen("/dev/ttyUSB0", 9600);

	if (serialPort == -1)
	{
		cout << "Failed open port" << endl;
		return 0;
```

```
}

delay(1000);

for (;;)
{
	cap >> frame;
	if (frame.empty())
	{
		cout << "Camera error" << endl;
		break;
	}
	width = frame.size().width;
	cvtColor(frame, hsv, COLOR_BGR2HSV);
	inRange(hsv, lowerBlue, upperBlue, mask);
	m = moments(mask, 1);

	if (m.m00 > 600)
	{
		x = m.m10 / m.m00;
		y = m.m01 / m.m00;
		center = width / 2;
		delta = center - x;

		circle(frame, Point(x, y), 5, Scalar(0, 0, 255), -1);
		circle(frame, Point(center, y), 5, Scalar(0, 255, 0), -1);
		line(frame, Point(center, y), Point(x, y), Scalar(0, 255, 255), 2);
		sprintf_s(buffer, "%d", delta);
		putText(frame, buffer, Point(width / 2, y - 20),
		FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255), 3);

		serialPrintf(serialPort, "%s\n", buffer);
```

```
                memset(buffer, 0, sizeof buffer);
            }
            else
            {
                cout << "Line was not detected" << endl;
            }

            namedWindow("Window", WINDOW_AUTOSIZE);
            moveWindow("Window", 70, 70);
            imshow("Window", frame);
            if (waitKey(10) >= 0) break;
        }

        destroyWindow("Window");
        serialClose(serialPort);
        return 0;
}
```

Приложение Б

Программный код модификации №1 основного алгоритма

```
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/videoio.hpp>
#include <iostream>
#include <vector>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <unistd.h>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
	Mat frame, hsv, mask, top, mid, bot, maskTop, maskMid, maskBot;
	Moments m, mTop, mMid, mBot;
	int height, width;
	int topX = 0, topY = 0, midX = 0, midY = 0, botX = 0, botY = 0, topDelta = 0,
	midDelta = 0, botDelta = 0, delta;
	vector<int>lowerBlue = { 75, 60, 50 };
	vector<int>upperBlue = { 130, 255, 255 };
	char showDeltaTop[5] = { 0 };
	char showDeltaMid[5] = { 0 };
	char showDeltaBot[5] = { 0 };
	char buffer[5] = { 0 };

	VideoCapture cap;
	cap.open(0);

	int serialPort = serialOpen("/dev/ttyUSB0", 9600);
```

```
if (serialPort == -1)
{
	cout << "Failed open port" << endl;
	return 0;
}

delay(1000);

for (;;)
{
	cap >> frame;
	if (frame.empty())
	{
		cout << "Camera error" << endl;
		break;
	}
	height = frame.size().height;
	width = frame.size().width;
	cvtColor(frame, hsv, COLOR_BGR2HSV);

	top = hsv(Range(0, (height / 3)), Range(0, width));
	mid = hsv(Range((height / 3), 2*(height / 3)), Range(0, width));
	bot = hsv(Range(2 * (height / 3), height), Range(0, width));

	inRange(top, lowerBlue, upperBlue, maskTop);
	mTop = moments(maskTop, 1);
	inRange(mid, lowerBlue, upperBlue, maskMid);
	mMid = moments(maskMid, 1);
	inRange(bot, lowerBlue, upperBlue, maskBot);
	mBot = moments(maskBot, 1);

	if (mTop.m00 > 600)
```

```
{
        topX = mTop.m10 / mTop.m00;
        topY = mTop.m01 / mTop.m00;

        topDelta = width / 2 - topX;
        sprintf_s(showDeltaTop, "%d", topDelta);

        circle(frame, Point(topX, topY), 5, Scalar(0, 0, 255), -1);
        circle(frame, Point(width / 2, topY), 5, Scalar(0, 255, 0), -1);
        putText(frame, showDeltaTop, Point(width / 2, topY - 20),
        FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255), 3);
        line(frame, Point(width / 2, topY), Point(topX, topY),
        Scalar(0, 255, 255), 2);
}
else
{
        cout << "The top of the line was not detected" << endl;
}
if (mMid.m00 > 600)
{
        midX = mMid.m10 / mMid.m00;
        midY = mMid.m01 / mMid.m00;

        midDelta = width / 2 - midX;
        sprintf_s(showDeltaMid, "%d", midDelta);

        circle(frame, Point(midX, midY + (height / 3)), 5, Scalar(0, 0, 255), -1);
        circle(frame, Point(width / 2, midY + (height / 3)), 5, Scalar(0, 255, 0), -1);
        putText(frame, showDeltaMid, Point(width / 2, midY + (height / 3) - 20),
        FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255), 3);
        line(frame, Point(width / 2, midY + (height / 3)), Point(midX, midY +
        (height / 3)), Scalar(0, 255, 255), 2);
```

```
}
else
{
	cout << "The middle part of the line was not detected" << endl;
}
if (mBot.m00 > 600)
{
	botX = mBot.m10 / mBot.m00;
	botY = mBot.m01 / mBot.m00;

	botDelta = width / 2 - botX;
	sprintf_s(showDeltaBot, "%d", botDelta);

	circle(frame, Point(botX, botY + (2 * height / 3)), 5, Scalar(0, 0, 255), -1);
	circle(frame, Point(width / 2, botY + (2 * height / 3)), 5,
	Scalar(0, 255, 0), -1);
	putText(frame, showDeltaBot, Point(width / 2, botY + (2 * height / 3) –
	20), FONT_HERSHEY_SIMPLEX, 1, Scalar(0, 255, 255), 3);
	line(frame, Point(width / 2, botY + (2 * height / 3)), Point(botX, botY +
	(2 * height / 3)), Scalar(0, 255, 255), 2);
}
else
{
	cout << "The bottom of the line was not detected" << endl;
}

topDelta = 0.25 * topDelta;
midDelta = 0.5 * midDelta;
botDelta = 2 * botDelta;
delta = (topDelta + midDelta + botDelta) / 3;

serialPrintf(serialPort, "%s\n", buffer);
```

```
            memset(buffer, 0, sizeof buffer);

            namedWindow("Window", WINDOW_AUTOSIZE);
            moveWindow("Window", 70, 70);
            imshow("Window", frame);
            if (waitKey(10) >= 0) break;
        }

        destroyWindow("Window");
        serialClose(serialPort);
        return 0;
}
```

Приложение В

Программный код модификации №2 основного алгоритма

```
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/videoio.hpp>
#include <iostream>
#include <vector>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <unistd.h>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
	Mat frame, hsv, mask, blur;
	vector<int>lowerBlue = { 75, 60, 50 };
	vector<int>upperBlue = { 130, 255, 255 };
	vector<vector<Point>>contours;
	vector<Vec4i> hierarchy;
	int delta, width, angle;
	double area, maxArea = 12000;
	char buffer[5] = { 0 };
	RotatedRect rect;
	Point2f rectPoints[4];

	VideoCapture cap;
	cap.open(0);

	int serialPort = serialOpen("/dev/ttyUSB0", 9600);
```

```
if (serialPort == -1)
{
	cout << "Failed open port" << endl;
	return 0;
}

delay(1000);

for (;;)
{
	cap >> frame;
	if (frame.empty())
	{
		cout << "Camera error" << endl;
		break;
	}

	width = frame.size().width;
	cvtColor(frame, hsv, COLOR_BGR2HSV);
	inRange(hsv, lowerBlue, upperBlue, mask);
	findContours(mask, contours, hierarchy, RETR_TREE,
	CHAIN_APPROX_NONE);

	if (contours.empty())
	{
		cout << "Line was not detected" << endl;
		continue;
	}

	for (int i = 0; i < contours.size(); i++)
	{
		area = contourArea(contours[i]);
```

```
		if (area > maxArea)
		{
			drawContours(frame, contours, i, Scalar(0, 255, 255), 3);
			rect = minAreaRect(contours[i]);
		}
}

rect.points(rectPoints);
delta = width / 2 - rect.center.x;
angle = rect.angle;

if (rect.size.width < rect.size.height)
{
		angle += 90;
}

for (int j = 0; j < 4; j++)
{
		line(frame, rectPoints[j], rectPoints[(j + 1) % 4], Scalar(0, 0, 255), 3);
}

circle(frame, rect.center, 5, Scalar(0, 0, 255), -1);
circle(frame, Point(width / 2, rect.center.y), 5, Scalar(0, 255, 0), -1);

line(frame, rect.center, Point(width / 2, rect.center.y), Scalar(0, 255, 255), 2);

sprintf_s(buffer, "%d", angle);
putText(frame, "Angle:", Point(10, 30), FONT_HERSHEY_SIMPLEX, 1,
Scalar(0, 0, 255), 3);
putText(frame, buffer, Point(120, 30), FONT_HERSHEY_SIMPLEX, 1,
Scalar(0, 0, 255), 3);
memset(buffer, 0, sizeof buffer);
```

```
            sprintf_s(buffer, "%d", delta);
            putText(frame, "Delta:", Point(10, 80), FONT_HERSHEY_SIMPLEX, 1,
            Scalar(0, 0, 255), 3);
            putText(frame, buffer, Point(120, 80), FONT_HERSHEY_SIMPLEX, 1,
            Scalar(0, 0, 255), 3);

            serialPrintf(serialPort, "%s\n", buffer);
            memset(buffer, 0, sizeof buffer);

            namedWindow("Window", WINDOW_AUTOSIZE);
            moveWindow("Window", 70, 70);
            imshow("Window", frame);

            if (waitKey(10) >= 0) break;
        }

        destroyWindow("Window");
        serialClose(serialPort);
        return 0;
}
```