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

Кафедра Прикладная математика и информатика
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Управление корпоративными информационными процессами
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Исследование методов компьютерного зрения и разработка алгоритмов распознавания объектов окружающей среды в беспилотных транспортных средствах»

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

Научный руководитель к.т.н., доцент, Д.Г. Токарев
(ученая степень, звание, И.О. Фамилия)

Тольятти 2025

Оглавление

Введение

Данная научно-исследовательская работа посвящена разработке программных алгоритмов для обнаружения и распознавания объектов окружающей среды в беспилотных транспортных средствах с использованием методов компьютерного зрения.

Актуальность представленной магистерской диссертации обусловлена стремительным развитием робототехники и компьютерных технологий в последние годы, а также их широким применением в различных областях человеческой деятельности. Ярким примером этого является активное внедрение искусственного интеллекта в сфере логистики. Данный процесс характеризуется автоматизацией и роботизацией складов и логистических центров крупных компаний путем использования автоматически управляемых транспортных средств (Automated Guided Vehicles, AGV) и автономных мобильных роботов (Autonomous Mobile Robots, AMR) для перемещения грузов, навигация которых осуществляется при помощи систем компьютерного зрения. Данная научно-исследовательская работа направлена на разработку простых и эффективных программных алгоритмов, которые позволят беспилотным транспортным средствам ориентироваться в пространстве посредством распознавания дорожной разметки и направляющих дорожных знаков, а также обнаруживать пешеходов для предотвращения столкновений.

Проблема исследования: подавляющее большинство современных программных алгоритмов для обнаружения объектов окружающей среды в беспилотных транспортных средствах основаны на использовании машинного обучения и искусственных нейронных сетей, что ограничивает их применение в небольших маломощных автономных транспортных роботах за счет высоких требований к вычислительным мощностям бортового оборудования.

Объектом исследования является процесс обработки кадров видеопотока с бортовой камеры автономного транспортного средства программными средствами.

Предметом исследования являются методы компьютерного зрения для решения задач обнаружения и распознавания объектов окружающей среды на изображениях.

Целью исследования является разработка программных алгоритмов обнаружения и распознавания дорожной разметки, дорожных знаков и пешеходов в реальном времени на видео с бортовой камеры беспилотного транспортного средства, основанные на применении классических методов компьютерного зрения и не уступающие в быстродействии существующим нейросетевым решениям.

Гипотеза исследования: разработанные в рамках научной работы алгоритмы на основе классических методов компьютерного зрения достаточно точно обнаружат объекты окружающей среды на кадрах видеопотока с бортовой камеры автономного транспортного средства, а также продемонстрируют уровень быстродействия, сравнимый с показателями существующих программных решений на основе искусственных нейронных сетей.

В соответствии с поставленной целью были выделены следующие основные задачи научной работы:

- исследовать существующие методы компьютерного зрения для решения задач обнаружения и распознавания объектов на изображениях;
- разработать программный алгоритм обнаружения и распознавания дорожной разметки;
- разработать программный алгоритм обнаружения и распознавания дорожных знаков;
- разработать программный алгоритм обнаружения движения пешеходов для предотвращения столкновений;

– провести апробацию разработанных алгоритмов на видео с видеорегистраторов, оценить их быстродействие и сравнить с существующими нейросетевыми решениями.

При выполнении данной магистерской диссертации были последовательно пройдены следующие этапы:

– постановочный (постановка проблемы исследования, определение предмета и объекта исследования, изучение литературы в предметной области, формулировка цели, гипотезы и задач исследования);
– исследовательский (анализ существующих решений проблемы исследования, разработка собственного решения, формулировка предварительных выводов);
– заключительный (апробация разработанного решения, анализ полученных результатов, подтверждение или опровержение гипотезы).

Научная новизна исследования заключается в разработке эффективных алгоритмов обнаружения и распознавания дорожной разметки, дорожных знаков и пешеходов на видео в реальном времени на языке программирования C++ при помощи классических методов компьютерного зрения, которые были бы сравнимы по уровню быстродействия с существующими нейросетевыми программными решениями.

Теоретическая значимость исследования заключается в том, что положения и выводы, полученные в ходе написания магистерской диссертации, приведут к приращению научных знаний в области исследования компьютерного зрения, а также могут способствовать развитию научного интереса в изучении данного направления.

Практическая значимость исследования заключается в реализации теоретических аспектов в ходе практических экспериментов и в возможности применять результаты работы в будущих разработках в области компьютерного зрения.

Публикация по теме исследования:

Микряков П. С. «Обнаружение и распознавание линий дорожной разметки при помощи алгоритмов компьютерного зрения» / научный журнал «Вестник магистратуры», 2025. №1 (160) ISSN 2223-4047.

На защиту выносятся:

- программный алгоритм обнаружения и распознавания дорожной разметки;
- программный алгоритм обнаружения и распознавания дорожных знаков;
- программный алгоритм обнаружения движения пешеходов для предотвращения столкновений;

Представленная магистерская диссертация состоит из введения, четырех глав, заключения и списка используемой литературы и используемых источников.

Работа изложена на 79 страницах, содержит 58 рисунков, 2 таблицы и 30 источников.

Использованные в диссертации картинки и фрагменты видео находятся в свободном доступе, являются общественным достоянием или собственными изображениями.

Глава 1 Обзор современного состояния исследований в области компьютерного зрения

1.1 Понятие, цель и задачи компьютерного зрения

«Компьютерное зрение (Computer Vision) — это область искусственного интеллекта, которая занимается созданием программ и систем, позволяющих компьютерам анализировать и понимать визуальную информацию, такую как изображения и видео» [7].

Цель компьютерного зрения – получение полезной информации из входных визуальных данных, таких как изображения и видео, для выполнения поставленной задачи [10].

Основные задачи компьютерного зрения:

- обнаружение объектов. Данная задача состоит в том, чтобы на входном изображении идентифицировать и определить местоположение определенных объектов, например, положение пешехода или автомобиля на кадре с видеорегистратора. Процесс обнаружения объектов состоит из двух частей: локализации и классификации объектов.
- классификация изображений. Данная задача состоит в том, чтобы присвоить входному изображению метку или категорию, обнаружив и идентифицировав на нем один или несколько определенных объектов, таких как, например, человек, автомобиль и т.д.
- сегментация изображений. Данная задача состоит в том, чтобы выделить на входном изображении значимые области, отбросив сегменты, не содержащие полезную информацию, что значительно упростит дальнейший анализ.
- распознавание лица и фигуры человека. Данная задача состоит в том, чтобы на входном изображении обнаружить и

идентифицировать лицо или силуэт человека, а также определить такие ключевые характеристики, как его пол и возраст.

- восстановление изображений. Данная задача состоит в том, чтобы реконструировать старое, выцветшее или поврежденное изображение за счет удаления шума, размытости, повреждений.
- сравнение изображений. Данная задача состоит в том, чтобы найти на двух или нескольких входных изображениях схожие или идентичные элементы с целью выявления совпадений или различий между ними.
- реконструкция сцены. Данная задача состоит в том, чтобы создать трёхмерную модель реальной сцены, например комнаты, на основе набора двумерных изображений, полученных с разных ракурсов.
- трекинг. Данная задача состоит в том, чтобы отследить движение определенных объектов на последовательности кадров видеопотока с целью определения их положения, траектории и поведения во времени [19].

Обнаружение объектов является одной из ключевых задач компьютерного зрения и лежит в основе таких передовых систем, как беспилотные автомобили, роботы-курьеры, автоматизированные системы сортировки и транспортировки грузов на складах [11].

1.2 Актуальность задачи обнаружения и распознавания объектов окружающей среды в беспилотных транспортных средствах

Актуальность данного исследования обусловлена стремительным развитием и внедрением беспилотных транспортных средств в сфере логистики.

Ярким примером данной тенденции является корпорация Amazon. На сегодняшний день количество роботизированных систем, функционирующих на складах и в логистических центрах компании, составляет около 750 тысяч

единиц, и это число продолжает стремительно увеличиваться [13]. Большинство систем представляют собой автономные мобильные роботы для транспортировки грузов, использующие для своей навигации технологии компьютерного зрения. Самой узнаваемой и распространенной моделью в арсенале торгового гиганта является оранжевый робот Kiva (рисунок 1), который за последние годы успел стать своего рода символом стремления Amazon к автоматизации и роботизации [23].

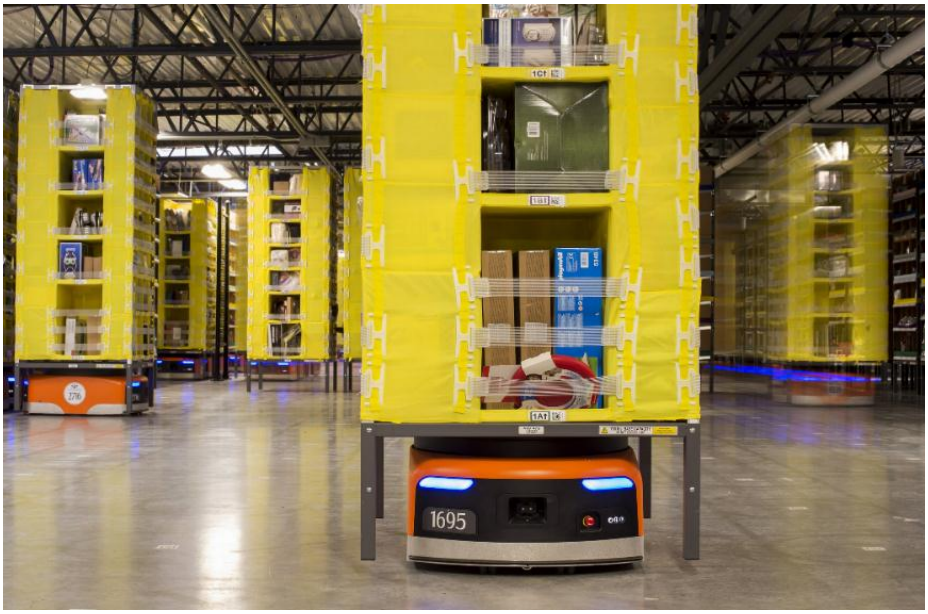

Рисунок 1 – Робот Kiva компании Amazon

Другим примером оптимизации производственных процессов посредством внедрения беспилотных транспортных средств стал отечественный автоконцерн АвтоВАЗ [3]. Так, на производстве появились роботизированные тележки, получившие название «Антонина». Принцип их работы основан на применении систем компьютерного зрения и искусственного интеллекта (рисунок 2). Основной задачей данного автоматического напольного транспорта (АНТ) стала транспортировка деталей и инструментов внутри цехов предприятия.

Рисунок 2 – Автоматическая тележка «Антонина»

Так как АвтоВАЗ имеет разветвленную сеть дорог, снабженных разметкой, дорожными знаками и пешеходными переходами, добавление алгоритмов распознавания объектов окружающей среды, например, позволило бы использовать «Антонин» для транспортировки грузов между различными цехами и производственными корпусами на всей территории завода, что значительно бы увеличило их мобильность и эффективность.

Еще одним примером может выступить Яндекс, который уже несколько лет применяет для транспортировки заказов роботов-доставщиков, обслуживающих несколько определенных районов Москвы (рисунок 3) [5]. Алгоритмы распознавания разметки, дорожных знаков и пешеходов являются неотъемлемой частью системы ориентации в пространстве этих беспилотных транспортных средств.

Рисунок 3 – Робот-доставщик компании Яндекс

Программные алгоритмы обнаружения объектов окружающей среды также могут быть внедрены и в так называемых «коридорных» роботов. Эти автономные транспортные средства предназначены для перемещения документов, посылок, мелких грузов и оборудования по внутренним маршрутам здания, чаще всего — в офисах, больницах, архивах и административных учреждениях, что позволяет автоматизировать логистику между отделами и делает их частью системы управления корпоративными информационными процессами. Примерами таких роботов являются южнокорейский GAEMI (рисунок 4) и американский TUG (рисунок 5). Распознавание разметки, дорожных знаков и пешеходов позволит увеличить мобильность таких автономных транспортных средств и даст возможность осуществлять транспортировку полезных грузов и документации между отдельными зданиями или корпусами предприятия.

Рисунок 4 – «Коридорный» робот GAEMI

Рисунок 5 – «Коридорный» робот TUG

Выводы по главе 1

В главе 1 магистерской диссертации было дано определение компьютерному зрению, рассмотрены его цель и задачи; была подтверждена актуальность задачи обнаружения и распознавания объектов окружающей среды в беспилотных транспортных средствах; были приведены примеры участия автономных транспортных роботов в управлении корпоративными информационными процессами.

Глава 2 Анализ методов компьютерного зрения для решения задачи обнаружения объектов на изображении

2.1 Традиционные методы компьютерного зрения

«Традиционное» компьютерное зрение представляет собой набор методов для обработки и анализа изображений и видео, которые используют в своей работе классические алгоритмы и ручное выделение признаков для извлечения информации из визуальных данных.

Традиционные методы компьютерного зрения можно условно разделить на две большие категории в зависимости от их назначения:

- предварительная обработка изображений;
- выделение признаков.

Рассмотрим наиболее известные, распространенные и эффективные методы, представляющие собой основу «традиционного» компьютерного зрения.

2.1.1 Предварительная обработка изображений

Предварительная обработка представляет собой процесс подготовки визуальных данных для дальнейшего анализа. Целю данного этапа является улучшение качества рассматриваемого изображения, устранение шума, выделение ключевых элементов и регионов интереса, а также приведение изображения к формату, который лучше всего подходит для решения поставленной задачи.

Методы компьютерного зрения, использующиеся для предварительной обработки изображений, также могут быть разделены на несколько категорий в зависимости от их функций:

- преобразование изображений;
- улучшение качества изображений;
- уменьшение шума;
- морфологические операции.

2.1.1.1 Преобразование изображений

К методам компьютерного зрения, направленным на преобразование входных визуальных данных, относятся следующие техники и подходы:

Смещение изображения. Данный метод представляет собой геометрическое преобразование, при котором все пиксели изображения перемещаются на определенное расстояние по горизонтали и/или вертикали. Пример использования описанного метода представлен на рисунке 6.

Рисунок 6 – Смещение изображения

Отражение изображения. Данный метод представляет собой геометрическое преобразование, при котором изображение «зеркально» отражается относительно заданной оси (горизонтальной, вертикальной или обеих). Пример использования описанного метода представлен на рисунке 7.

Рисунок 7 – Отражение изображения

Масштабирование изображения. Данный метод представляет собой геометрическое преобразование, при котором изображение изменяет свой размер за счет увеличения или уменьшения ширины и высоты. Пример использования описанного метода представлен на рисунке 8.

Рисунок 8 – Масштабирование изображения

Поворот изображения. Данный метод представляет собой геометрическое преобразование, при котором изображение поворачивается на определенный угол вокруг заданной точки. Пример использования описанного метода представлен на рисунке 9.

Рисунок 9 – Поворот изображения

Обрезка изображения. Данный метод представляет собой преобразование, при котором сохраняется лишь определенная область изображения, а все что лежит за ее пределами удаляется. Пример использования описанного метода представлен на рисунке 10.

Рисунок 10 – Обрезка изображения

Преобразование перспективы. Данный метод представляет собой геометрическое преобразование, которое имитирует эффект изменения точки зрения или наклона камеры. Преобразование перспективы описывается с помощью матрицы гомографии, которая имеет размеры 3x3 и связывает координаты точек на исходном кадре с новыми координатами на целевом изображении. Пример использования описанного метода представлен на рисунке 11.

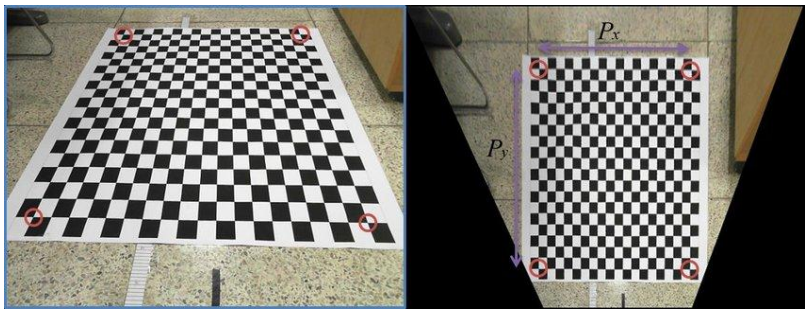

Рисунок 11 – Преобразование перспективы

2.1.1.2 Улучшение качества изображений

К методам компьютерного зрения, направленным на улучшение качества визуальных данных, относятся следующие техники и подходы:

Регулировка яркости и контраста. Данный метод представляет собой преобразование, направленное на улучшение визуальных характеристик

изображения посредством изменения интенсивности пикселей. Пример использования описанного метода представлен на рисунке 12.

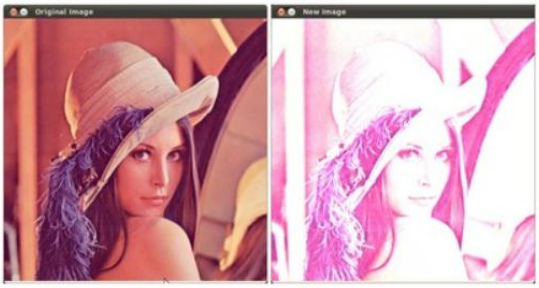

Рисунок 12 – Регулировка яркости и контраста

Увеличение резкости. Данный метод представляет собой преобразование, направленное на улучшение визуальных характеристик изображения за счет усиления контуров объектов и границ между областями с разной интенсивностью. Пример использования описанного метода представлен на рисунке 13.

Рисунок 13 – Увеличение резкости

Инверсия. Данный метод представляет собой преобразование, направленное на улучшение визуальных характеристик изображения за счет изменения значений пикселей на противоположные. Пример использования описанного метода представлен на рисунке 14.

Рисунок 14 – Инверсия

Выравнивание гистограммы. Данный метод представляет собой преобразование, направленное на улучшение визуальных характеристик изображения за счет равномерного перераспределения интенсивностей пикселей с целью повышения контраста и улучшения видимости отдельных деталей и объектов. Пример использования описанного метода представлен на рисунке 15.

Рисунок 15 – Выравнивание гистограммы

Преобразование цветового пространства. Данный метод представляет собой преобразование, направленное на улучшение визуальных характеристик изображения за счет изменения модели представления цвета. Наиболее популярными цветовыми пространствами являются модели RGB, которая описывает все многообразие оттенков комбинациями трех основных цветов (красного, зеленого и синего), и HSV, в которой координатами

выступают тон, насыщенность и яркость. Пример использования описанного метода представлен на рисунке 16.

Рисунок 16 – Преобразование цветового пространства

2.1.1.3 Уменьшение шума

К методам компьютерного зрения, направленным на уменьшение шума, присутствующего во входных визуальных данных, относятся следующие техники и подходы:

Медианный фильтр. Работа данного метода состоит из следующих шагов:

1. Программа выбирает пиксель и рассматривает его окрестности (обычно это квадратная область размерами 3x3 или 5x5).
2. Программа сортирует значения всех соседних пикселей в заданной области и выбирает медианное (усредненное) значение.
3. Значение изначального центрального пикселя заменяется на медианное.

Данная процедура проделывается для всех пикселей изображения.

Пример использования описанного метода представлен на рисунке 17.

Рисунок 17 – Медианный фильтр

Фильтр Гаусса. Работа данного метода состоит из следующих шагов:

1. Программа выбирает пиксель и рассматривает его окрестности (обычно это квадратная область размерами 3x3 или 5x5).
2. Программа определяет взвешенную сумму значений соседних пикселей в заданной области на основе специальной матрицы, веса в которой вычисляются на основе гауссовой функции.
3. Значение изначального центрального пикселя заменяется на вычисленное.

Данная процедура проделывается для всех пикселей изображения.

Пример использования описанного метода представлен на рисунке 18.

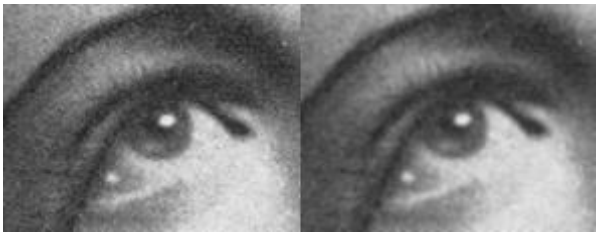

Рисунок 18 – Фильтр Гаусса

2.1.1.4 Морфологические операции

Морфологические операции представляют собой набор методов, которые используются для обработки формы объектов на бинарных или полутоновых изображениях. Данные преобразования основаны на теории

множеств и в своей работе используют особые структурные элементы (ядро или маску), которые определяют область их действия. К морфологическим операциям относятся следующие техники и подходы:

Эрозия. Данный метод обычно применяется к бинарным изображениям, пиксели которых имеют всего два значения: 0 (черный, фон) и 1 (белый, объект). Эрозия используется для уменьшения размера объектов, а также для удаления мелких деталей или разделения соединенных элементов. Ее работа состоит из следующих шагов:

1. Программа выбирает пиксель и рассматривает его окрестности (обычно это квадратная область размерами 3x3 или 5x5).
2. Если все пиксели в заданной области имеют значение 1 и принадлежат объекту, то центральный пиксель также принимает значение 1 и остается частью объекта.
3. Если хотя бы один пиксель в заданной области имеет значение 0 и принадлежит фону, то центральный пиксель также принимает значение 0 и становится частью фона.

Данная процедура проделывается для всех пикселей изображения.

Пример использования описанного метода представлен на рисунке 19.

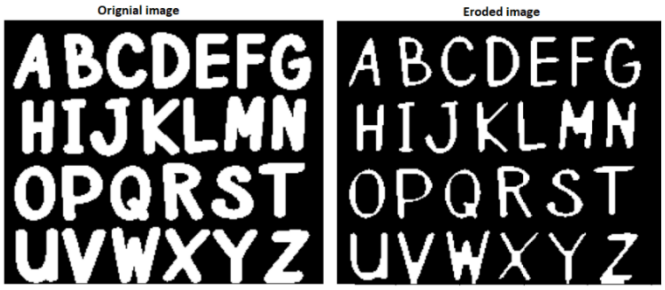

Рисунок 19 – Эрозия

Дилатация. Данный метод обычно применяется к бинарным изображениям, пиксели которых имеют всего два значения: 0 (черный, фон) и 1 (белый, объект). Дилатация используется для увеличения размера объектов,

а также для заполнения мелких отверстий или соединения близко расположенных элементов. Ее работа состоит из следующих шагов:

1. Программа выбирает пиксель и рассматривает его окрестности (обычно это квадратная область размерами 3x3 или 5x5).
2. Если хотя бы один пиксель в заданной области имеет значение 1 и принадлежит объекту, то центральный пиксель также принимает значение 1 и добавляется к объекту.

Данная процедура проделывается для всех пикселей изображения.

Пример использования описанного метода представлен на рисунке 20.

Рисунок 20 – Дилатация

Открытие. Данный метод является комбинацией двух описанных выше морфологических операций: эрозии и дилатации и состоит из двух шагов:

1. Программа уменьшает размер объектов с помощью эрозии, при этом удаляя мелкие детали и шум.
2. Размеры основных объектов восстанавливаются при помощи дилатации.

Пример использования описанного метода представлен на рисунке 21.

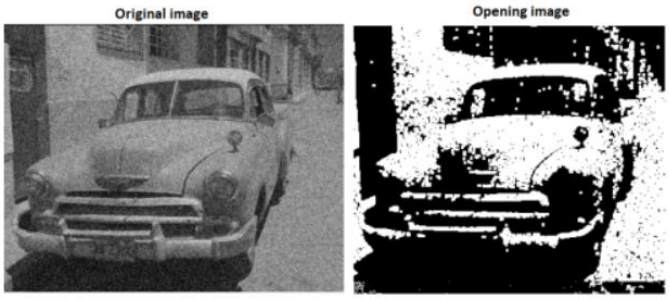

Рисунок 21 – Открытие

Закрытие. Данный метод также является комбинацией двух описанных выше морфологических операций: эрозии и дилатации и состоит из двух шагов:

1. Программа увеличивает размер объектов с помощью дилатации, при этом заполняя мелкие отверстия и соединяя близко расположенные элементы.
2. Размеры основных объектов восстанавливаются при помощи эрозии.

Пример использования описанного метода представлен на рисунке 22.

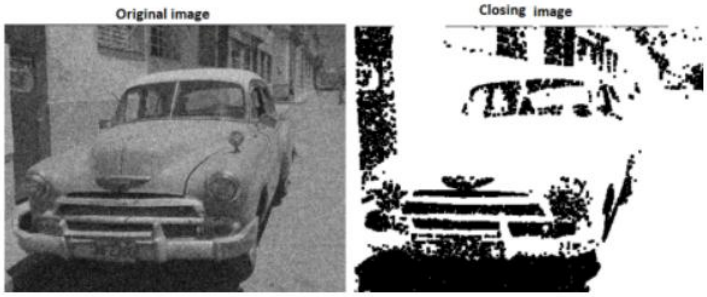

Рисунок 22 – Закрытие

2.1.2 Выделение признаков

Выделение признаков представляет собой процесс извлечения ключевой информации из входных визуальных данных. Целю данного этапа является нахождение определенных характеристик, таких как углы, границы или цвет, характеризующих искомые объекты в кадре [12]. К методам

компьютерного зрения, осуществляющим выделение признаков, относятся следующие техники и подходы:

Пороговая фильтрация. Данный метод представляет собой базовую операцию по бинаризации изображения путем присвоения пикселям значений 1 или 0 в зависимости от того, попадают ли они в заданный цветовой диапазон. Пороговая фильтрация широко применяется для сегментации объектов и выделения областей интереса. Пример использования описанного метода представлен на рисунке 23.

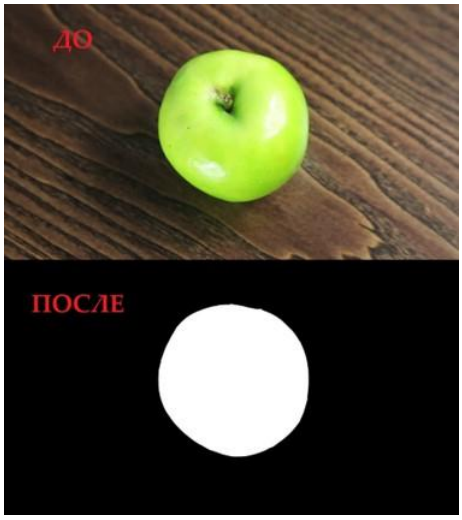

Рисунок 23 – Пороговая фильтрация

Детектор границ Кэнни. Данный метод, разработанный в 1986 году Джоном Кэнни, представляет собой оператор обнаружения краев изображения на основе многоступенчатого алгоритма, включающего сглаживание кадра с помощью фильтра Гаусса, вычисление градиентов, подавление локальных не-максимумов, двойную пороговую фильтрацию и трассировку границ с гистерезисом [15]. Пример использования описанного метода представлен на рисунке 24.

Рисунок 24 – Детектор границ Кэнни

Гистограмма направленных градиентов (Histogram of Oriented Gradients, HOG). Данный метод представляет собой нахождение направлений и величин градиентов яркости (изменений интенсивности) в локальных областях изображения, которые содержат важную информацию о границах и форме объектов в кадре [18]. Пример использования описанного метода представлен на рисунке 25.

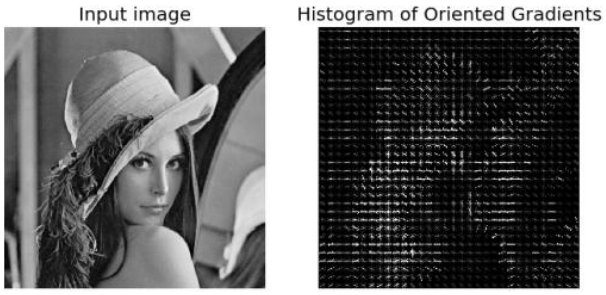

Рисунок 25 – Гистограмма направленных градиентов

Масштабно-инвариантная трансформация признаков (Scale-Invariant Feature Transform, SIFT). Данный метод основан на поиске ключевых точек на изображении, устойчивых к изменению масштаба, поворотам, смене

освещения и небольшим искажениям. Пример использования описанного метода представлен на рисунке 26.

Рисунок 26 – Масштабно-инвариантная трансформация признаков

Speeded-Up Robust Features (SURF). Данный метод представляет собой детектор локальных характеристик изображения, представленный Гербертом Бэем в 2006 году. SURF был разработан как более быстрая и эффективная альтернатива SIFT. Он так же имеет инвариантность к изменению масштаба, поворотам и смене освещения, но при этом работает значительно быстрее своего предшественника благодаря оптимизированным вычислениям [14]. Пример использования описанного метода представлен на рисунке 27.

Рисунок 27 – SURF

2.2 Методы компьютерного зрения на основе глубокого обучения

Глубокое обучение (Deep learning) – это вид машинного обучения, в основе которого лежит использование искусственных нейронных сетей, способных самообучаться на большом наборе данных.

Суть глубокого обучения состоит в том, что искусственный интеллект самостоятельно находит решение поставленной задачи, учится на собственных ошибках и после каждой итерации обучения выдает более точный результат.

Кратко рассмотрим основные компоненты современных искусственных нейронных сетей:

- нейроны. Нейрон – это базовая единица искусственной нейронной сети, которая принимает входные данные, обрабатывает их с помощью функции активации и передает результат на следующий слой.
- слои. Искусственная нейронная сеть обычно состоит из нескольких слоев, которые делятся на три категории: входной слой, скрытые слои и выходной слой. Входной слой принимает данные, скрытые слои обрабатывают их, а выходной слой выдает конечный результат. Количество скрытых слоев и нейронов в них может варьироваться в зависимости от поставленной задачи, и их правильная настройка напрямую влияет на эффективность модели.
- функции активации. Функции активации определяют, будет ли нейрон активен или нет. Они добавляют нелинейность в модель, позволяя искусственной нейронной сети обучаться более сложным паттернам.
- веса. Каждая связь между нейронами имеет вес, который определяет важность этой связи. Веса могут быть положительными или отрицательными, что позволяет искусственной нейронной сети усиливать или ослаблять определенные сигналы.

– функция потерь. Функция потерь измеряет, насколько хорошо искусственная нейронная справляется с поставленной задачей, и показывает разницу между предсказаниями и реальными целевыми значениями.
– оптимизаторы. Оптимизатор представляет собой метод, используемый для настройки весов модели с целью снижения функции потерь.
– архитектура искусственной нейронной сети. Архитектура определяет, как будут соединяться слои и нейроны, какие функции активации будут использоваться и как будет производиться обучение.
– наборы обучающих данных. Данные для обучения искусственной нейронной сети формируются из обучающей выборки. Их качество и разнообразие оказывают непосредственное влияние на эффективность модели.

Выделим наиболее популярные и эффективные методы распознавания объектов в кадре, использующие искусственные нейронные сети:

RCNN (Region-based Convolutional Neural Network) – это архитектура глубокого обучения, специально разработанная для задач обнаружения объектов на изображениях [20].

Основные этапы работы RCNN:

1. Выделение регионов-кандидатов. На первом этапе RCNN использует алгоритм Selective Search, объединяющий соседние пиксели изображения на основе их цвета, для генерации множества регионов-кандидатов, представляющих собой потенциальные области кадра, где могут находиться искомые объекты.
2. Извлечение признаков. На втором этапе каждый регион-кандидат принимает фиксированный размер и передается в предобученную сверточную нейронную сеть, которая извлекает признаки, создавая особые векторы.

3. Классификация объектов. Векторы признаков передаются в классификатор, который определяет, содержится ли в регионе объект и к какому классу он принадлежит.
4. Уточнение границ. Для повышения точности локализации объектов используется регрессия границ, которая позволяет уточнить координаты ограничивающего прямоугольника вокруг объекта.

Fast RCNN (Fast Region-based Convolutional Neural Network) представляет собой улучшенную версию архитектуры RCNN. В отличие от модели-предшественницы, где каждый регион-кандидат обрабатывается отдельно, Fast RCNN изначально пропускает все изображение через сверточную нейронную сеть, создавая общую карту признаков, что значительно повышает скорость работы [6].

Faster RCNN (Faster Region-based Convolutional Neural Network) является улучшенной версией Fast RCNN. Основное нововведение Faster RCNN заключается в использовании еще одной небольшой искусственной нейронной сети (Region Proposal Network) для генерации регионов-кандидатов, что еще больше ускоряет процесс обнаружения объектов на изображении.

YOLO (You Only Look Once) – это современная архитектура глубокого обучения. В отличие более ранних методов, таких как RCNN, Fast RCNN и Faster RCNN, которые обрабатывают изображение в несколько этапов, YOLO выполняет обнаружение объектов за один проход через нейронную сеть. Это делает YOLO значительно быстрее, что позволяет использовать его в реальном времени.

Основные этапы работы YOLO:

1. Разделение изображения на сетку. Входное изображение разделяется на фиксированное количество ячеек, каждая из которых отвечает за обнаружение объектов внутри своих границ.

2. Прогнозирование объектов. Каждая ячейка сети предсказывает наличие объектов, а также их характеристики: координаты ограничивающего прямоугольника, степень уверенности, классы.
3. Объединение предсказаний. YOLO объединяет предсказания всех ячеек сетки и оставляет только те ограничивающие прямоугольники, которые имеют наибольшую степень уверенности в наличии объекта.

2.3 Сравнение методов компьютерного зрения

Рассмотрим основные достоинства и недостатки рассмотренных подходов.

Среди основных преимуществ традиционных методов компьютерного зрения можно выделить следующие:

- интерпретируемость. Традиционные методы построены вокруг ручного извлечения признаков. Логику работы программ на их основе легко проследить. Это дает возможность разобраться в том, какие именно признаки используются для принятия тех или иных решений.
- низкие требования к вычислительным мощностям. Традиционные методы не требуют больших вычислительных мощностей и могут работать на одном центральном процессоре без привлечения графического процессора.
- работа с небольшим количеством данных. Традиционные методы могут эффективно работать, оперируя малыми наборами данных.
- простота реализации. Многие классические алгоритмы достаточно легко реализовать и настроить, не имея глубоких знаний в программировании.

Перечислим недостатки традиционных методов компьютерного зрения:

– низкая точность при решении трудных задач. Признаки, извлеченные вручную, зачастую не учитывают сложные закономерности в данных, что ограничивает их применимость к решению комплексных задач.
– ограниченная масштабируемость и гибкость. Классические алгоритмы обычно нацелены на выполнение одной конкретный задачи и оперируют ограниченным набором данных, поэтому они плохо приспособлены для решения комплексных проблем и работы с большими объемами информации.

Среди основных преимуществ методов компьютерного зрения, основанных на глубоком обучении, можно выделить следующие:

– высокая точность при решении трудных задач. Благодаря многоступенчатому обучению и большому объему данных искусственные нейронные сети обеспечивают высокую точность при решении комплексных задачах.
– масштабируемость и гибкость. Искусственные нейронные сети хорошо настраиваются для решения самых разнообразных задач любой сложности.

Перечислим недостатки методов компьютерного зрения, основанных на глубоком обучении:

– высокие требования к вычислительным мощностям. Для обучения искусственных нейронных сетей требуется использование мощных графических процессоров и большого объема памяти, что делает их довольно дорогостоящими в использовании.
– работа только с большим количеством данных. Искусственные нейронные сети требуют больших объемов размеченных данных для обучения. Малое количество данных может привести к так называемому «переобучению», когда модель демонстрирует высокую точность на обучающих данных, но плохо работает с новой информацией.

– плохая интерпретируемость. Искусственные нейронные сети работают как «черные ящики», что затрудняет понимание того, почему программа приняла то или иное решение.
– сложность реализации. Для разработки и настройки искусственной нейронной сети требуются глубокие знания в области машинного обучения, а также хорошие навыки программирования [21].

Обобщим особенности применения рассмотренных методов компьютерного зрения в Таблице 1.

Таблица 1 – Сравнение методов компьютерного зрения

Аспекты сравнения	Традиционные методы компьютерного зрения	Методы на основе глубокого обучения
Извлечение признаков	Ручное, на основе алгоритмов, написанных программистом	Автоматическое, на основе обучения на большом объеме данных
Требования к количеству данных	Могут работать с ограниченным объемом данных	Требуют большой объем размеченных данных для обучения
Требования к вычислительным мощностям	Низкие; могут работать на одном центральном процессоре	Высокие; требуют использования графических процессоров
Эффективность и точность	Хорошо справляются с простыми задачами, однако имеют низкую точность при решении сложных комплексных задач	Одинаково хорошо справляются как с простыми, так и со сложными задачами, всегда демонстрируя высокую точность
Интерпретируемость	Хорошая; логику их работы можно без проблем проследить и понять	Плохая; работают как «черные ящики»
Масштабируемость и гибкость	Плохая; обычно предназначены для решения одной конкретной задачи	Хорошая; могут быть использованы для решения широкого спектра задач любой сложности
Сложность реализации	Средняя; не требуют глубоких знаний и продвинутых навыков программирования	Высокая; требуют глубоких знаний в области машинного обучения и хороших навыков программирования
Адаптируемость	Плохо адаптируются к изменениям условий	Хорошо адаптируются к изменениям условий при наличии соответствующих обучающих данных
Возможность работы в	Хорошо подходят для	Могут работать в режиме

режиме реального времени	работы в реальном времени на не очень мощном оборудовании	реального времени, но требуют мощное оборудование

Таким образом, традиционные методы компьютерного зрения подходят для решения простых конкретных задач при ограниченных объемах данных в условиях недостатка высококвалифицированных программистов и отсутствия высокопроизводительного оборудования. Модели глубокого обучения же хорошо справляются с широким спектром комплексных задач, требующих высокой точности и адаптируемости к изменяющимся условиям, однако предъявляют высокие требования к вычислительным ресурсам, знаниям и навыкам разработчиков, объемам и качеству размеченных обучающих данных [4].

Выводы по главе 2

В главе 2 магистерской диссертации были рассмотрены наиболее популярные и эффективные методы компьютерного зрения в контексте решения задачи распознавания объектов на изображении: традиционные методы и методы, основанные на глубоком обучении; были выявлены преимущества и недостатки обоих подходов, проведено их сравнение.

Глава 3 Разработка программных алгоритмов распознавания объектов окружающей среды в беспилотных транспортных средствах

3.1 Выбор инструментов разработки

3.1.1 Классические алгоритмы компьютерного зрения

На основании проведенного в прошлой главе сравнения классических и нейросетевых методов компьютерного зрения можно сделать вывод, что в качестве основных подходов при разработке программных алгоритмов распознавания объектов окружающей среды для небольших и маломощных беспилотных транспортных средств, используемых в складской логистике, целесообразно использовать традиционные методы компьютерного зрения. Данное решение обусловлено следующими причинами:

- низкие требования к вычислительным ресурсам. Традиционные методы не требуют для своей работы мощных и дорогостоящих процессоров, в отличие от искусственных нейронных сетей, которые нуждаются в значительных вычислительных ресурсах во время обучения. Это особенно важно в контексте применения разрабатываемых алгоритмов в автоматически управляемых транспортных средствах (Automated Guided Vehicles, AGV) и автономных мобильных роботах (Autonomous Mobile Robots, AMR), которые зачастую построены на базе не самых производительных микроконтроллеров или одноплатных компьютеров.
- высокая скорость работы. Классические алгоритмы работают быстрее, чем искусственные нейронные сети, особенно на слабом «железе», что критично для систем, требующих обработки данных в реальном времени, таких как беспилотные транспортные средства, которые должны быстро реагировать на изменения в окружающей среде, чтобы избежать столкновения или потери маршрута.

– простота реализации и настройки. Решения на основе традиционных методов проще реализовать и настроить, особенно если объекты для распознавания имеют четкие признаки, например, дорожные знаки и линии разметки. Кроме того, для их работы не требуется сложная инфраструктура или большие объемы маркированных обучающих данных.
– стабильность в контролируемых условиях. Если беспилотные транспортные средства работают в контролируемой среде, например, на складе с фиксированным освещением, заранее определенными маршрутами перемещения и известным расположением объектов и препятствий, классические алгоритмы могут показывать сопоставимо высокую точность и надежность в сравнении со значительно более дорогостоящими моделями на основе глубокого обучения.

3.1.2 Язык программирования C++

C++ представляет собой высокоуровневый компилируемый язык программирования общего назначения, который подходит для создания самых разнообразных приложений. Главными достоинствами C++ является его высокая производительность и скорость, которые достигаются благодаря нескольким факторам.

Во-первых, C++ относится к категории компилируемых языков программирования. Это означает, что программа, написанная на данном языке, сначала проходит этап компиляции, в ходе которого она преобразуется компилятором в машинный код и сохраняется в виде исполняемого файла. Такой файл затем напрямую взаимодействует с аппаратной составляющей, что обеспечивает высокую скорость выполнения программы. Интерпретируемые языки программирования, такие как Python, функционируют иначе. Программы, написанные на этих языках, не компилируются заранее, а преобразуются в машинный код уже

непосредственно в процессе выполнения с помощью интерпретатора, что приводит к снижению скорости их работы.

Во-вторых, C++ относится к языкам программирования с сильной статической типизацией. Это означает, что проверка соответствия данных указанным типам осуществляется на этапе компиляции. Следовательно, программист должен заранее определять типы переменных до их использования. Такой подход делает синтаксис более строгим, но при этом помогает уменьшить количество ошибок и ускорить выполнение программ. Сильная типизация также не допускает смешения различных типов данных в выражениях и не позволяет выполнять их автоматическое неявное преобразование. Это требует от разработчиков большей внимательности при написании кода, но также способствует увеличению скорости выполнения программ.

В-третьих, унаследовав от языка C широкий спектр возможностей для управления памятью, C++ дает возможность программистам самостоятельно определять, где и как долго хранить данные, а также контролировать низкоуровневые ресурсы, такие как регистры процессора и кэш. В отличие от других языков программирования, C++ не имеет автоматической сборки мусора, что способствует повышению производительности, поскольку исключаются затраты времени на удаление неиспользуемых объектов. Однако такая особенность возлагает на разработчиков дополнительную ответственность за управление памятью, включая ее выделение и своевременное освобождение, что требует повышенного внимания при написании кода.

Кроме того, C++ является универсальным языком программирования. Множество библиотек и инструментов разработки, адаптированных почти под все распространенные операционные системы, позволяет с легкостью переносить написанные на этом языке программы с одной платформы на другую.

Таким образом, быстрый, универсальный и производительный язык программирования C++ отлично подходит для реализации программной части данной магистерской диссертации.

3.1.3 Библиотека компьютерного зрения OpenCV

OpenCV (Open Source Computer Vision Library) представляет собой библиотеку компьютерного зрения с открытым исходным кодом. Основная ее реализация выполнена на C/C++, однако существуют интерфейсы и для других популярных языков программирования, включая Python, Java, MATLAB и др. Библиотека совместима с различными операционными системами, такими как Windows, Linux, macOS, iOS и Android. OpenCV распространяется на условиях лицензии BSD, что позволяет свободно использовать ее как в приложениях с открытым исходным кодом, так и в закрытых коммерческих проектах [27].

«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.).[9]»

Структура библиотеки OpenCV показана на рисунке 28.

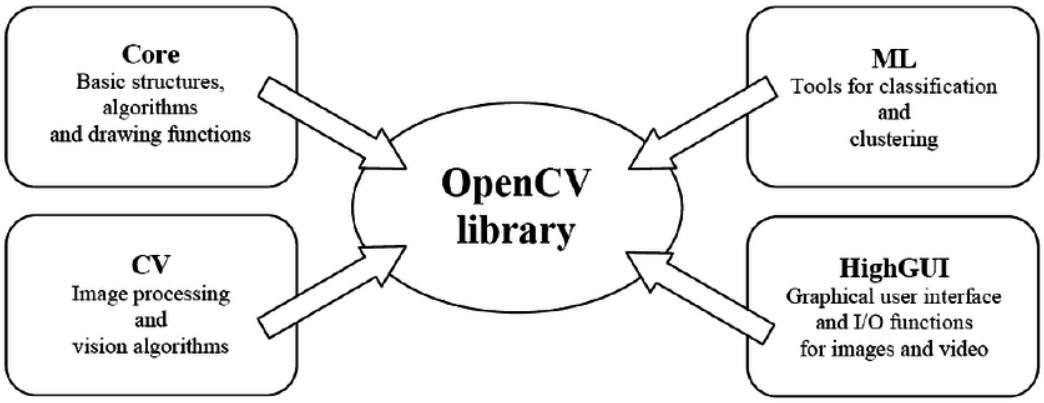

Рисунок 28 – Структура библиотеки OpenCV

Таким образом, богатый набор функций, универсальность и популярность у разработчиков по всему миру делают OpenCV отличным инструментом для реализации программной части данной магистерской диссертации [2].

3.1.4 Среда разработки Microsoft Visual Studio

Visual Studio представляет собой интегрированную среду разработки (Integrated Development Environment, IDE) для создания программного обеспечения, выпущенную компанией Microsoft.

Среди преимуществ данной IDE можно выделить:

Универсальность. Visual Studio поддерживает широкий спектр языков программирования (C++, C#, Python, Visual Basic, JavaScript/TypeScript и др.) и платформ (Windows, macOS, Linux, Android, iOS), позволяет разрабатывать десктопные, мобильные и облачные приложения, игры с использованием Unity или Unreal Engine, системное ПО.

Богатый набор инструментов. Visual Studio предоставляет разработчикам широкий спектр инструментов для создания, отладки, тестирования и развертывания приложений:

- IntelliSense – система автоматического дополнения кода, которая ускоряет написание программ и снижает количество ошибок;
- встроенный отладчик – поддерживает пошаговое выполнение программ, создание точек останова и анализ переменных;
- рефакторинг – автоматическое улучшение структуры кода;
- профилирование и диагностика для анализа производительности и поиска узких мест;
- поддержка тестирования (модульного, нагрузочного и автоматизированного);
- интеграция с Git – управление версиями и совместная работа над кодом.

Интеграция с экосистемой Microsoft. Visual Studio тесно связана с другими продуктами и сервисами Microsoft, что позволяет использовать современные технологии компании для создания инновационных решений:

- .NET и .NET Core – поддержка разработки кроссплатформенных приложений;
- Azure – упрощенное создание, тестирование и развертывание облачных приложений;
- SQL Server – работа с базами данных через Entity Framework и другие ORM;
- Windows SDK – разработка приложений для Windows с использованием последних API;
- Office 365 – создание надстроек для Office и интеграция с облачными сервисами.

Большое сообщество. Visual Studio имеет огромное сообщество разработчиков по всему миру. Опытные программисты активно дают советы, создают руководства и видеоуроки, разрабатывают и поддерживают

расширения, которые добавляют в Visual Studio новые функции, тогда как специалисты Microsoft активно взаимодействует с пользователями, учитывая их предложения и пожелания при обновлении IDE. Благодаря этому новички всегда могут рассчитывать найти помощь и поддержку на форумах, в блогах и социальных сетях. Все это делает Visual Studio не только мощным инструментом, но и удобной средой для обучения и профессионального роста.

Таким образом, все вышеперечисленные достоинства делают Microsoft Visual Studio отличным инструментом для реализации программной части данной магистерской диссертации.

3.2 Разработка программных алгоритмов

3.2.1 Разработка модуля распознавания дорожной разметки

Алгоритм работы программного модуля обнаружения и распознавания дорожной разметки состоит из 11 последовательных этапов:

1. Подключение всех необходимых для работы программы заголовочных файлов и библиотек и объявление переменных.
2. Инициализация камеры.
3. Захват изображения, поступающего с камеры.
4. Проверка успешности захвата кадра.
5. Изменение разрешения кадра.
6. Изменение перспективы на вид с высоты птичьего полета.
7. Преобразование полученного кадра из цветового пространства BGR в цветовую модель HSV.
8. Цветовая сегментация и извлечение бинарного изображения с идентифицированной разметкой.
9. Расчет гистограммы кадра.
10. Применение метода скользящих окон.
11. Получение направляющих точек.

Блок-схема описанного выше алгоритма работы программы представлена на рисунке 29.

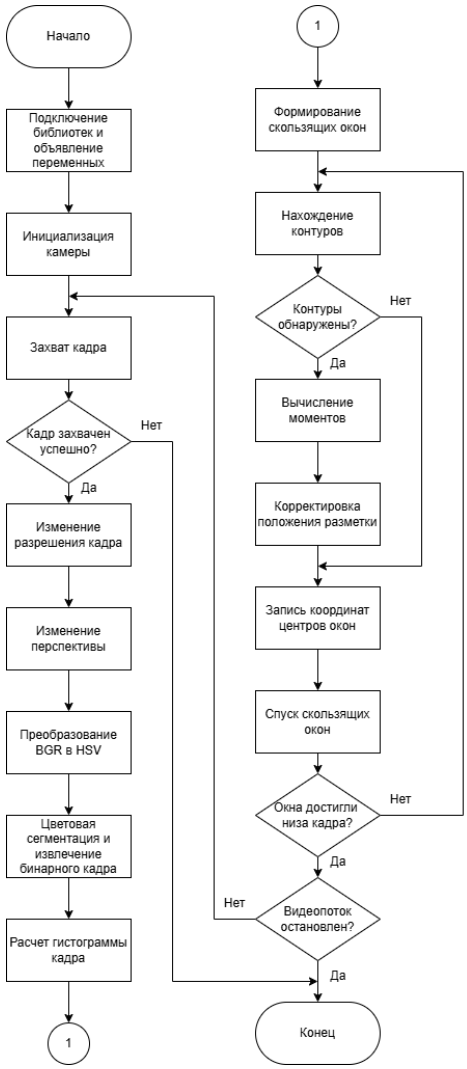

Рисунок 29 – Блок-схема алгоритма работы программного модуля распознавания дорожной разметки

Проведем детальный разбор кода программы.

Первым делом производится подключение всех необходимых для работы заголовочных файлов и библиотек:

– 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;

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

Далее происходит инициализация камеры с помощью объекта cap класса VideoCapture библиотеки OpenCV, предназначенного для захвата видео и чтения видеофайлов. Так как в ходе разработки и апробации программного модуля вместо реального устройства использовались видеозаписи с автомобильных видеорегистраторов, взятые из открытых источников, то аргументом для объекта cap будет являться путь к тому или иному видеофайлу, хранящемуся на локальном диске компьютера. Структура записи, отвечающей за это, в коде выглядит примерно следующим образом:

```
VideoCapture cap("D:\\...\\VideoName.mp4");
```

После этого проводится проверка успешности открытия видео при помощи метода isOpened, который возвращает значение true, если видеофайл был открыт успешно, или false, если открыть видеофайл по каким-то причинам не удалось, записывая его в контрольную переменную типа bool под названием video_check:

```
bool video_check = cap.isOpened();
```

При возникновении ошибки в процессе открытия видеофайла программа выведет в консоль соответствующее сообщение, после чего будет ожидать от пользователя нажатия любой клавиши, которое приведет к прекращению ее работы. Данное «аварийное» завершение программы реализовано следующим образом:

```
if (video_check == false)
{cout << "Failed open video file" << endl;
 waitKey(0);
 return -1;}
```

Основная часть программы, осуществляющая манипуляции над входными визуальными данными, находится в бесконечно цикле, выход из которого предусмотрен только в 3 случаях: 1) программа успешно завершила работу с видеофайлом или видеопотоком, обработав все поступившие кадры; 2) программа завершила свою работу в результате какой-либо неисправности, приведшей к прекращению трансляции кадров; 3) программа принудительно завершила свою работу вследствие нажатия пользователем клавиши esc. Проверка первых двух условий выхода из цикла осуществляется при помощи следующей конструкции:

```
bool frame_check = cap.read(frame);
if (frame_check == false)
{break;}
```

Метод read захватывает кадры видеопотока объекта cap, записывает их в матрицу frame и возвращает значение true, если запись кадра была произведена успешно, или false, если трансляция кадров была остановлена,

передавая его затем в контрольную переменную типа bool под названием frame_check. При завершении видеофайла или возникновении неисправности, такой как, например, отключение камеры, программа выйдет из бесконечного цикла посредством команы break.

На этом подготовительный этап завершается: все последующие команды направлены непосредственно на трансформацию и анализ видеоданных. Рассмотрим эту «рабочую» часть кода на примере представленного кадра из видео с видеорегистратора, найденного на просторах сети Интернет (рисунок 30).

Рисунок 30 – Исходный кадр

Первым преобразованием входного изображения является изменение его разрешения. Это нужно для ускорения обработки видеопотока в реальном времени, так как на анализ менее четкого кадра требуется меньше вычислительных ресурсов. Кроме того, для обнаружения хорошо просматриваемых линий дорожной разметки высокая детализация не требуется. Таким образом, для понижения разрешения входного видеопотока с видеорегистратора со стандартных 1920x1080 или 1280x720 пикселей до 640x360 используется функция resize (src, dst, dsize, interpolation), где

– src – матрица, содержащая входное изображение;
– dst – матрица для записи выходного изображения;
– dsize – разрешение выходного изображения;

– interpolation – использующийся метод интерполяции.

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

Size frame_size(640, 360);

resize(frame, resized_frame, frame_size, INTER_NEAREST);

Далее происходит изменение перспективы кадра на вид с высоты птичьего полета. Данная манипуляция значительно упростит дальнейший анализ видеопотока, выделив лишь интересующую нас область дорожного полотна, находящуюся перед автономным транспортным средством и содержащую линии дорожной разметки (рисунок 31). Достичь этого можно при помощи функции warpPerspective (src, dst, M, dsize, flags), где:

– src – матрица, содержащая входное изображение;
– dst – матрица для записи выходного изображения;
– M – матрица трансформации размером 3x3;
– dsize – размер выходного изображения;
– flags – использующийся метод интерполяции.

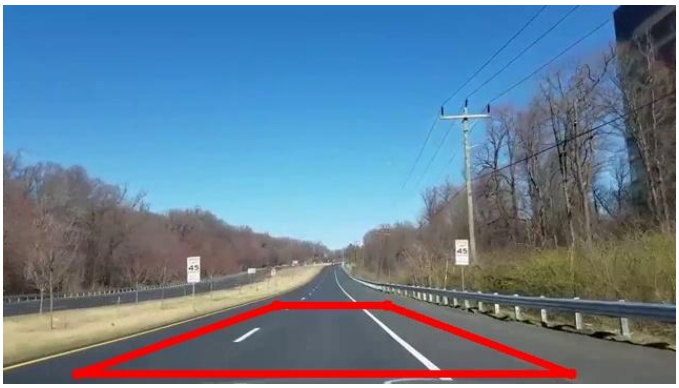

Рисунок 31 – Область интереса

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

matrix = getPerspectiveTransform(pts_1, pts_2);

Size frame_size(360, 180);

warpPerspective(frame, transformed_frame, matrix, frame_size, INTER_NEAREST);

Матрица трансформации matrix получается при помощи функции getPerspectiveTransform (src_points, dst_points), где:

– src_points – массив типа vector, содержащий координаты вершин четырехугольника, определяющего область на входном изображении, перспективу которой необходимо изменить;
– dst_points – массив типа vector, содержащий координаты вершин четырехугольника, определяющего часть выходного изображения, в которую будет помещена преобразованная область;

Результат измерения перспективы кадра на вид с высоты птичьего полета представлен на рисунке 32.

Рисунок 32 – Вид на дорогу с высоты птичьего полета

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

– src – матрица, содержащая входной кадр, который необходимо преобразовать;

– dst – матрица для записи выходного преобразованного кадра;

– code – код требуемого преобразования.

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

cvtColor(transformed_frame, hsv, COLOR_BGR2HSV);

Цветовая модель HSV лучше подходит для обнаружения ярко-белых ограничительных линий разметки за счет независимости параметров, отвечающих за оттенок (Hue), насыщенность (Saturation) и яркость (Value), что делает систему менее чувствительной к изменениям освещенности и позволяет более гибко настраивать параметры при дальнейшей сегментации изображения.

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 [8].

Результат изменения цветового пространства представлен на рисунке 33.

Рисунок 33 – Кадр в цветовом пространстве HSV

Далее переходим к цветовой сегментации кадра с помощью функции inRange(src, lowerb, upperb, dst), где:

– src – матрица, содержащая входное изображение;

– lowerb – нижняя граница цветового диапазона;

– upperb – верхняя граница цветового диапазона;

– dst – матрица для записи выходного изображения;

Данная функция проверяет попадание пикселей входного изображения в цветовой диапазон, заданный при помощи заранее определенных границ. 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. На выходе получается бинарное изображение. Запись, отвечающая за это, в коде выглядит следующим образом:

```
vector<int> lower = { 0,0,170 };
vector<int> upper = { 255,50,255 };
inRange(hsv, lower, upper, bin);
```

Результат цветовой сегментации представлен на рисунке 34.

Рисунок 34 – Бинарное представление кадра

Следующий этап алгоритма представляет собой поиск положения линий разметки на оси абсцисс с помощью построения гистограммы изображения. Для упрощения анализа и экономии вычислительных ресурсов обрабатывается только нижняя половина кадра:

```
rows = bin.rows;
cols = bin.cols;
half = bin(Rect(0, rows / 2, cols, rows / 2));
```

Посредством цикла программа подсчитывает количество всех белых пикселей в каждом столбце, сохраняя результат в массив hist типа vector:

```
vector<int> hist(cols, 0);
for (int i = 0; i < half.rows; ++i) {
for (int j = 0; j < half.cols; ++j) {
if (half.at<uchar>(i, j) == 255) {
hist[j]++;}}}
```

Затем гистограмма делится на правую и левую части, для каждой из которых также при помощи циклов находятся столбцы с максимальным количеством белых пикселей, которые и будут обозначать положение линий дорожной разметки:

```
mid = cols / 2;
maxl = 0;
for (int i = 0; i < mid; ++i) {
if (hist[i] > maxl) {
maxl = hist[i];
left = i;}}
maxr = 0;
for (int i = mid; i < cols; ++i) {
if (hist[i] > maxr) {
maxr = hist[i];
right = i;}}
```

Полученная гистограмма показана на рисунке 35.

Рисунок 35 – Гистограмма кадра

Далее к бинарному изображению применяется метод скользящих окон.

Первым делом, в верхней части кадра относительно полученного ранее положения линий дорожной разметки формируется два окна размером 50x15 пикселей, внутри которых выполняется поиск контуров при помощи функции findContours(image, contours, hierarchy, mode, method), где:

– image – исследуемое изображение;

– contours – массив типа vector, в который будут сохранены все найденные контуры;

– 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.

– mode – режим извлечения найденных контуров (в данном случае это CV_RETR_TREE – способ, при котором программа извлекает все контуры и группирует их в многоуровневую иерархию);

– method – метод аппроксимации контуров (в данном случае это CV_CHAIN_APPROX_SIMPLE – способ, при котором программа склеивает все горизонтальные, вертикальные и диагональные сегменты контуров, оставляя только их конечные точки).

Затем для каждого найденного контура вычисляются моменты, которые представляют собой набор скалярных значений, описывающих геометрические свойства контура. Так, момент нулевого порядка M_{00} характеризует собой площадь контура, а с помощью моментов первого порядка M_{10} и M_{01} можно найти координаты центроида контура, воспользовавшись формулами (1) и (2) [1].

$$x = \frac{M_{10}}{M_{00}}. \#(1)$$

$$y = \frac{M_{01}}{M_{00}}. \#(2)$$

Вычисленные x координаты центроидов контуров участвуют в корректировке положения окон по горизонтальной оси посредством изменения переменных left и right. При этом в массивы left_center и right_center типа vector записываются координаты центров окон, которые являются основными выходными данными данного программного модуля и могут быть использованы, например, в качестве опорных точек для вычисления уравнений прямых, описывающих линии разметки.

После этого стартовое положение окон, характеризующееся переменной start и равнявшееся изначально 180, т.е. высоте кадра, смещается вниз на 15 пикселей. Описанная процедура повторяется 12 раз, пока скользящие окна не доберутся до низа изображения.

Структура, отвечающая за реализацию данного этапа, в коде выглядит следующим образом:

```
mask = bin.clone();
vector<Point> left_center;
vector<Point> right_center;
while (start > 0) {
if (left < 25) left = 25;
win = mask(Rect(left - 25, start - 15, 50, 15));
findContours(win, contours, RETR_TREE, CHAIN_APPROX_SIMPLE);
```

```
for (const auto& contour : contours) {
Moments M = moments(contour);
if (M.m00 != 0) {
cx = static_cast<int>(M.m10 / M.m00);
cy = static_cast<int>(M.m01 / M.m00);
left = left - 25 + cx;}}
if (right > 295) right = 295;
win = mask(Rect(right - 25, start - 15, 50, 15));
contours.clear();
findContours(win, contours, RETR_TREE, CHAIN_APPROX_SIMPLE);
for (const auto& contour : contours) {
Moments M = moments(contour);
if (M.m00 != 0) {
cx = static_cast<int>(M.m10 / M.m00);
cy = static_cast<int>(M.m01 / M.m00);
right = right - 25 + cx;}}
Point2f lc(left, (start + start - 15) / 2);
Point2f rc(right, (start + start - 15) / 2);
left_center.push_back(lc);
right_center.push_back(rc);
start -= 15;}
start = 180;
```

Применение метода скользящих окон представлено на рисунке 36.

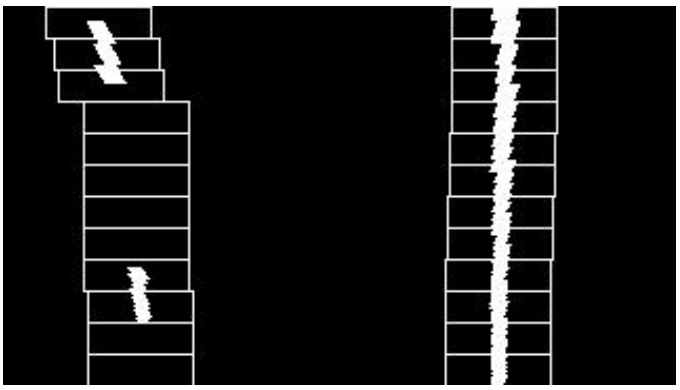

Рисунок 36 – Метод скользящих окон

Центры окон, представляющие собой опорные точки для дальнейшего определения положения беспилотного транспортного средства в пространстве, представлены на рисунке 37.

Рисунок 37 – Опорные точки

Визуализация работы программного модуля по обнаружению и распознаванию линий дорожной разметки представлена на рисунке 38.

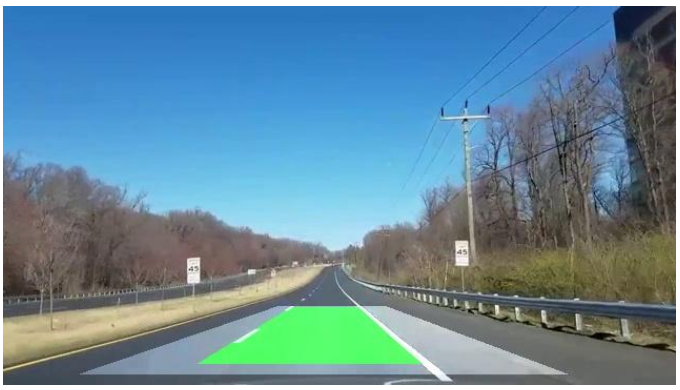

Рисунок 38 – Визуализация результатов работы модуля

3.2.2 Разработка модуля распознавания дорожных знаков

Алгоритм работы программного модуля обнаружения и распознавания дорожных знаков состоит из 9 последовательных этапов:

1. Подключение всех необходимых для работы программы заголовочных файлов и библиотек и объявление переменных.
2. Инициализация камеры.
3. Захват изображения, поступающего с камеры.
4. Проверка успешности захвата кадра.
5. Цветовая сегментация и извлечение бинарного изображения.
6. Поиск контуров объектов на черно-белом кадре и выделение из них наибольшего по площади.
7. Приведение выделенного фрагмента к фиксированному размеру и применение морфологических операций для уменьшения шума.
8. Побитовое сравнение фрагмента с заранее заданным шаблоном.
9. Распознавание дорожного знака.

Блок-схема описанного выше алгоритма работы программы представлена на рисунке 39.

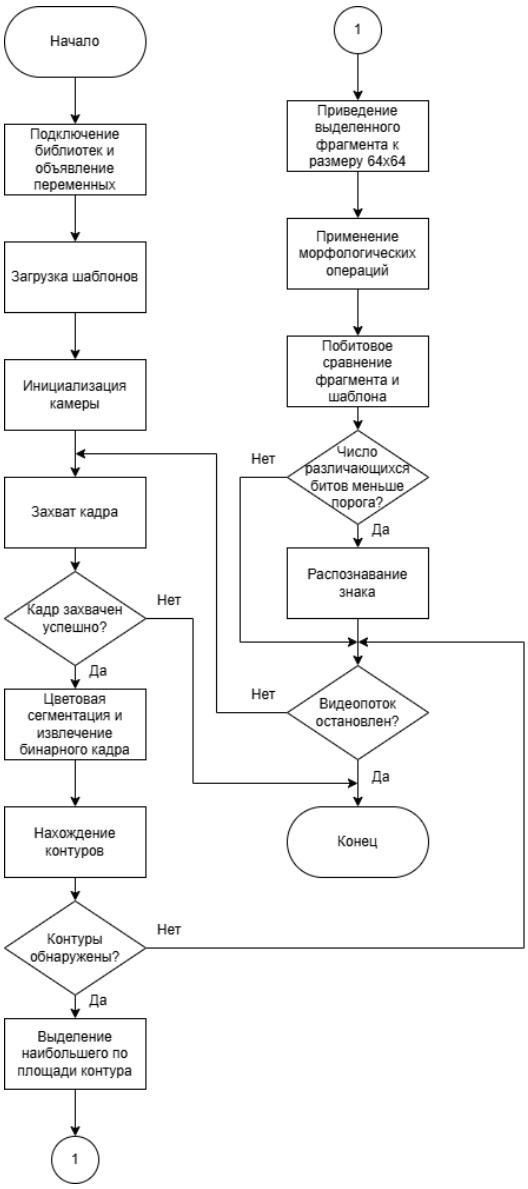

Рисунок 39 – Блок-схема алгоритма работы программного модуля распознавания дорожных знаков

Проведем детальный разбор кода программы.

Как и в модуле распознавания дорожной разметки подготовительный этап для данного алгоритма включает в себя следующие действия: подключение библиотек OpenCV, объявление всех необходимых для работы кода переменных и структур, инициализацию камеры, захват кадров видеопотока в рамках бесконечного цикла, проверку успешности захвата изображения. В качестве дополнения здесь появляется процесс записи в

матрицы заранее подготовленных бинарных изображений-шаблонов размером 64x64 пикселя тех дорожных знаков, что подлежат обнаружению. В качестве примера рассмотрим распознавание знака «Движение прямо» (рисунок 40) на видео с видеорегистратора, взятого из сети Интернет (рисунок 41).

Рисунок 40 – Дорожный знак «Движение прямо»

Рисунок 41 – Исходный кадр

«Рабочая» часть алгоритма начинается с цветовой сегментации кадра при помощи функции inRange(), которая выделяет все темно-синие области в поле зрения камеры, формируя бинарное изображение.

Далее на получившемся черно-белом кадре находятся контуры объектов посредством команды findContours(), которые затем ранжируются в порядке увеличения площади:

sort(contours.begin(), contours.end(), [](const vector<Point>& a, const vector<Point>& b){return contourArea(a) > contourArea(b);});

Самый большой из них помещается в bounding box, после чего содержимое этого прямоугольника записывается в специальную матрицу holder и приводится к фиксированному размеру 64x64 пикселя:

rect = boundingRect(contours[0]);

holder = frame(rect);

resize(holder, holder, Size(64, 64));

Предполагается, что данный выделенный фрагмент (рисунок 42) и содержит искомый знак. К нему вновь применяется цветовая сегментация, а также морфологическая операция «Закрытие», состоящая из дилатации и эрозии и призванная уменьшить шум:

inRange(holder, Scalar(70, 40, 20), Scalar(120, 100, 80), holder);

dilate(holder, holder, Mat(), Point(-1, -1), 1);

erode(holder, holder, Mat(), Point(-1, -1), 1);

После этого обработанный фрагмент (рисунок 43) сравнивается с загруженным ранее шаблоном знака (рисунок 44) при помощи функции norm(src1, src2, normType), где

- src1 – матрица, содержащая выделенный фрагмент;
- src2 – матрица, содержащая шаблон знака;
- normType – тип нормы, математическая метрика, которая будет использоваться для сравнения изображений;

Рисунок 42 – Фрагмент кадра до уменьшения шума

Рисунок 43 – Фрагмент кадра после уменьшения шума

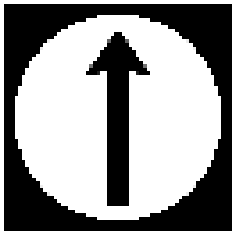

Рисунок 44 – Шаблон знака

В данном случае был применен аргумент NORM_HAMMING, который вычисляет расстояние Хэмминга между двумя матрицами, т.е. общее количество различающихся битов во всех пикселях. Данное число помещается в переменную difference и показывает, насколько сильно различаются выделенный из кадра фрагмент с дорожным знаком и шаблон. Запись, отвечающая за это, в коде выглядит следующим образом:

int difference = norm(arrow, holder, NORM_HAMMING);

Далее difference сравнивается с заранее определенным порогом. Если количество различающихся пикселей меньше данной отметки, распознавание знака считается успешным и участок кадра с ним выделяется прямоугольной рамкой с подписью. Визуализация результатов работы программного модуля по обнаружению и распознаванию дорожных знаков представлена на рисунке 45.

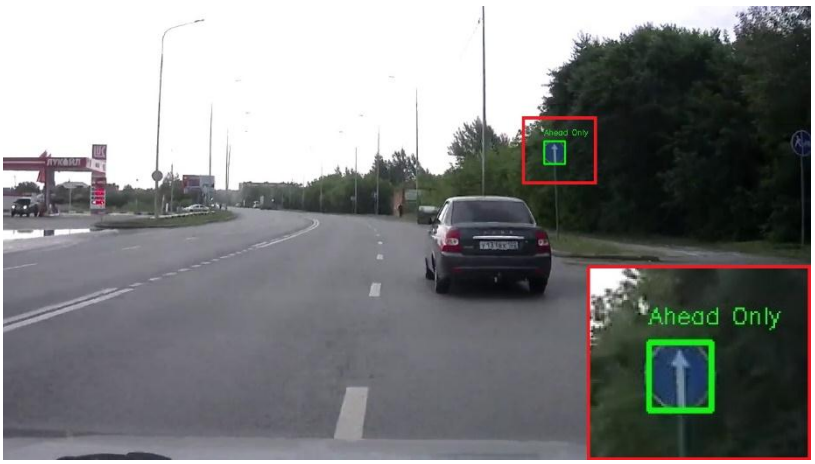

Рисунок 45 – Визуализация результатов работы модуля

3.2.3 Разработка модуля обнаружения пешеходов

Алгоритм работы программного модуля обнаружения пешеходов состоит из 8 последовательных этапов:

1. Подключение всех необходимых для работы программы заголовочных файлов и библиотек и объявление переменных.
2. Инициализация камеры.
3. Захват изображения, поступающего с камеры.
4. Проверка успешности захвата кадра.
5. Инициализация HOG-дескриптора для обнаружения пешеходов.
6. Инициализация трекера для отслеживания пешеходов.
7. Переобнаружение пешеходов и перезапуск трекера каждые 15 кадров.
8. Визуализация распознавания и отслеживания пешеходов.

Блок-схема описанного выше алгоритма работы программы представлена на рисунке 46.

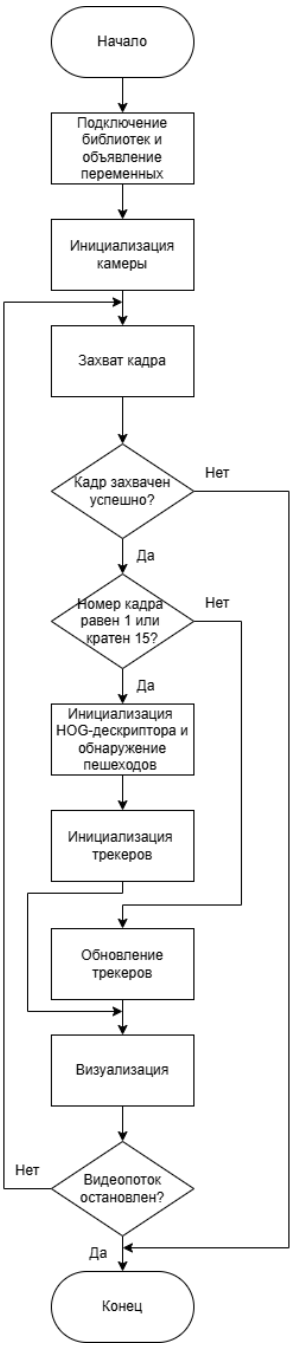

Рисунок 46 – Блок-схема алгоритма работы программного модуля обнаружения пешеходов

Проведем детальный разбор кода программы.

Как и в предыдущих модулях подготовительный этап для данного алгоритма включает в себя следующие действия: подключение библиотек OpenCV, объявление всех необходимых для работы кода переменных и

структур, инициализацию камеры, захват кадров видеопотока, проверку успешности захвата изображения.

Далее происходит инициализация HOG-дескриптора со стандартным предобученным SVM-классификатором для обнаружения людей:

hog.setSVMDetector(HOGDescriptor::getDefaultPeopleDetector());

Гистограмма направленных градиентов (Histogram of Oriented Gradients, HOG) представляет собой метод выделения признаков на изображении, основанный на анализе направлений изменения яркости пикселей. Кратко рассмотрим принцип работы данного подхода. Первым делом изображение преобразуется в оттенки серого и разбивается на небольшие ячейки (обычно 8×8 пикселей), для каждой из которых рассчитываются градиенты – направления и величины изменения яркости каждого пикселя. После этого формируется гистограмма, отражающая, сколько градиентов ориентировано в каждом из заранее определенных направлений (например, в 9 угловых диапазонах от 0° до 180°). Для повышения устойчивости к изменениям освещения и контрастности, несколько ячеек объединяются в блоки и их гистограммы нормализуются. Итоговый дескриптор изображения формируется путём объединения всех нормализованных гистограмм. Полученный вектор признаков может быть использован для классификации объектов, таких как люди, с помощью обученного алгоритма, например SVM (Support Vector Machine).

Следом инициируется поиск пешеходов на кадрах видеопотока:

hog.detectMultiScale(frame, detections, 0, Size(4, 4), Size(16, 16), 1.05, 2);

После этого происходит инициализация MultiTracker. Этот компонент библиотеки OpenCV предназначен для одновременного отслеживания нескольких объектов в видеопотоке. После получения начальных координат прямоугольников пешеходов на первом кадре, MultiTracker автоматически обновляет положение каждого из них на последующих кадрах, используя выбранный алгоритм трекинга. Запись, отвечающая за это, в коде выглядит следующим образом:

```
multiTracker = makePtr<legacy::MultiTracker>();
for (const auto& detection : detections)
{multiTracker->add(legacy::TrackerKCF::create(), frame, detection);}
```

Далее каждые 15 кадров происходит переобнаружение объектов и обновление трекеров для распознавания новых пешеходов:

```
int frameNumber = 1;
while (cap.read(frame))
{frameNumber++;
if (frameNumber % 15 == 0)
{detections.clear();
hog.detectMultiScale(frame, detections, 0, Size(4, 4), Size(16, 16), 1.05, 2);
multiTracker = makePtr<legacy::MultiTracker>();
for (const auto& detection : detections)
{multiTracker->add(legacy::TrackerKCF::create(), frame, detection);}}
else{multiTracker->update(frame);}
```

Визуализация результатов работы программного модуля по обнаружению движения пешеходов для предотвращения столкновений представлена на рисунке 47.

Рисунок 47 – Визуализация результатов работы модуля

Выводы по главе 3

В главе 3 магистерской диссертации был сделан и обоснован выбор в пользу классических методов компьютерного зрения для реализации программных алгоритмов обнаружения и распознавания объектов окружающей среды в беспилотных транспортных средствах; были выбраны и описаны инструменты разработки программных алгоритмов: язык программирования высокого уровня C++, библиотека компьютерного зрения с открытым исходным кодом OpenCV, интегрированная среда разработки Microsoft Visual Studio; были разработаны, подробно описаны и проиллюстрированы с помощью рисунков и блок-схем программные алгоритмы обнаружения и распознавания дорожной разметки, обнаружения и распознавания дорожных знаков, обнаружения движения пешеходов для предотвращения столкновений.

Глава 4 Апробация разработанных программных алгоритмов

4.1 Условия тестирования

Для тестирования работоспособности разработанных программных алгоритмов были выбраны 8 видеозаписей с видеорегистраторов автомобилей, взятые из открытых источников на просторах сети Интернет.

В качестве метрик для определения эффективности работы программных алгоритмов были выбраны следующие показатели:

– точность – то, насколько правильно программа обнаружила и распознала полосы разметки, пешеходов и дорожные знаки, были ли ошибки или ложные срабатывания; определяется визуально и представляет собой экспертную оценку;

– быстродействие – количество кадров, обрабатываемое программой в секунду.

Тестирование было произведено на ПК со следующими характеристиками:

Таблица 2 – Характеристики ПК для тестирования

Параметры конфигурации	Характеристика
Процессор	Intel Core i5-10300H (2.5 ГГц)
ОЗУ	16 Гб
Видеокарта	GeForce GTX 1050 Ti
Жесткий диск	500 Гб

4.2 Апробация модуля распознавания дорожной разметки

Результаты тестирования модуля распознавания дорожной разметки представлены на рисунках 48, 49 и 50.

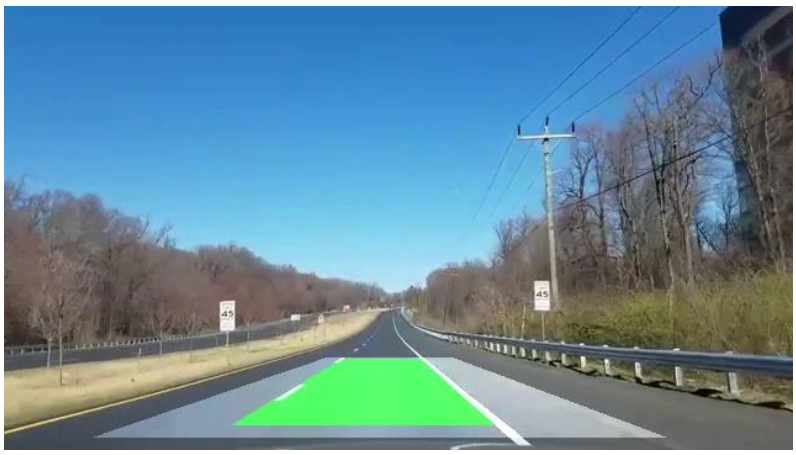

Рисунок 48 – Визуализация результатов работы модуля на видеозаписи №1

Рисунок 49 – Визуализация результатов работы модуля на видеозаписи №2

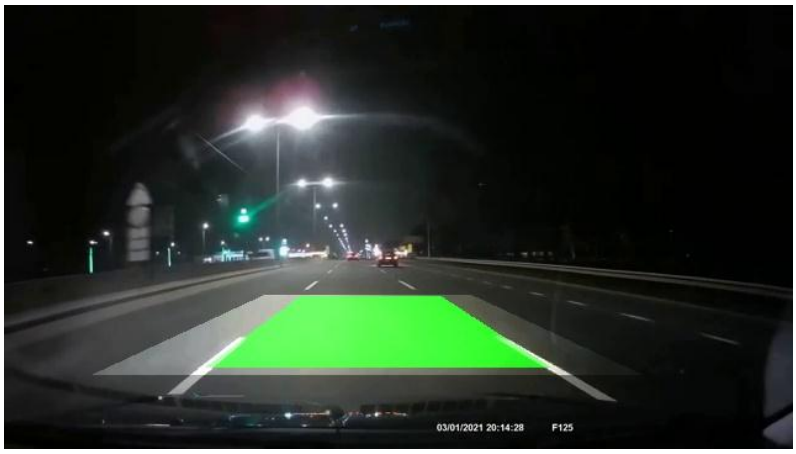

Рисунок 50 – Визуализация результатов работы модуля на видеозаписи №3

Во всех трех случаях программа продемонстрировала хорошую точность, корректно распознавая разметку и не допуская ошибок и неточностей.

Быстродействие программного модуля варьируется от 25 кадров в секунду при наличии полной визуализации всех этапов обработки до 100 кадров в секунду в случае, когда функция визуализации полностью отключена, что является отличным показателем для системы, которая должна функционировать в условиях реального времени. На рисунке 51 продемонстрировано сравнение FPS разработанного алгоритма и существующих нейросетевых решений из следующих научных работ:

– Detection-segmentation convolutional neural network for autonomous vehicle perception [17];
– Road Surface Defect Detection Algorithm Based on YOLOv8 [28];
– Inter-Region Affinity Distillation for Road Marking Segmentation [22];

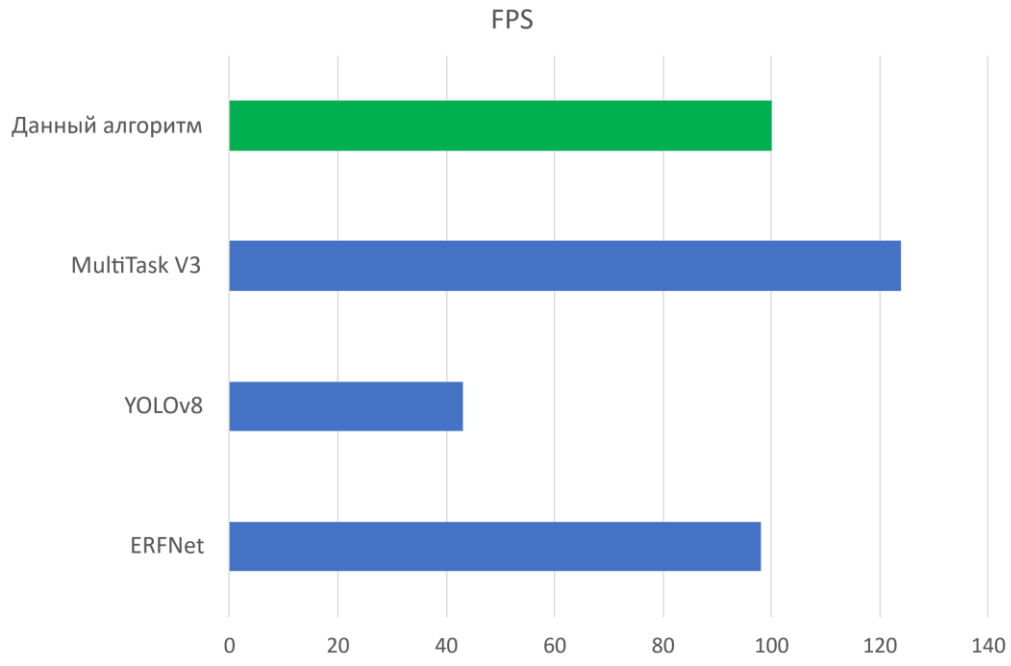

Рисунок 51 – Сравнение FPS

Из рисунка видно, что разработанный в рамках данной выпускной квалификационной работы программный модуль распознавания дорожной разметки на основе классических методов компьютерного зрения в вопросе быстродействия не уступает схожим нейросетевым алгоритмам.

Также стоит отметить гибкость алгоритма, так как он способен хорошо работать в условиях меняющегося освещения: в дневное время, в пасмурную погоду и ночью. Выявленным недостатком данного программного решения является необходимость корректировки области интереса для детектора разметки в соответствии с расположением камеры в транспортном средстве.

4.3 Апробация модуля распознавания дорожных знаков

Результаты тестирования модуля распознавания дорожных знаков представлены на рисунках 52, 53 и 54.

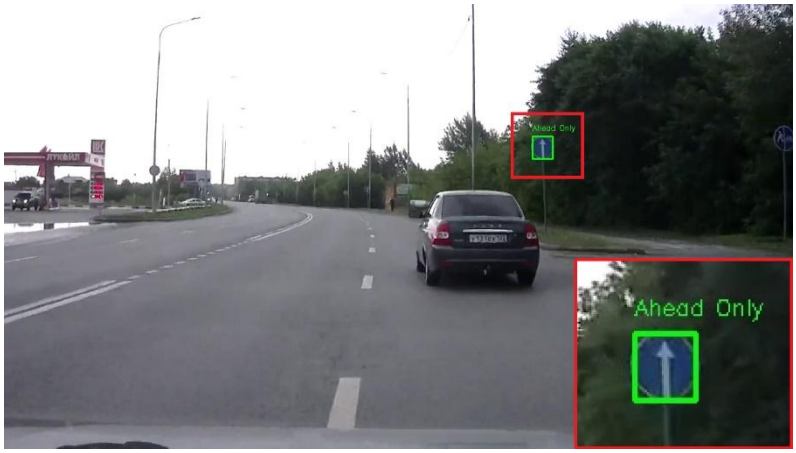

Рисунок 52 – Визуализация результатов работы модуля на видеозаписи №1

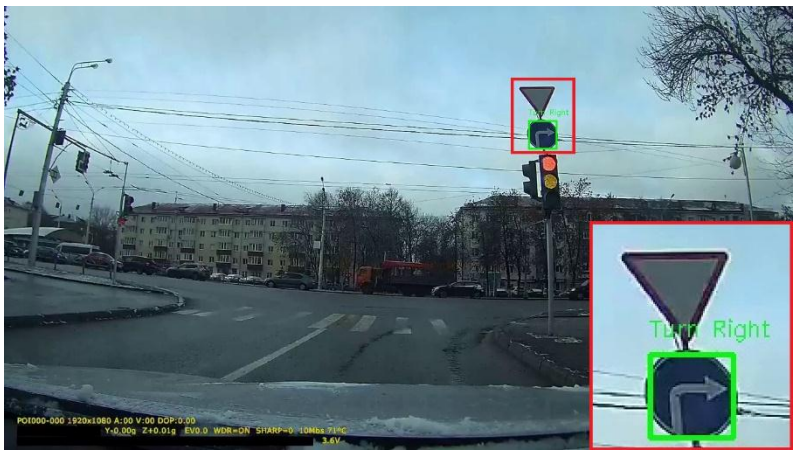

Рисунок 53 – Визуализация результатов работы модуля на видеозаписи №2

Рисунок 54 – Визуализация результатов работы модуля на видеозаписи №3

Во всех трех случаях программа продемонстрировала хорошую точность, корректно распознавая дорожные знаки и не допуская ошибок и неточностей.

Быстродействие программного модуля составляет в среднем 35 кадров в секунду, что вполне достаточно для функционирования системы в условиях реального времени. На рисунке 55 продемонстрировано сравнение FPS разработанного алгоритма и существующих нейросетевых решений из следующих научных работ:

– Traffic Sign Detection Based on the Improved YOLOv5 [29];

– A YOLOv8-CE-based real-time traffic sign detection and identification method for autonomous vehicles [25];
– A Robust TrafficSignNet Algorithm for Enhanced Traffic Sign Recognition in Autonomous Vehicles Under Varying Light Conditions [24];

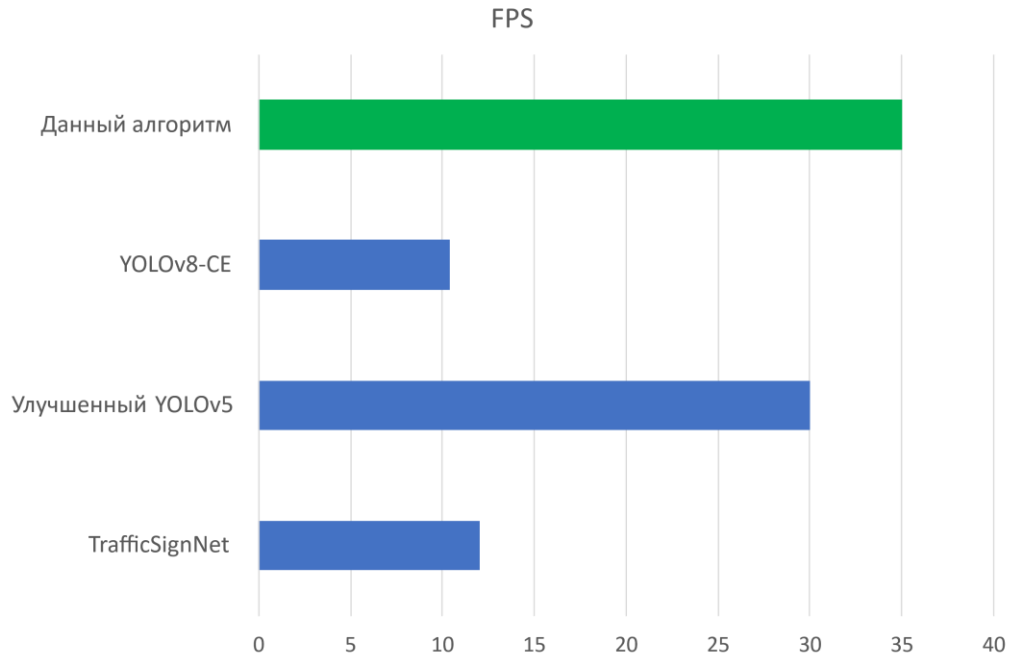

Рисунок 55 – Сравнение FPS

Из рисунка видно, что разработанный в рамках данной выпускной квалификационной работы программный модуль распознавания дорожных знаков на основе классических методов компьютерного зрения в вопросе быстродействия не только не уступает, но и превосходит схожие нейросетевые алгоритмы.

Также стоит отметить гибкость алгоритма, так как он способен распознавать различные знаки, имея соответствующие шаблоны. Выявленным недостатком данного программного решения является то, что выделение контура по наибольшей отсекаемой им площади не всегда

является корректным критерием определения области кадра, содержащей искомый дорожный знак.

4.4 Апробация модуля обнаружения пешеходов

Результаты тестирования модуля обнаружения пешеходов представлены на рисунках 56 и 57.

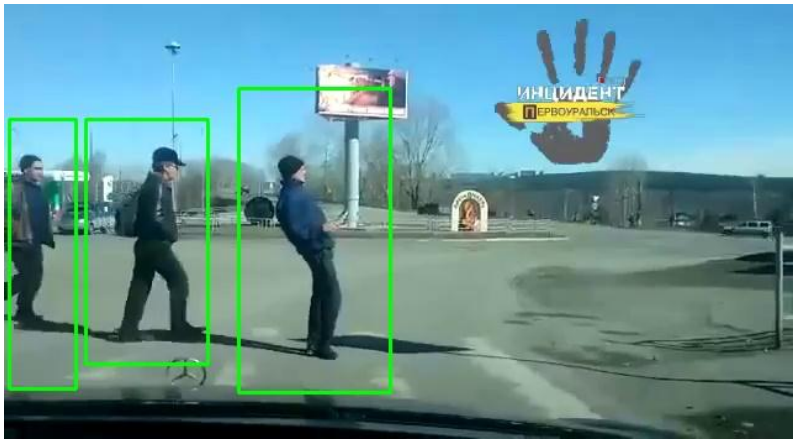

Рисунок 56 – Визуализация результатов работы модуля на видеозаписи №1

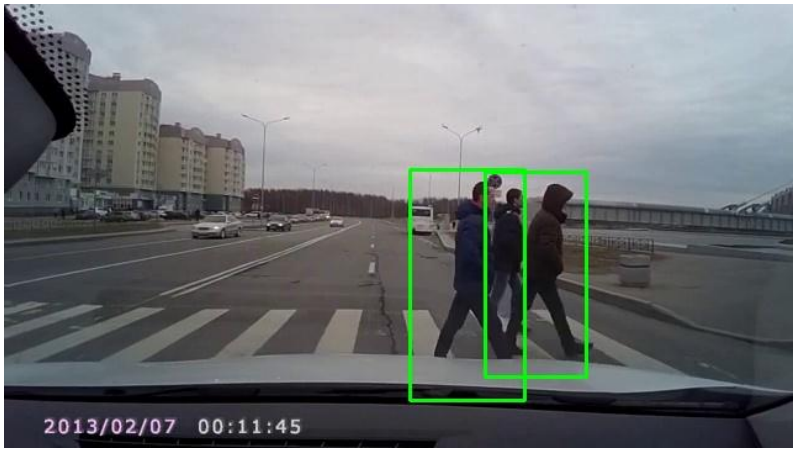

Рисунок 57 – Визуализация результатов работы модуля на видеозаписи №2

По результатам тестирования программа показала приемлемую точность, корректно распознавая всех пешеходов, однако иногда допуская такую ошибку, как наложение нескольких ограничивающих прямоугольников на одного пешехода.

Быстродействие программного модуля составляет примерно 20 кадров в секунду при одновременном отслеживании трех человек, что немного меньше требуемого уровня в 30 кадров в секунду для систем реального времени. При этом скорость работы модуля значительно снижается с увеличением числа отслеживаемых пешеходов. На рисунке 58 продемонстрировано сравнение FPS разработанного алгоритма и существующих нейросетевых решений из следующих научных работ:

– A Parallel Convolutional Neural Network for Pedestrian Detection [30];
– YOLOv2PD: An Efficient Pedestrian Detection Algorithm Using Improved YOLOv2 Model [26];
– Real-Time Pedestrian Detection With Deep Network Cascades [16];

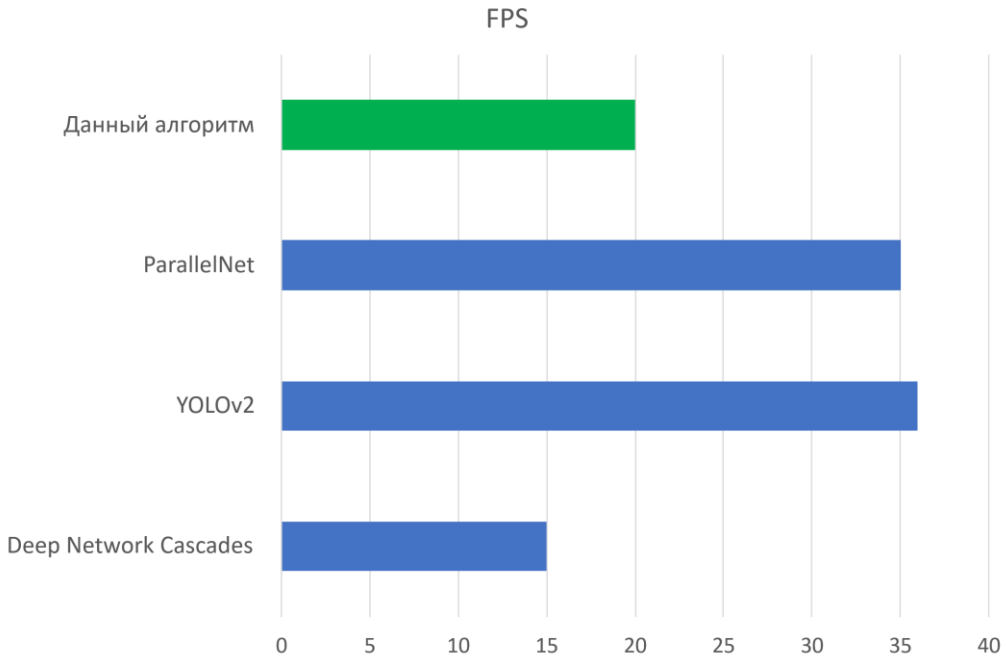

Рисунок 58 – Сравнение FPS

Из рисунка видно, что разработанный в рамках данной выпускной квалификационной работы программный модуль распознавания пешеходов на основе классических методов компьютерного зрения в вопросе быстродействия лишь немного уступает существующим нейросетевым алгоритмам.

Также стоит отметить гибкость алгоритма, так как он способен хорошо работать в условиях меняющегося освещения: в дневное время и в пасмурную погоду. Выявленным недостатком данного программного решения является периодически возникающая ошибка, при которой программа выделяет пешехода не одним, а сразу двумя ограничивающими прямоугольниками.

Выводы по главе 4

В главе 4 магистерской диссертации была проведена апробация разработанных алгоритмов обнаружения объектов окружающей среды; полученные результаты соответствуют ожиданиям и демонстрируют способность программных модулей достаточно эффективно распознавать дорожную разметку, дорожные знаки и пешеходов на кадрах видеозаписей с видеорегистраторов, а также конкурировать в быстродействии со схожими нейросетевыми решениями, что подтверждает гипотезу исследования.

Заключение

Представленная магистерская диссертация посвящена решению актуальной задачи обнаружения и распознавания объектов окружающей среды в беспилотных транспортных средствах с помощью классических методов компьютерного зрения.

В главе 1 диссертации была подтверждена актуальность задачи обнаружения и распознавания объектов окружающей среды в беспилотных транспортных средствах.

В главе 2 диссертации были рассмотрены наиболее популярные и эффективные методы компьютерного зрения в контексте решения задачи распознавания объектов на изображении: традиционные методы и методы, основанные на глубоком обучении. Были выявлены преимущества и недостатки обоих подходов, проведено их сравнение.

В главе 3 диссертации была обоснована целесообразность выбора классических методов компьютерного зрения для реализации программных алгоритмов обнаружения и распознавания объектов окружающей среды в беспилотных транспортных средствах. Были выбраны и описаны инструменты разработки: язык программирования, библиотека компьютерного зрения, среда программирования, а также разработаны и подробно описаны программные алгоритмы для обнаружения и распознавания дорожной разметки, дорожных знаков, пешеходов.

В главе 4 диссертации была проведена апробация разработанных алгоритмов обнаружения объектов окружающей среды на видеозаписях с видеорегистраторов автомобилей, взятых из открытых источников сети Интернет; было проведено сравнение разработанных алгоритмов с уже существующими решениями, основанными на технологиях глубокого обучения, по показателю FPS (Frames Per Second). Полученные результаты соответствуют ожиданиям и демонстрируют способность программных модулей достаточно эффективно распознавать дорожную разметку, дорожные

знаки и пешеходов на кадрах видеозаписей с бортовых камер транспортных средств, а также конкурировать в быстродействии со схожими нейросетевыми решениями, что подтверждает гипотезу исследования.

Для подтверждения гипотезы исследования был использованы данные, полученные в результате изучения научной отечественной и зарубежной литературы в области компьютерного зрения, а также моделирования работы системы технического зрения для решения конкретных задач.

Таким образом, в ходе выполнения данной выпускной квалификационной работы была решена актуальная научно-практическая задача и разработаны алгоритмы, которые способны достаточно точно распознавать дорожную разметку, дорожные знаки и пешеходов, при этом не уступая в быстродействии нейросетевым программным решениям.

Результаты, полученные в ходе данного исследования, могут быть полезны для исследователей и разработчиков, работающих в области компьютерного зрения.

Список используемой литературы и используемых источников

1. 33. OpenCV шаг за шагом. Сравнение контуров через суммарные характеристики – моменты [Электронный ресурс]. URL: https://robocraft.ru/computervision/867 (дата обращения: 23.02.2025).

2. 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.

3. В цехах АвтоВАЗа появились поющие тележки-роботы "Антонина", их сняли на видео [Электронный ресурс]. URL: https://rg.ru/2025/04/21/v-cehah-avtovaza-poiavilis-poiushchie-telezhki-roboty-antonina-ih-sniali-na-video.html?ysclid=m9zvbgq4qm926602934 (дата обращения: 26.04.2025).

4. Горячкин Б. С., Китов М. А. КОМПЬЮТЕРНОЕ ЗРЕНИЕ // E-Scio. 2020. №9 (48). URL: https://cyberleninka.ru/article/n/kompyuternoe-zrenie-1 (дата обращения: 22.01.2025).

5. Как Яндекс запускает роботов-доставщиков в новых районах и городах [Электронный ресурс]. URL: https://habr.com/ru/companies/yandex/articles/887684/ (дата обращения: 26.04.2025).

6. Клетте Рейнхард. Компьютерное зрение. Теория и алгоритмы. М. : ДМК Пресс, 2019. 506 с.

7. Компьютерное зрение в 2024 году: Главные задачи и направления [Электронный ресурс]. URL: https://habr.com/ru/companies/otus/articles/810207/ (дата обращения: 26.04.2025).

8. 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.

9. Микряков, П. С. Система технического зрения для мобильного робота: специальность 11.03.04 «Электроника и наноэлектроника»: бакалаврская работа / Микряков Павел Сергеевич; Тольяттинский государственный университет. – Тольятти, 2023. – 93 с.

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

11. Селянкин В.В. Решение задач компьютерного зрения: Учебное пособие. Таганрог : Изд-во ЮФУ, 2016. 92 с.

12. Селянкин В.В., Скороход С.В. Анализ и обработка изображений в задачах компьютерного зрения: учебное пособие. Таганрог : Изд-во ЮФУ, 2015. 82 с.

13. Скрытая угроза: только за два последних года Amazon трудоустроила 400 тыс. роботов на своих объектах [Электронный ресурс]. URL: https://www.ixbt.com/live/science/skrytaya-ugroza-tolko-za-dva-poslednih-goda-amazon-trudoustroila-400-tys-robotov-na-svoih-obektah.html (дата обращения: 26.04.2025).

14. 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.

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

16. Angelova A. Real-Time Pedestrian Detection with Deep Network Cascades / Angelova A., Krizhevsky A., Vanhoucke V., Ogale A.S., Ferguson D. // British Machine Vision Conference, 2015.

17. Baczmanski M. Detection-segmentation convolutional neural network for autonomous vehicle perception / Baczmanski M., Synoczek R., Wasala M., Kryjak T. // TechRxiv, July 10, 2023.

18. Computer Vision Algorithms [Электронный ресурс]. URL: https://www.geeksforgeeks.org/computer-vision-algorithms/ (дата обращения: 22.04.2025).

19. Computer Vision Tasks [Электронный ресурс]. URL: https://www.geeksforgeeks.org/computer-vision-tasks/ (дата обращения: 22.04.2025).

20. 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.

21. Difference between Traditional Computer Vision Techniques and Deep Learning-based Approaches [Электронный ресурс]. URL: https://www.geeksforgeeks.org/difference-between-traditional-computer-vision-techniques-and-deep-learning-based-approaches/ (дата обращения: 21.04.2025).

22. Hou Y. Inter-Region Affinity Distillation for Road Marking Segmentation / Hou Y., Ma Z., Liu C., Hui T.-W., Loy C.C. // IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020.

23. How Amazon launched the warehouse robotics industry [Электронный ресурс]. URL: https://www.techtarget.com/searchaws/feature/How-Amazon-launched-the-warehouse-robotics-industry (дата обращения: 26.04.2025).

24. Kandasamy K. A Robust TrafficSignNet Algorithm for Enhanced Traffic Sign Recognition in Autonomous Vehicles Under Varying Light Conditions / Kandasamy K., Natarajan Y., Ramasamy S. // Neural Processing Letters. – 2024. – Vol. 56. – P. 1–22.

25. Luo Y. A YOLOv8-CE-based real-time traffic sign detection and identification method for autonomous vehicles / Luo Y., Ci Y., Zhang H., Wu L. // Digital Transportation and Safety. – 2024. – Vol. 3, No. 3. – P. 82–91.

26. Murthy C.B. YOLOv2PD: An Efficient Pedestrian Detection Algorithm Using Improved YOLOv2 Model / Murthy C.B., Hashmi M.F., Muhammad G.,

Alqahtani S. // Computers, Materials & Continua. – 2021. – Vol. 69, No. 3. – P. 3015–3031.

27. Open Source Computer Vision Library [Официальный сайт разработчика OpenCV]. URL: https://opencv.org/ (дата обращения: 17.03.2025).

28. Sun Z. Road Surface Defect Detection Algorithm Based on YOLOv8 / Sun Z., Zhu L., Qin S., Yu Y., Ju R., Li Q. // Electronics. – 2024. – Vol. 13, No. 12. – Article 2413.

29. Zhang R. Traffic Sign Detection Based on the Improved YOLOv5 / Zhang R., Zheng K., Shi P., Mei Y., Li H., Qiu T. // Applied Sciences. – 2023. – Vol. 13, No. 17. – Article 9748.

30. Zhu M. A Parallel Convolutional Neural Network for Pedestrian Detection / Zhu M., Wu Y. // Electronics. – 2020. – Vol. 9, No. 9. – Article 1478.