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

ИНСТИТУТ МАШИНОСТРОЕНИЯ

(наименование института полностью)

Кафедра «Промышленная электроника»

(наименование)

11.04.04 «Электроника и наноэлектроника»

(код и наименование направления подготовки)

Электронные приборы и устройства

(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Распознание фигуры человека в офисных помещениях средствами компьютерного зрения»

Студент Н.А. Краснопевцева

(И.О. Фамилия) *(личная подпись)*

Научный руководитель к.т.н., А.А. Шевцов

(ученая степень, звание, И.О. Фамилия)

Тольятти 2022

Содержание

Введение

Одна из самых перспективных наук о компьютерах и программах – компьютерное зрение. Это важнейшая область в искусственном интеллекте, включающая сразу несколько этапов: распознавание содержимого изображения, определение объекта и его классифицирование [5].

Поиск и обнаружение объектов на фотографии или видео со временем становится лишь актуальнее. Например, уже почти каждый смартфон способен распознавать лицо владельца. Уже на государственном уровне используется биометрия, например, при получении гражданами банковской карты.

В то же время, растет необходимость использования систем компьютерного зрения на производствах. По-прежнему случаются несчастные случаи из-за неосторожности и несоблюдения техники безопасности. И если у распознавания лица есть сторонники и противники, то повышение уровня безопасности с помощью контроля за соблюдением техники безопасности может вызвать лишь положительную реакцию.

Система компьютерного зрения, как и любая система компьютерного видеонаблюдения, состоит из камер, сервера записи данных, сервера обработки данных и рабочей станции оператора.

На данный момент подобные системы видеонаблюдения применяются в таких сферах как:

— промышленные процессы,

— энергетические системы,

— мониторинг несанкционированных вторжений,

— визуальное сопровождение сотрудников.

На производстве существует множество факторов, воздействие которых при определенных условиях может вызвать профессиональные заболевания сотрудников и снижение работоспособности.

Согласно ГОСТ 12.0.003-2015 вредные и опасные факторы классифицируются на следующие группы: физические (движущиеся машины и механизмы, электрический ток, повышенный уровень электромагнитных и ионизирующих излучений, шума, вибрации и т.д.), химические (токсичные, мутагенные, канцерогенные и т.д.), психофизиологические и социальные (перенапряжение анализаторов, физические нагрузки, нервно-психические перегрузки и т.д.) [3].

При наличии какого-либо из факторов на рабочем месте, вместо работника во вредную среду можно поместить видеокамеру, за которой живой человек может наблюдать в режиме реального времени, что позволит ему в любой момент при необходимости вмешаться в процесс.

С помощью видеонаблюдения технические специалисты могут контролировать работоспособность энергетических систем. В некоторых отраслях температура и уровень электроэнергии должны поддерживаться постоянными.

Мониторинговая компания может поручить своим специалистам следить за системами отопления, вентиляции воздуха и электрическими системами, чтобы убедиться в отсутствии перебоев в работе. Если с оборудованием когда-либо возникнет проблема, обученные технические специалисты в кратчайшее время смогут направить на это место ремонтный персонал.

Использование систем видеонаблюдения в режиме реального времени с целью отслеживания несанкционированного вторжения позволяет экономить не только человеко-часы, но и оборудование и энергию. Также данная система позволит избавиться от необходимости круглосуточного дежурства рядом с охраняемым объектом. Технические специалисты могут удаленно отслеживать несанкционированное открытие дверей и сразу отправлять соответствующий сигнал пользователю.

Визуальное сопровождение сотрудников — чрезвычайно популярная услуга, которую предоставляют компании, занимающиеся мониторингом

систем видеонаблюдения. Всякий раз, когда сотрудник заканчивает рабочий день и собирается идти к своей машине, он может вызвать специалиста по мониторингу, который заранее осмотрит территорию и сообщит, безопасно ли идти. Сотрудник также может попросить специалиста наблюдать за ним, пока он идет к своей машине [17].

Задача данной магистерской диссертации заключается в том, чтобы отойти от использования мощных стационарных платформ, и перевести функционал подобных систем на мобильные платформы, которые используются в так называемой embedded-электронике, то есть в электронике, внедряемой в те или иные изделия, например, из категории «умный дом».

1 Общие сведения о встраиваемых системах

Встроенные системы всегда функционируют как часть готового устройства — именно это подразумевается под термином «встроенные». Это недорогие, маломощные небольшие компьютеры, встроенные в другие механические или электрические системы. Как правило, они состоят из процессора, блока питания, памяти и коммуникационных портов. Данные системы используют коммуникационные порты для передачи данных между процессором и периферийными устройствами — часто другими встроенными системами — с использованием коммуникационного протокола. Процессор интерпретирует эти данные с помощью минимального программного обеспечения, хранящегося в памяти. Программное обеспечение обычно очень специфично для функции, которую выполняет встроенная система.

На сегодняшний день встраиваемые системы используются в самых разных технологиях в следующих отраслях:

— автомобили,

— мобильные телефоны,

— промышленные машины,

— медицинское оборудование.

Современные автомобили обычно состоят из множества компьютеров или встроенных систем, предназначенных для выполнения различных задач внутри автомобиля. Некоторые из этих систем выполняют основные служебные функции, а другие предоставляют развлекательные или ориентированные на пользователя функции. Некоторые встроенные системы в автомобилях включают круиз-контроль, датчики резервного копирования, управление подвеской, навигационные системы и системы подушек безопасности и т.д.

Мобильные телефоны состоят из множества встроенных систем, включая программное и аппаратное обеспечение с графическим интерфейсом

пользователя, операционной системы, камеры, микрофона и модуля ввода-вывода.

Они могут содержать такие встроенные системы как датчики, а также сами могут быть встроенными системами. Промышленные машины часто имеют встроенные системы автоматизации, которые выполняют определенные функции контроля и управления.

Медицинское оборудование может содержать такие встроенные системы как датчики и механизмы управления. Подобное оборудование должно быть очень удобным в использовании, чтобы здоровье человека не подвергалось опасности из-за предотвратимых ошибок машин. Это означает, что они часто включают более сложную операционную систему и графический интерфейс.

1.1 Характеристики встраиваемых систем

Основная характеристика встроенных систем заключается в том, что они ориентированы на конкретные задачи.

Кроме того, встроенные системы могут включать следующие характеристики:

— обычно состоят из аппаратного, программного и микропрограммного обеспечения;

— могут быть встроены в более крупную систему для выполнения определенной функции, поскольку они созданы для специализированных задач внутри системы, а не для различных задач;

— могут быть либо микропроцессорными, либо микроконтроллерными. И те, и другие представляют собой интегральные схемы, обеспечивающие вычислительную мощность системы;

— часто используются для обнаружения и вычислений в реальном времени в устройствах Интернета вещей (IoT), которые представляют

собой устройства, подключенные к Интернету и не требующие пользователя для работы;

— могут различаться по сложности и функциям, что влияет на тип используемого программного обеспечения, встроенного программного обеспечения и аппаратного обеспечения;

— часто требуется выполнять свои функции в условиях ограниченного времени, чтобы поддерживать нормальное функционирование более крупной системы [14].

1.2 Структура встраиваемых систем

Встроенные системы различаются по сложности, но, как правило, состоят из трех основных элементов:

— аппаратное обеспечение,

— программное обеспечение и прошивки,

— операционная система реального времени (ОСРВ).

Аппаратное обеспечение встроенных систем основано на микропроцессорах и микроконтроллерах. Микропроцессоры очень похожи на микроконтроллеры и, как правило, относятся к ЦП (центральному процессору), который интегрирован с другими основными вычислительными компонентами, такими как микросхемы памяти и процессоры цифровых сигналов (DSP). В микроконтроллерах эти компоненты встроены в один чип.

Программное обеспечение для встраиваемых систем может различаться по сложности. Однако микроконтроллеры промышленного класса и встроенные системы IoT обычно работают с очень простым программным обеспечением, требующим небольшого объема памяти.

Встроенные системы не всегда включены во встроенные системы, особенно в системы меньшего масштаба. ОСРВ определяют, как работает система, контролируя программное обеспечение и устанавливая правила во время выполнения программы.

Общая схема структуры подобных систем представлена на рисунке 1.

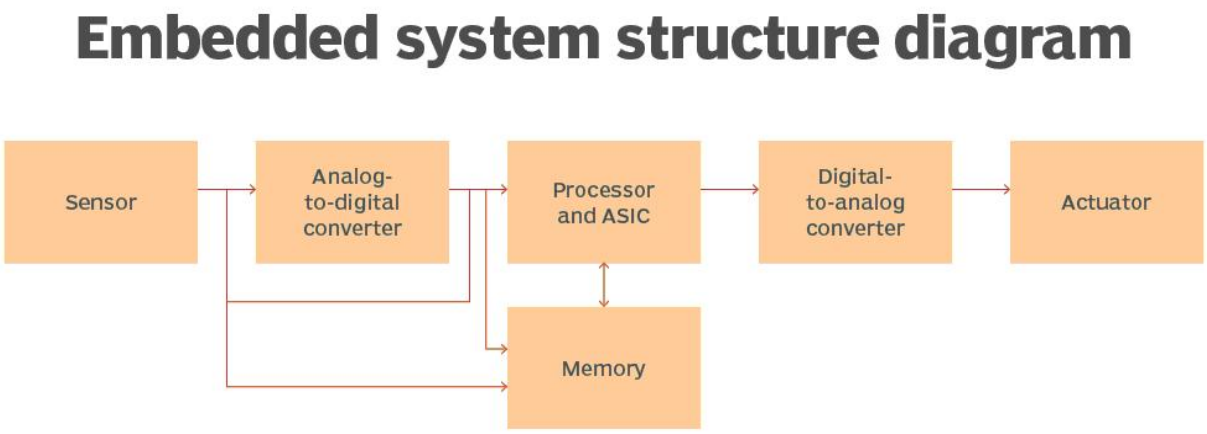

Рисунок 1 – Структура встроенных систем

С точки зрения аппаратного обеспечения базовая встраиваемая система будет состоять из следующих элементов:

— датчики, преобразующие физические данные в электрический сигнал;

— аналого-цифровые преобразователи, преобразующие аналоговый электрический сигнал в цифровой;

— процессоры, обрабатывающие цифровые сигналы и сохраняющие их в памяти;

— цифро-аналоговые (DA) преобразователи, преобразовывающие цифровые данные от процессора в аналоговые данные;

— приводы, сравнивающие фактический выход с сохраненным в памяти выходом и выбирающие правильный;

— датчик, считывающий внешние входные данные.

Преобразователи делают входные данные доступными для чтения процессору, а процессор превращает эту информацию в полезные выходные данные для встроенной системы.

1.3 Программное обеспечение для встраиваемых систем

Одним из ключевых элементов любой встраиваемой системы является программное обеспечение, используемое для запуска микроконтроллера.

Программное обеспечение можно написать разными способами:

— машинный код,

— язык программирования.

Машинный код — это самый простой код, который используется для процессорного модуля. Код обычно представляет собой шестнадцатеричный код и содержит основные инструкции для каждой операции процессора. В наши дни эта форма кода редко используется для встраиваемых систем.

Написание машинного кода очень трудоемко, требует много времени, сложно для понимания и отладки. Поэтому для решения этой задачи очень часто используются языки программирования высокого уровня. Обычно используются такие языки, как C, C++ и т. д.

Код встроенной системы обычно хранится в энергонезависимой памяти на плате процессора. Код называется прошивкой — идея состоит в том, что он не обновляется так же, как программное обеспечение, а хранится во встроенной системе и не может быть изменен пользователем. Часто есть возможность обновить программное обеспечение, но это может означать замену карты памяти, на которой хранится прошивка, или обновление другим способом.

Часто в разработке прошивки могут использоваться дополнительные инструменты. Программы могут усложняться, и необходимо обеспечить правильную работу прошивки встроенной системы.

Таким образом, в разделе рассмотрены основные сферы применения встраиваемых систем. Проанализирована их структура и основные характеристики, позволяющие подобным устройствам оптимизировать многие процессы.

2 Выбор инструментов

Традиционной вычислительной платформой для встраиваемых систем является микроконтроллер. Микроконтроллеры припаяны к печатной плате вместе с памятью, устройствами ввода/вывода (I/O) и другой периферийной электроникой. Существует два предпочтительных метода разработки программного обеспечения: сборка с нуля с использованием библиотек, предоставляемых производителем микросхем и операционная система реального времени (RTOS).

Устройства первого типа хоть и являются гибкими и могут быть настроены для многих приложений, однако они ограничены в ресурсах, таких как память и вычислительная мощность.

У одноплатных компьютеров (SBC), относящихся к устройствам второго типа, есть преимущества перед первыми, поскольку они позволяют разрабатывать код прямо на плате и используя при этом мышь, клавиатуру и монитор. Одноплатный компьютер собран на одной печатной плате, на которой установлены микропроцессор, оперативная память, системы ввода-вывода и другие модули, необходимые для функционирования компьютера.

Обычно в комплект входит операционная система (ОС), наиболее популярной из которых является Linux.

Существует две широкие категории SBC: те, которым нужна пассивная объединительная плата, и те, которые могут стоять отдельно. Те, которые встраиваются в объединительную плату, имеют преимущество стандартного интерфейса для карт расширения (например, ISA, PCI и PC-104), но автономные варианты также имеют средства для расширения, хотя и менее стандартизированные, через один или несколько разъемов [29].

2.1 Анализ рынка одноплатных компьютеров

На рынке одноплатных компьютеров лидирующие места занимают такие модели как Raspberry Pi, Orange Pi Prime, Banana Pi, Rock64, ASUS Tinker board S, Libre Computer Renegade и Libre Computer Renegade Elite.

Для сравнения вышеупомянутых моделей, была составлена таблица их основных характеристик (таблица 1) [12].

Таблица 1 – Характеристики одноплатных компьютеров

Модель	**SoC**	**Процессор**	**Графика**	**Ядра**	**Частота**	**Размер**	**Цена**
Raspberry Pi 3B+	Broadcom BCM2837B0	ARM Cortex A53	Broadcom VideoCore IV	4	1.4 ГГц	85.6 x 56.5 мм	35$
Banana Pi M3	Allwinner A83T	ARM Cortex-A7	PowerVR 544MP1	8	1.8 ГГц	92 x 60 мм	68$
Rock64	Rockchip RK3328	ARM Cortex A53	Mali 450MP2	4	1.5 ГГц	56 x 85 мм	45$
Asus Tinker board S	Rockchip RK3288	ARM Cortex-A17	Mali T760 MP4	4	1.8 ГГц	54 x 86 мм	92$
Libre Computer Renegade	Rockchip RK-3328	ARM Cortex-A53	Mali 450MP2	4	1.5 ГГц	85 x 56 мм	80$
Libre Computer Renegade Elite	Rockchip RK3399	ARM Cortex-A72 + Cortex-A53	Mali-T860	6	2.0 ГГц	120 x 72 мм	100$

Исходя из данных таблицы, можно сказать, что наиболее доступным вариантом является Raspberry Pi, и тем временем почти не уступает по характеристикам своим конкурентам.

Для решения поставленных задач, был сделан выбор в пользу одноплатного компьютера Raspberry Pi 2 Model B. Внешний вид данной модели представлен на рисунке 2.

Рисунок 2 – Raspberry Pi 2 Model B

Raspberry Pi второго поколения использует 4-ядерный процессор BroadcomBCM2836 с архитектурой ARMv7Cortex-A7. Его тактовая частота составляет 900 МГц. Объем памяти - 1 ГБ. На данной модели присутствуют порты Ethernet 100M и USB 2.0 [26].

Raspberry Pi 2 Model B является оптимальным вариантом, поскольку имеет хорошую производительность, возможность установки модуля камеры, достаточное количество портов и расширений и тем самым выигрывает в соотношении цена/качество.

Перед подключением внешних устройств к одноплатному компьютеру, следует сначала разобраться в назначении основных контактов (пинов). Raspberry Pi 2 Model B имеет распиновку, изображенную на рисунке 3.

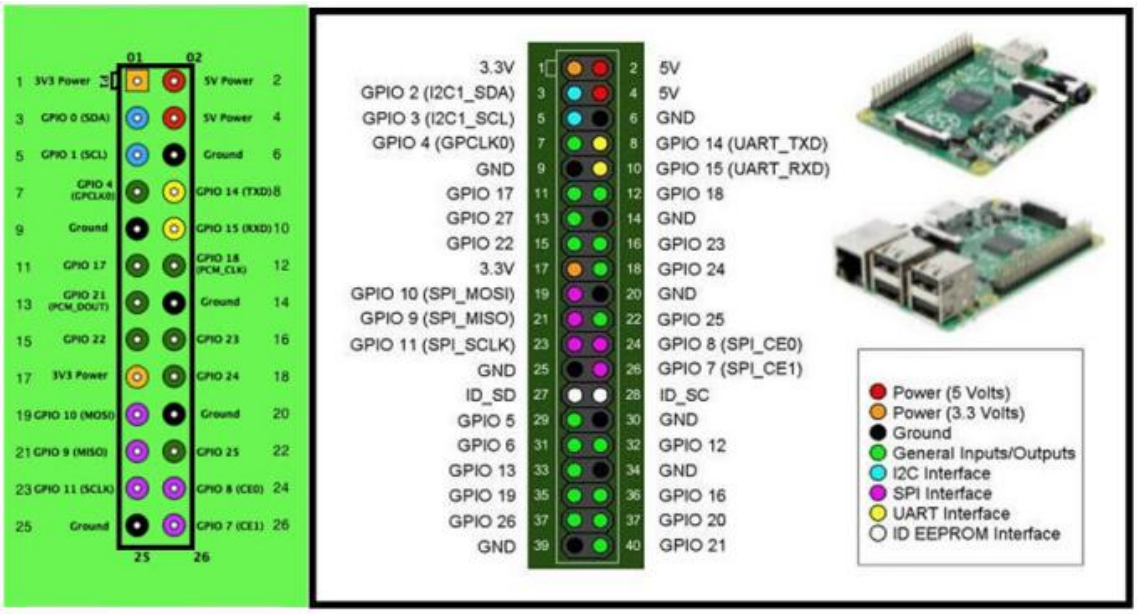

Рисунок 3 – Распиновка Raspberry Pi

У Raspberry Pi распиновка включает в себя два ряда штырьков. Совокупное количество же пинов равняется 40.

Первый, расположенный слева, предназначен для работы с устройствами, для работы которых требуется напряжение величиной 3,3 Вольт. Второй, расположенный слева, служит для работы с устройствами, для которых требуется напряжение 5 Вольт [8].

Следующее, что нужно знать о распиновке GPIO Raspberry Pi – назначение всех штырьков. Всего существует три типа пинов:

- питающие (при включении подают электричество);

- порты (выводящие и принимающие информацию);

- заземляющие.

Питающие пины обозначены на плате словом «Power» (1,2,4,17 пины). Заземляющие – словом «Ground» (6, 9,14,20,25,30,34,39 пины). Выводящие и принимающие информацию порты обозначаются словом «BCM» и занимают все оставшиеся пины на плате [9].

Стоит отметить, что нумерация в Rasberry Pi выполняется по горизонтали. То есть: 1 – 3,3V, 2 – 5V, 3 – порт, 4 – 5V, 5 – порт, 6 – заземление, 7 – порт, 8 – первый порт для 5-вольтных устройств и т.д.

2.2 Анализ современных библиотек компьютерного зрения

Возможности современных программных библиотек компьютерного зрения обеспечивают решение ряда важных практических задач: анализ содержания изображений, поиск и распознавание заданных объектов, выявление текста, отслеживание движений объектов, выявление общих элементов на сравниваемых изображениях, реализация методов обучения для баз видеоданных и т.д [11]. Существует ряд популярных библиотек, возможности которых необходимо проанализировать для выбора одной наиболее подходящей под задачу данной работы.

2.2.1 TensorFlow

TensorFlow - одна из самых популярных комплексных платформ машинного обучения с открытым исходным кодом, включающая в себя полный набор инструментов, ресурсов и библиотек. Данная библиотека особенно полезна для создания и развертывания приложений, связанных с компьютерным зрением, в основе которого лежит машинное обучение.

TensorFlow является одним из самых простых инструментов компьютерного зрения, при этом позволяя пользователям рсамостоятельно азрабатывать модели машинного обучения, связанные с компьютерным зрением. Библиотека решает ряд таких задач как распознавание лиц, классификация изображений, обнаружение объектов и многое другое. Tensorflow поддерживает различные языки, такие как Python, C, C ++, Java или JavaScript.

Для реальных проектов компьютерного зрения TensorFlow Lite представляет собой упрощенную реализацию для машинного обучения на устройстве с использованием периферийных устройств.

К достоинствам данной библиотеки можно отнести совместимость с несколькими языками, обеспечение постоянных обновлений и хорошую производительность. К недостаткам TensorFlow можно отнести чрезвычайно ресурсоемкий инструментарий [10].

2.2.2 CUDA

CUDA (Compute Unified Device Architecture) — это платформа параллельных вычислений и модель интерфейса прикладного программирования (API), разработанная американской технологической компанией NVIDIA. Платформа позволяет разработчикам использовать мощность графических процессоров для ускорения работы ресурсоемких приложений.

В набор инструментов входит библиотека NVIDIA Performance Primitives (NPP), которая обеспечивает ускорение изображений, видео и сигналов с GPU-ускорением для различных областей, включая компьютерное зрение. Кроме того, архитектура CUDA полезна для широкого круга задач, таких как распознавание лиц, обработка изображений, воспроизведение 3D-графики и т.д. Обработка изображений в реальном времени с помощью Nvidia CUDA поддерживается для периферийных реализаций искусственного интеллекта, чтобы выполнять вывод ИИ на периферийных устройствах, таких как Jetson TX2 [30].

Он поддерживает различные языки программирования, включая C, C++, Python, Fortran или MATLAB, а также совместим с большинством операционных систем.

К преимуществам данной библиотеки относится большое количество встроенных примитивов для обработки изображений и сигналов, быстрота и эффективность, высокая мощность и производительность при анализе видео,

поддержка нескольких языков. К недостаткам можно отнести довольно большое энергопотребление и ограниченную кросс-платформенную гибкость.

2.2.3 OpenCV

Другой не менее популярной библиотекой компьютерного зрения является OpenCV.

OpenCV — это библиотека программного обеспечения для машинного обучения и компьютерного зрения с открытым исходным кодом. Созданный с целью предоставления общей инфраструктуры для приложений компьютерного зрения, OpenCV обеспечивает доступ к более чем 2500 классическим и современным алгоритмам.

Эти алгоритмы полезны для многих задач, включая обнаружение и распознавание лиц, устранение эффекта красных глаз, идентификацию объектов, извлечение 3D-моделей объектов, отслеживание движущихся объектов и объединение нескольких кадров в изображение с высоким разрешением [21].

OpenCV имеет несколько интерфейсов, таких как C ++, Python, Java и MATLAB, и поддерживает большинство операционных систем, включая Windows, Android, Linux и Mac. Библиотека компьютерного зрения широко используется международными компаниями, включая Google, Facebook, IBM, Toyota, Sony, Honda и Microsoft.

К преимуществам OpenCV можно отнести доступ к более чем 2500 алгоритмам, возможность настройки кода для достижения определенных целей, стандартный инструмент для обработки изображений и поддержку большого сообщества. Однако изучение данной библиотеки может показаться несколько сложным, особенно тем, кто только начинает обучение в сфере компьютерного зрения.

2.2.4 MatLab

Чаще всего процесс обучения компьютерному зрению в университетах начинают с программы MatLab. Она имеет специальный инструмент для компьютерного зрения.

Инструмент Computer Vision Toolbox предоставляет алгоритмы и инструменты для проектирования и моделирования систем компьютерного зрения, а также обработки видео. Он позволяет обнаруживать, отслеживать и распознавать объекты в видеокадрах, а также обнаруживать лица и определенные черты лица человека.

Набор инструментов предоставляет алгоритмы и функции для создания собственных систем распознавания и поиска изображений. Можно создать свою систему обнаружения или распознавания объектов, выбрав и назначив интересующие объекты и обучив классификатор. Обнаруженные объекты можно отслеживать во времени с помощью алгоритмов KLT и фильтра Калмана.

Таким образом, можно сделать следующие выводы: MATLAB более удобен в разработке и представлении данных, однако OpenCV намного быстрее в исполнении. В то же время OpenCV сравнительно сложнее изучить из-за отсутствия документации и кодов обработки ошибок. Этот недостаток заставляет начинающих пользователей компьютерного зрения чаще склоняться к MATLAB. Но после приобретения опыта работы с OpenCV некоторые профессионалы предлагают придерживаться его, поскольку это наиболее полная библиотека с открытым исходным кодом для компьютерного зрения и имеет большое сообщество пользователей [22].

Некоторые профессионалы также предлагают MATLAB, поскольку он полезен для быстрого прототипирования, а его код очень легко отлаживать. К тому же у него хорошая документация и поддержка.

Недостатком MATLAB является то, что он не обладает открытым исходным кодом, и лицензия довольно дорогая, а его программы не портируются. Однако MATLAB - это полный научный пакет, состоящий из массивной IDE со своим собственным языком.

Таким образом, MATLAB подходит для изучения концепций компьютерного зрения в качестве исследователей и студентов университетов, которые могут позволить себе это программное обеспечение, но OpenCV

удобнее при создании готовых к производству реальных проектов компьютерного зрения.

Что касается TensorFlow, то эта библиотека идеально подходит для сложных ситуаций, где невозможно обойтись силами OpenCV. То есть это библиотека, предназначенная больше для машинного обучения, нежели для считывания изображения и распознавания обыкновенных объектов на простых фонах. Поэтому, как правило, большинство специалистов в области компьютерного зрения начинают с OpenCV, а к помощи TensorFlow прибегают в сложных случаях.

Таким образом, в данном разделе были проанализированы современные модели одноплатных компьютеров. В качестве инструмента для выполнения выпускной квалификационной работы был выбран одноплатный компьютер Raspberry Pi 2 Model B. В результате исследования возможностей современных библиотек компьютерного зрения, было принято решение начать реализовывать поставленную задачу, используя библиотеку OpenCV.

3 Разработка программного обеспечения

В качестве инструментов для выполнения работы был выбран одноплатный компьютер Raspberry Pi 2 Model B, библиотека компьютерного зрения OpenCV и язык программирования Python. Выбор данного языка программирования обусловлен следующими его преимуществами:

— встроенная поддержка "длинной арифметики", комплексных чисел, списков, словарей, стеков, очередей и т.д;

— кроссплатформенность;

— поддержка всех кодировок;

— интерпретируемость языка;

— легкость в написании кода, что позволяет выражать алгоритмы кратко и просто.

В последние годы Python стал одним из самых популярных языков программирования в мире. Он используется во всем, от машинного обучения до создания веб-сайтов и тестирования программного обеспечения. Его могут использовать как разработчики, так и начинающие программисты.

Из сотен существующих языков программирования Python остается популярным среди таких известных компаний и организаций как Google, Meta, Venmo, Spotify, Netflix и Dropbox.

Язык предоставляет несколько библиотек и сред компьютерного зрения, которые помогают автоматизировать задачи обнаружения и визуализации.

Перед началом работы по распознаванию человеческих образов, необходимо настроить оборудование и ознакомиться с простейшими алгоритмами распознавания объектов в выбранной библиотеке.

3.1 Подготовка оборудования

При разработке схемотехнической части электронного устройства в первую очередь необходимо составить структурную схему.

На рисунке 4 представлена структурная схема разрабатываемого устройства.

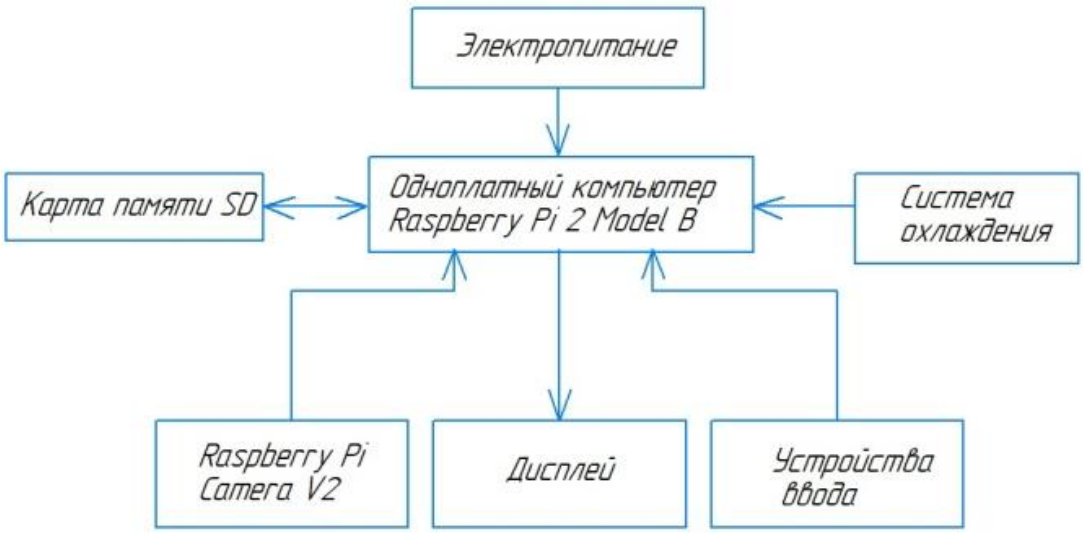

Рисунок 4 – Структурная схема разрабатываемого устройства

Таким образом, в разрабатываемое устройство включены:

— одноплатный компьютер Raspberry Pi;

— модуль камеры Raspberry Pi;

— вентилятор, осуществляющий охлаждение быстро нагревающегося компьютера;

— карта SD;

— дисплей;

— устройства ввода (клавиатура и компьютерная мышь);

— источник электропитания с номинальным напряжением 5V и минимальным током 1,8 А.

Схема электрическая соединений разрабатываемой системы представлена на рисунке 5.

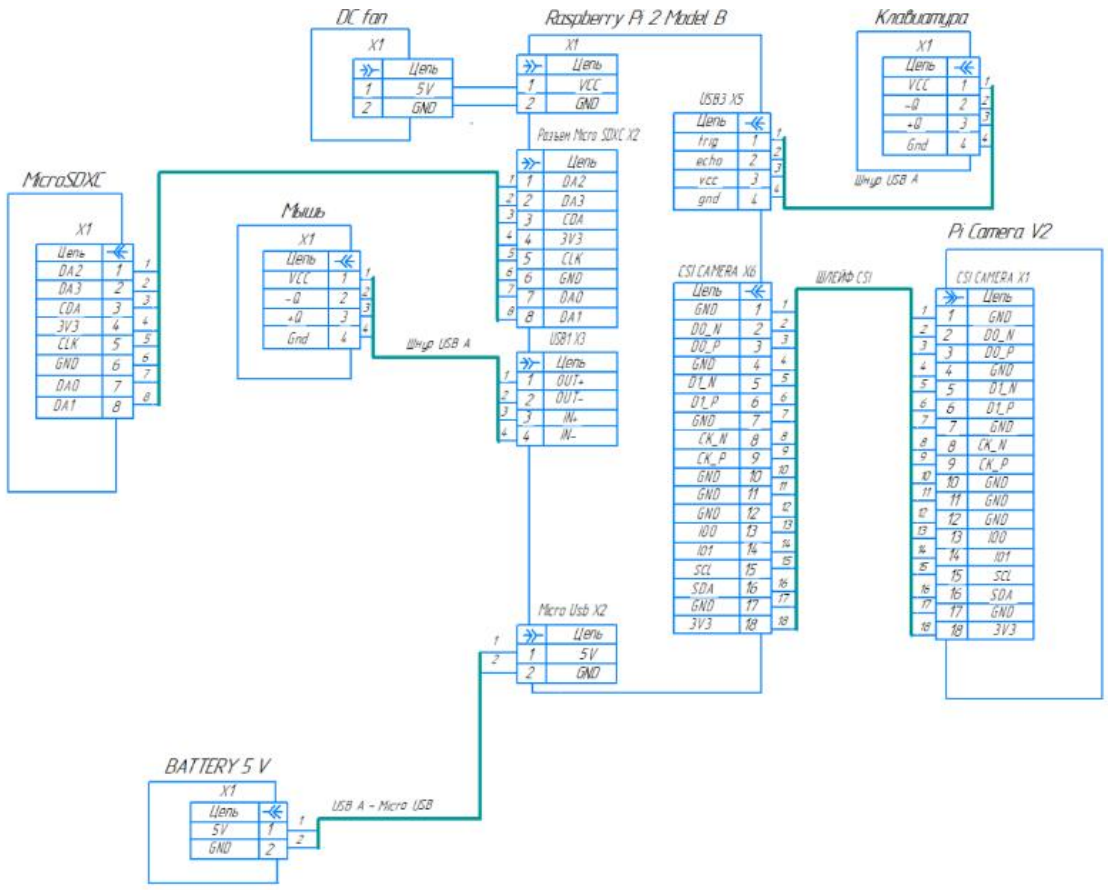

Рисунок 5 – Схема электрическая соединений

Как видно из рисунка, большинство элементов подключено к одноплатному компьютеру через USB-порты. В качестве источника питания выступает внешний аккумулятор, имеющий напряжение не менее 5V.

Подготовка к работе Raspberry Pi состоит из следующих этапов:

— подключение вентилятора,

— настройка ПО,

— подключение камеры.

Подключение вентилятора к компьютеру считается обязательным шагом перед работой с Raspberry Pi. Это связано с тем, что одноплатные

компьютеры довольно быстро нагреваются. При высоких температурах подобные устройства нередко выходят из строя и не подлежат дальнейшему их использованию. Схема подключения вентилятора к Raspberry Pi представлена на рисунке 6.

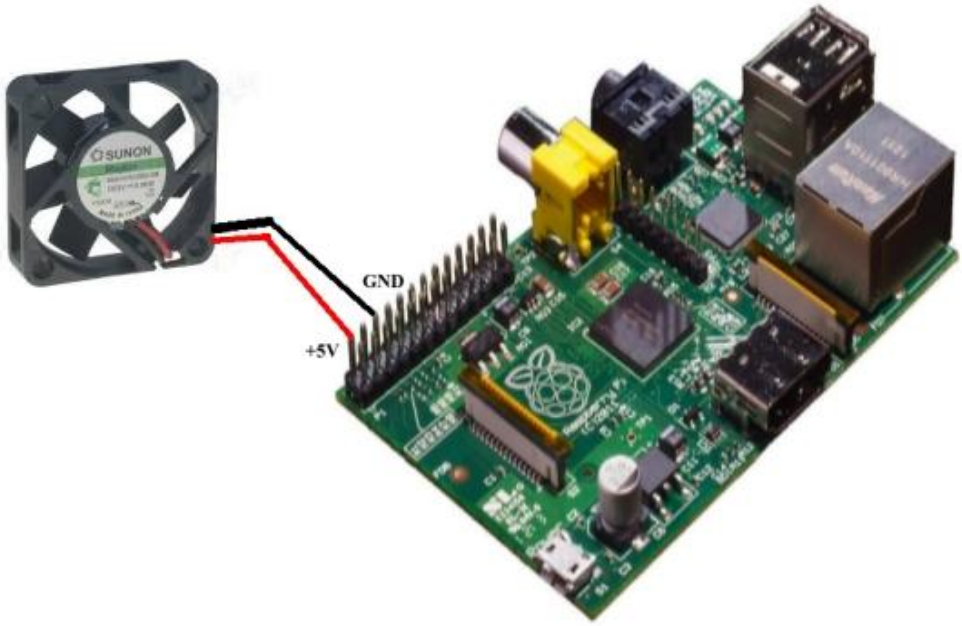

Рисунок 6 - Подключение вентилятора к Raspberry Pi

Как видно из рисунка, вентилятор располагается над радиатором и имеет только 2 провода: питание и заземление.

Настройка ПО Raspberry Pi состоит из нескольких шагов, строгое выполнение которых необходимо для дальнейшей работой с системой.

Первый шаг — загрузить Raspberry Pi Imager с официального сайта Raspberry Pi Этот инструмент позволит выбрать ОС, автоматически загрузить ее и записать на SD-карту по вашему выбору.

Инструмент Raspberry Pi Imager доступен в Windows, macOS и Ubuntu (рисунок 7).

Рисунок 7 – Загрузка Raspberry Pi

После выбора и загрузки нужной операционной системы Raspberry Pi Imager необходимо следовать инструкциям по установке (рисунок 8).

Рисунок 8 – Raspberry Pi Imager

Второй шаг — Выбор ОС в Raspberry Pi Imager.

В Raspberry Pi Imager доступны 3 версии ОС Raspberry Pi (рисунок 9). В данной работе был сделан выбор в пользу ОС Raspberry Pi (32-bit).

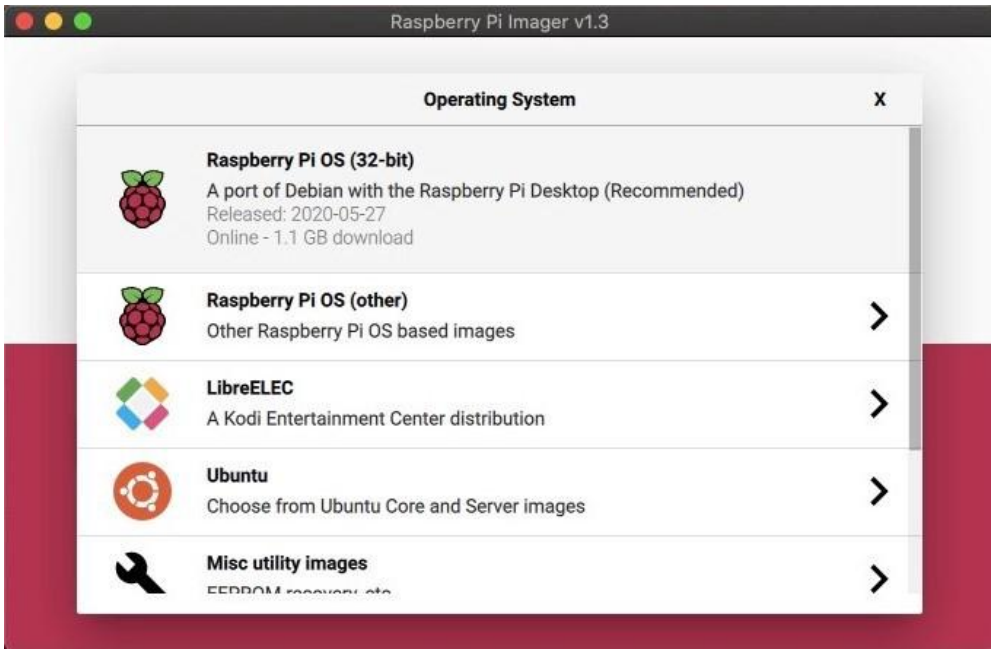

Рисунок 9 – Выбор ОС в Raspberry Pi Imager

Данная версия включает в себя графический интерфейс и больше установленного программного обеспечения, чем неполная версия. Размер составляет около 2,5 ГБ.

Третий шаг — выбор SD-карты.

На данном этапе необходимо подключить к одноплатному компьютеру SD-карту, чтобы скопировать выбранную ранее ОС.

На рисунке 10 показано меню, в котором необходимо выбрать SD-карту, подключенную к компьютеру.

Рисунок 10 – Выбор SD-карты

Четвертый шаг — Запись ОС на SD-карту.

На рисунке 11 показано меню, в котором необходимо нажать на кнопку «WRITE». Этот шаг запишет выбранную ОС на SD-карту и запустит проверку успешного копирования.

Рисунок 11 – Запись ОС на SD-карту

Скорость процесса записи зависит от выбранной ОС. Как правило, это занимает всего несколько минут.

После получения сообщения об успешной записи ОС на SD-карту (рисунок 12) можно перейти к загрузке Raspberry Pi.

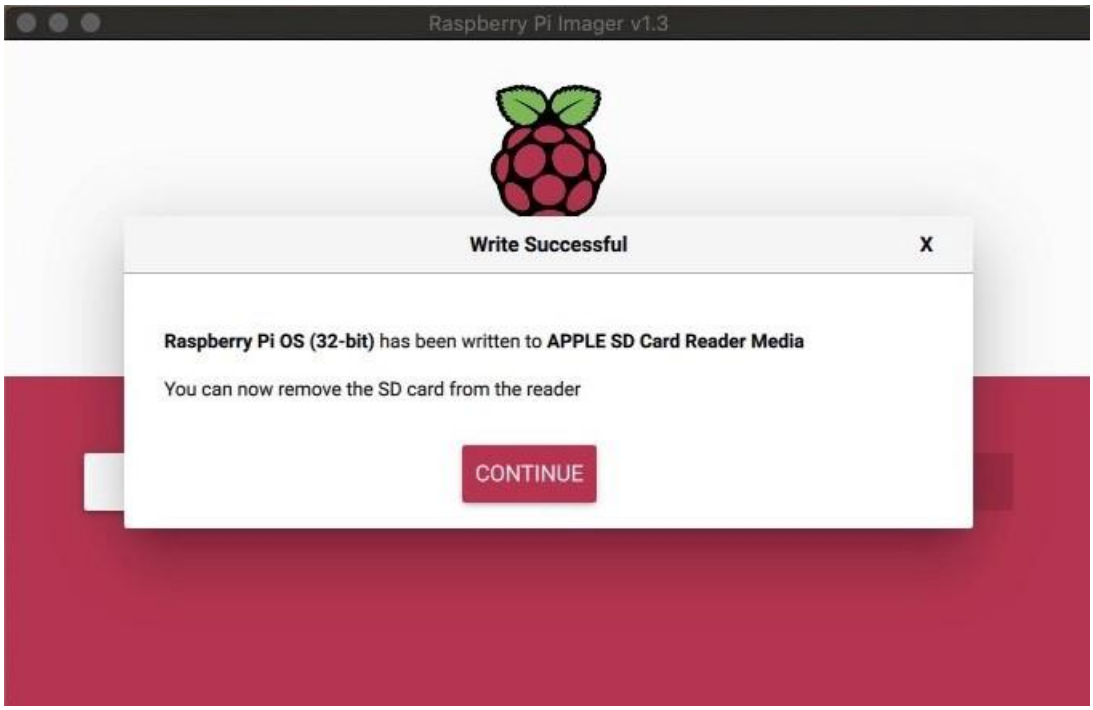

Рисунок 12 – Сообщение об успешном копировании

Пятый шаг — Загрузка Raspberry Pi

Для реализации данного шага необходимо вставить карту microSDHC в Raspberry Pi. Затем подключить Raspberry Pi к источнику питания, клавиатуре, мыши и монитору.

При включении компьютера, в первую очередь на экран выведется диалоговое окно «Добро пожаловать в Raspberry Pi» (рисунок 13).

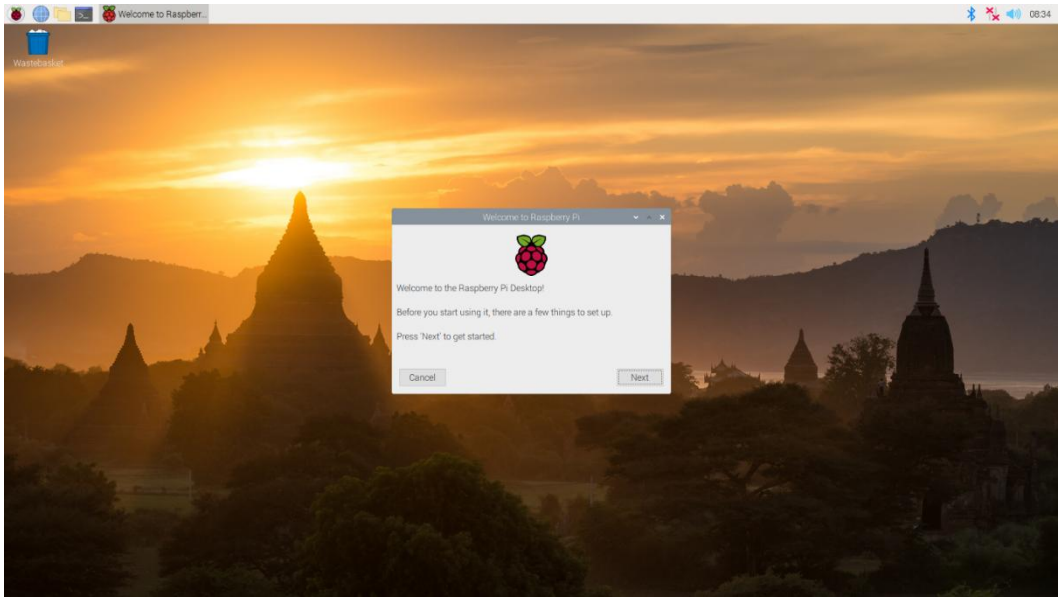

Рисунок 13 – Диалоговое окно при первом включении компьютера

Далее следует выбор страны, языка, часового пояса, а также рекомендуется включить надежные пароли для учетных записей, поскольку по умолчанию в Raspberry Pi всегда устанавливается имя пользователя — «pi», а пароль — «малина».

На следующем экране пользователю задается вопрос, есть ли вокруг рабочего стола черная рамка, поскольку рабочий стол должен занимать весь экран. Если это не так, ОС Raspberry Pi может внести коррективы, чтобы заполнить черное пространство. Это изменение вступит в силу после перезапуска Raspberry Pi.

На следующем экране будет предложено подключить Raspberry Pi к беспроводной сети (рисунок 14).

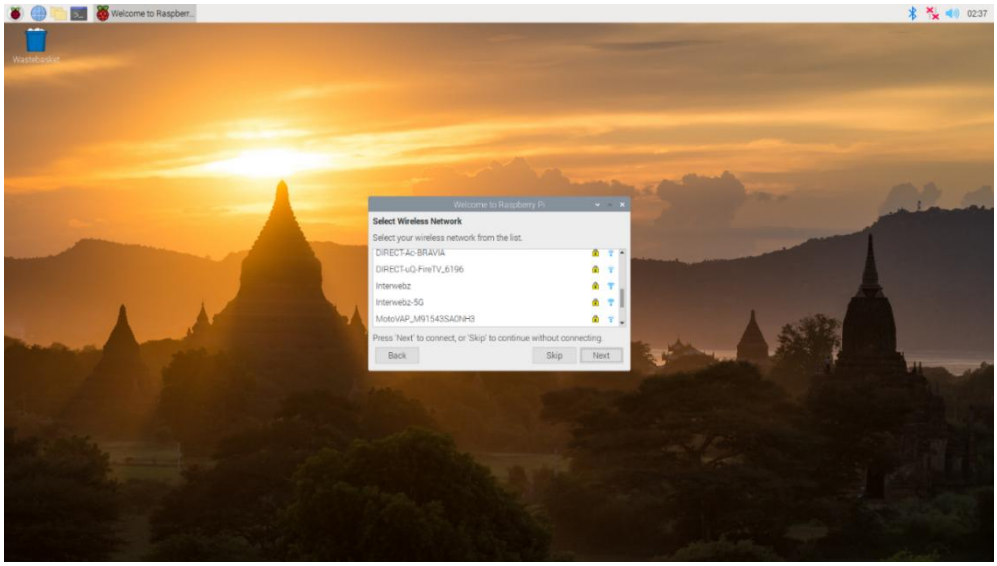

Рисунок 14 – Выбор доступных беспроводных сетей

На следующем экране система спросит, хотите ли вы проверить и при необходимости обновить операционную систему и приложения. Для выполнения этого шага требуется подключение к Интернету. Данный этап можно пропустить.

Последний экран настройки сообщает, что настройка завершена и Raspberry Pi готов к работе (рисунок 15).

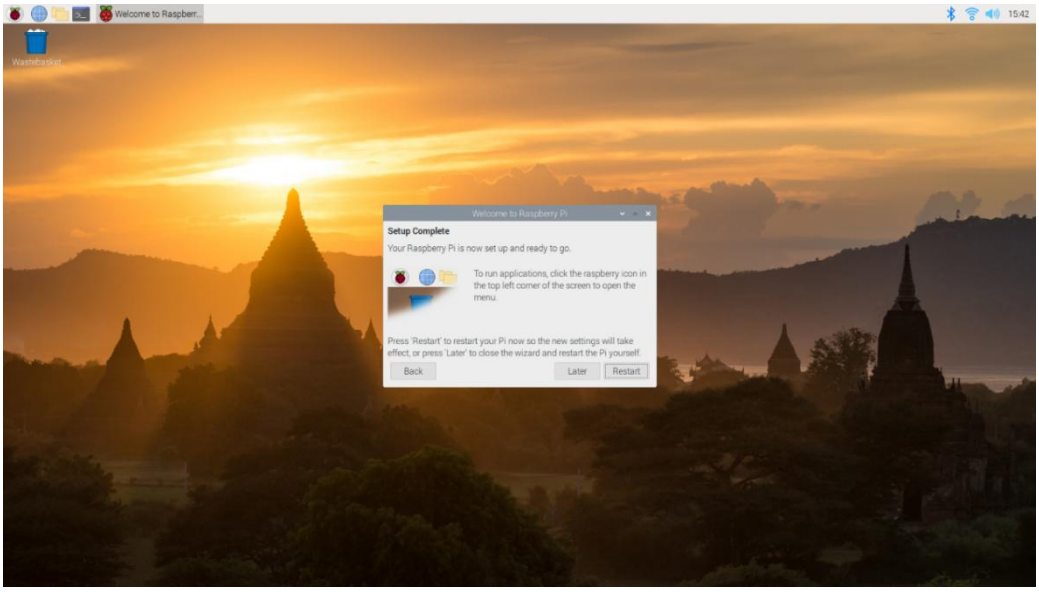

Рисунок 15 – Завершение настройки Raspberry Pi

Следует перезапустить Raspberry Pi, чтобы любые системные изменения вступили в силу [13].

Далее необходимо подключить модуль камеры Raspberry Pi. Он имеет шлейф, который подключается в CSI-разъем одноплатного компьютера.

CSI разъем для камер Raspberry Pi заслуживает особого внимания. Взаимодействие CAM1_CK_S и CAM1_CK-P обеспечивает тактовый импульс для полос данных Mipi для камеры. С помощью них они подключаются к контактам MIPIM Clock Positive (MPCP) и MIPIM Clock (MCN) ICC камеры.

Настройка контактов Cam1 D0_N, а также контакты CAM1 D0_P предназначены для вывода данных MIPI Data Positive (MDpi) и MDN Negative. CAM1_D1-N и CAM1_D2P - предназначены для вывода данных MIPI Data Positive (MDPI) и MIPI Data negative (MDN) в полосу данных 1 камеры 1.

SCL и SDA – являются двумя основными шинами, которые играют роль меньшей последовательной шины и обеспечивают последовательную связь, благодаря которой пользователь может выбрать разрешение камеры. Эти контакты нужно подключать прямо к ведомым интерфейсам SCCB, находящимся внутри IC камеры.

Встроенный выход SCL обеспечивает стандартный входной тактовый сигнал последовательного интерфейса и обычный последовательный доступ SDA для ввод/выпуск данных [27].

Схема подключения модуля камеры к Raspberry Pi изображена на рисунке 16.

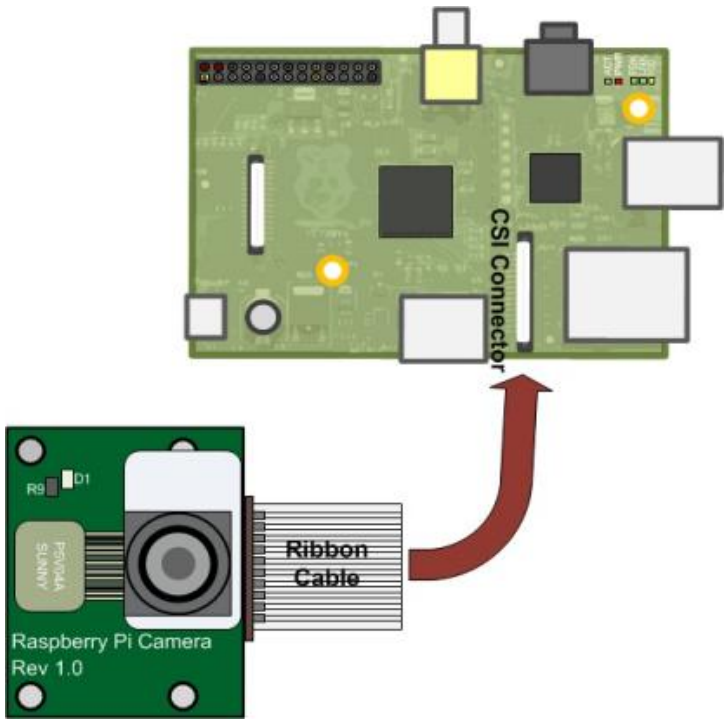

Рисунок 16 – Схема подключения модуля камеры к Raspberry

После этого необходимо программно обеспечить подключение камеры. Необходимо открыть утилиту конфигурирования и найти в ней вкладку «Interfaces» (рисунок 17).

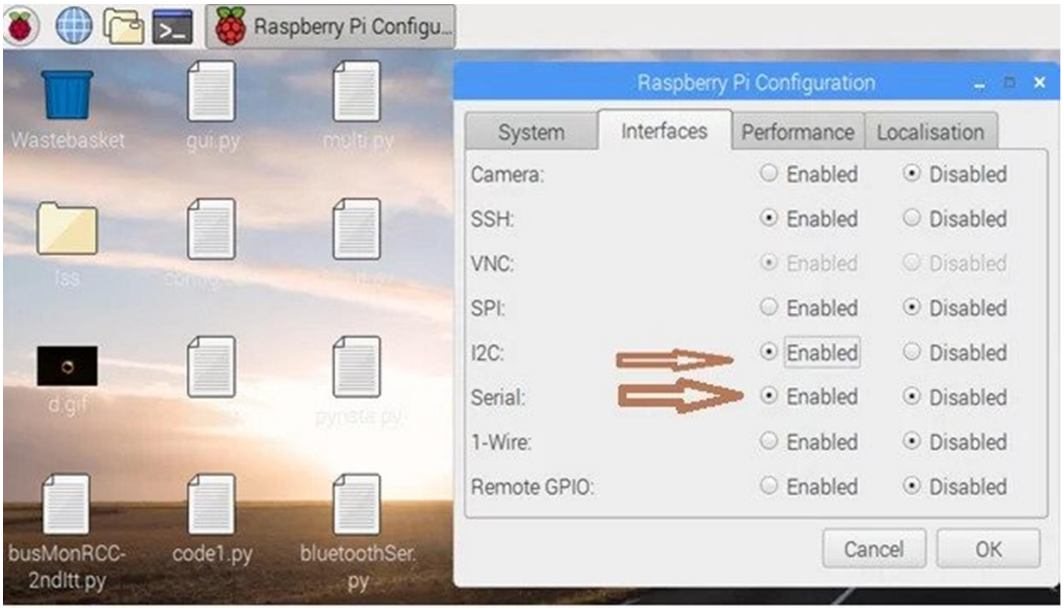

Рисунок 17 – Утилита конфигурирования Raspberry Pi

Оказавшись в данном меню, потребуется найти переключатель Camera, поставить его в положение «Enabled» и перезагрузить компьютер.

3.2 Начало работы с библиотекой OpenCV

Самым начальным этапом работы с данной библиотекой является загрузка, отображение и сохранение изображения.

Вышеперечисленные действия с использованием языка программирования Python производятся следующим образом:

```
def image_loading():
    image = cv2.imread('123.jpg',cv2.IMREAD_GRAYSCALE)
    cv2.imshow('image_loading', image)
    cv2.waitKey(0)
    cv2.imwrite('image_gray', image)
```

Функция cv2.imread() служит для загрузки изображения в программу, первым ее аргументом является путь к изображению, а вторым аргументом указывается, в каком цветовом пространстве будет считано считать изображение и по умолчанию он принимает значение cv2.IMREAD_COLOR. Для того чтобы считать изображение в цветовом пространстве RGB следует указать аргумент cv2.IMREAD_COLOR, в оттенках серого — cv2.IMREAD_GRAYSCALE. В зависимости от выбранного цветового пространства функция вернет либо 2D, либо 3D массив NumPy.

Функция cv2.imshow() служит для отображения изображения на экране. Первым аргументов функции является название окна, а вторым аргументом изображение, загруженное с диска.

Благодаря функции cv2.waitKey() выполнение программы будет приостановлено, и полученное изображение будет отображаться на экране до тех пор, пока пользователь не нажмет на любую кнопку.

Функция cv2.imwrite() записывает полученное изображение в файл в формате jpg, где первый аргумент – название будущего файла и его расширение, а второй - изображение, которое мы хотим сохранить.

Для определения высоты, ширины и количества каналов у изображения используется атрибут shape:

Высота изображения - img.shape[0],

Ширина изображения - img.shape[1],

Количество каналов - img.shape[2].

Вычисление количества каналов доступно только для цветных изображений, поскольку изображения в оттенках серого представлены в виде 2D массива [20].

Также можно получить доступ к значению определенного пикселя путем указания его координат x и y. Однако важно помнить, что библиотека OpenCV хранит каналы формата RGB в обратном порядке – т.е. в порядке синего, зеленого и красного цветов.

3.2.1 Обнаружение простейших геометрических фигур

После того, как изображение успешно загружено в программу и готово к дальнейшей обработке, можно приступить к нахождению простейших геометрических фигур.

Для этого следует воспользоваться функцией minAreaRect(), служащей для поиска прямоугольников на изображении [25].

Функция minAreaRect() пытается найти прямоугольник максимального размера, который может вписаться в заданный замкнутый контур. Важно отметить, что данная функция не определяет, является ли контур прямоугольным, она пытается вписать в него прямоугольник оптимальным способом (рисунок 18)

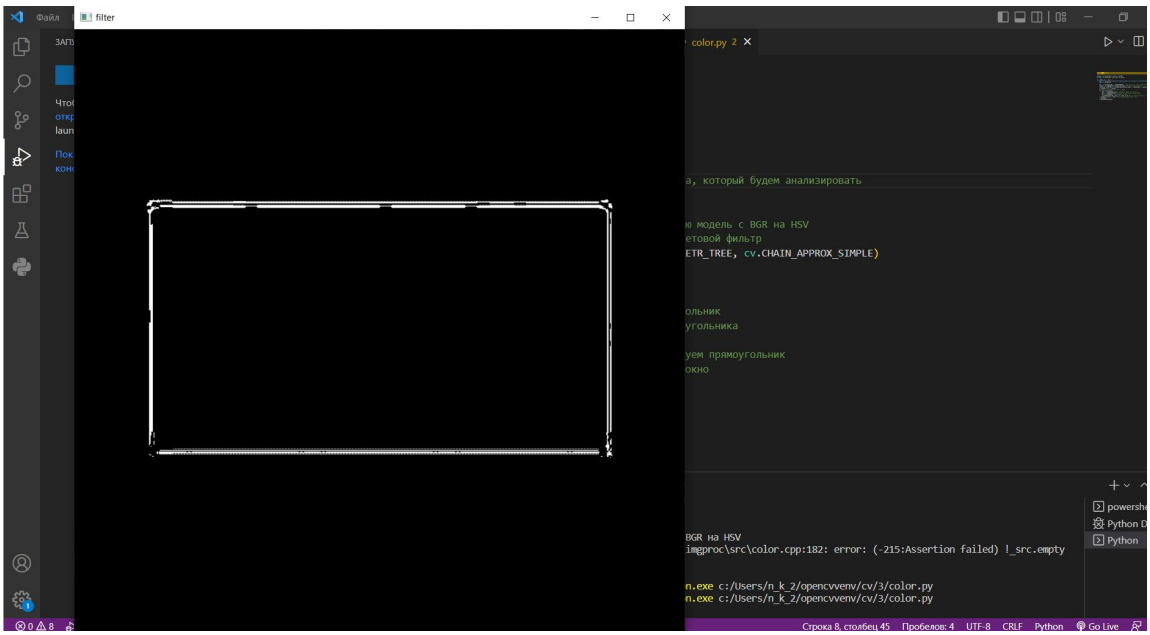

Рисунок 18 – Результат обнаружения прямоугольника на изображении

Для того чтобы обнаружить круги на изображениях, необходимо воспользоваться функцией cv2.HoughCircles():

cv2. HoughCircles (image, method, dp, minDist), где

image: 8-битное, одноканальное изображение.

method: Определяет метод обнаружения кругов на изображениях.

minDist: Минимальное расстояние между координатами центра (x, y) обнаруженных кругов. Если minDist слишком маленький, то функция может ошибочно обнаружить несколько кругов в одном и том же месте. Если minDist слишком большой, то некоторые круги могут вовсе не обнаружиться.

param1: Значение градиента, используемое для обработки краев.

param2: Пороговое значение для метода cv2.HOUGH_GRADIENT. Чем меньше порог, тем больше кругов будет обнаружено (включая ложные круги). Чем больше порог, тем потенциально будет возвращено большее количество кругов.

minRadius: Минимальный размер радиуса (в пикселях).

maxRadius: Максимальный размер радиуса (в пикселях) [15].

Результат выполнения программы по нахождению кругов на изображении представлен на рисунке 19.

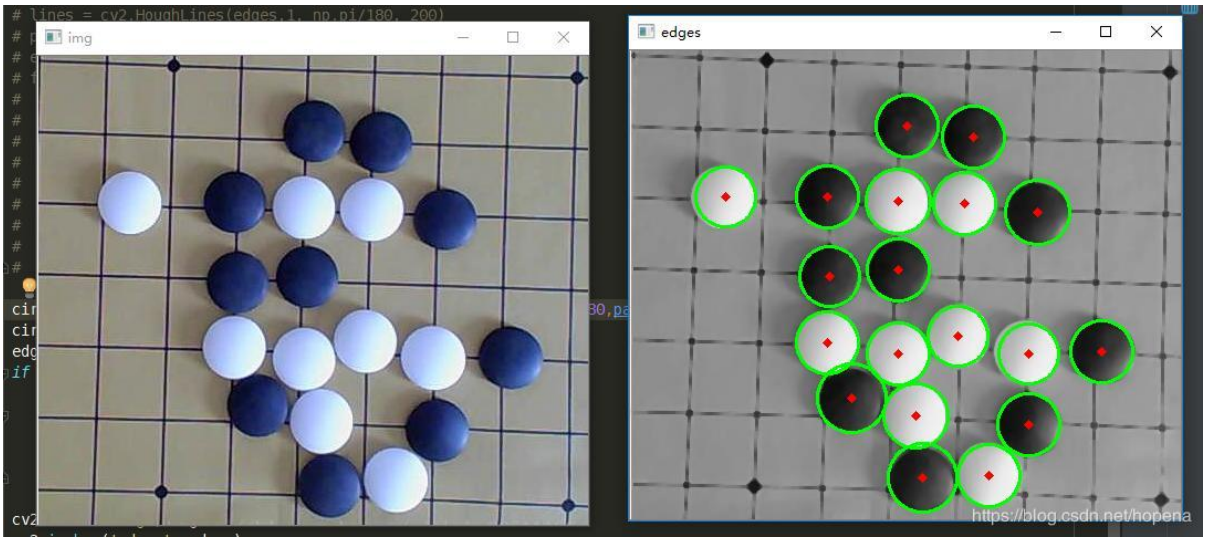

Рисунок 19 – Результат обнаружения кругов на изображении

С помощью функции aboutPolyDP(), служащей для вычисления и аппроксимации полигональной кривой с заданной точностью, можно распознать более сложные геометрические фигуры. Ее суть заключается в том, что она приближает форму контура к другой форме с меньшим количеством вершин в зависимости от указанной точности.

Функция BoundingRect() определяет граничные точки прямоугольника, а функция putText() помещает текст поверх изображения. Результат выполнения программы приведен на рисунке 20.

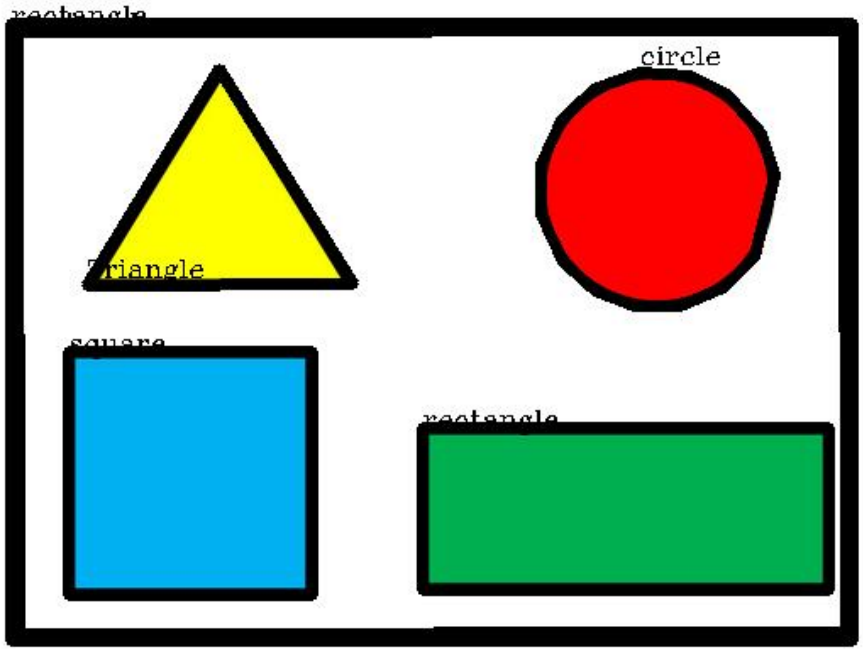

Рисунок 20 – Результат распознавания различных геометрических фигур на изображении

3.2.2 Реализация алгоритма поиска цветового пятна в OpenCV

Стандартный прием для поиска цветового пятна в OpenCV складывается из двух этапов:

На первом этапе применяется цветовой фильтр, и кадр преобразуется в черно-белое изображение. После объект искомого цвета превращается в белое пятно, а всё остальное заливается черным цветом.

На втором этапе используется алгоритм вычисления моментов. Момент изображения — это суммарная характеристика пятна, представляющая собой сумму всех точек (пикселей) этого пятна. При этом имеется множество подвидов моментов, характеризующие разные свойства изображения. Например, момент нулевого порядка m00 — это количество всех точек, составляющих пятно. Момент первого порядка m10 представляет собой

сумму x координат точек, а m01 — сумму y координат. Имеются также моменты m11, m20, m02, m22 и т.д.

Формула для вычисления моментов проста, и возможно посчитать пиксели вручную. Однако стандартные функции OpenCV написаны на языках более низкого уровня, чем python, и работают быстрее. Стандартная функция для вычисления моментов выглядит следующим образом:

moments(img, binary),

где img – это предварительно обработанное изображение, а binary - указание алгоритма вычисления веса каждой точки [23].

В черно-белом изображении, полученном на первом этапе, пиксели могут быть черными, белыми, а могут быть и серыми. В таком случае, если аргумент «binary» равен 1, то вес всех точек с цветом, отличным от нуля будет равен единице. В противном случае, вес черной точки будет равен 0, а белой точки — 255.

Результатом функции moments является массив до третьего порядка. Чтобы вычислить координаты центра пятна искомого цвета понадобятся только моменты первого и нулевого порядка:

m01 = moments['m01']

m10 = moments['m10']

dArea = moments['m00']

Для определения координат x и y, моменты первого порядка делятся на нулевой момент:

x = int(m10 / dArea)

y = int(m01 / dArea)2

Таким образом будут найдены средние координаты всех точек, что и будет являться центром искомого пятна.

Для написания данной программы на языке Python, необходимо в самом начале импортировать необходимые пакеты для работы с компьютерным зрением (OpenCV), массивами (NumPy). В данном случае

необходимо также подгрузить файл «video», в котором находится код для получения видеопотока с камеры.

```
import  numpy
import  cv2
import video
```

Далее нужно задать верхнюю и нижнюю границы интересующего нас цвета - в данном случае желтого.

```
min = numpy.array((0, 51, 148), numpy.uint8)
max = numpy.array((84, 159, 254), numpy.uint8)
```

В основном цикле программы происходит преобразование изображения в черно-белый вариант и наложение фильтра на объект. Искомый цвет преобразутеся в белое пятно, а всё остальное – в черный. С помощью функции moment() и деления полученных моментов m10 и m01 на нулевой момент m00, находим средние координаты объекта, обводим его в круг и выводим на исходном изображении.

```
    if Area > 100:
        x = int(m10 / Area)
        y = int(m01 / Area)
        cv2.circle(image, (x, y), 10, (0,0,255), -1)
    cv2.imshow('yellow_object', image)
```

Результат выполнения программы по обнаружению объекта заданного цвета представлен на рисунке 21.

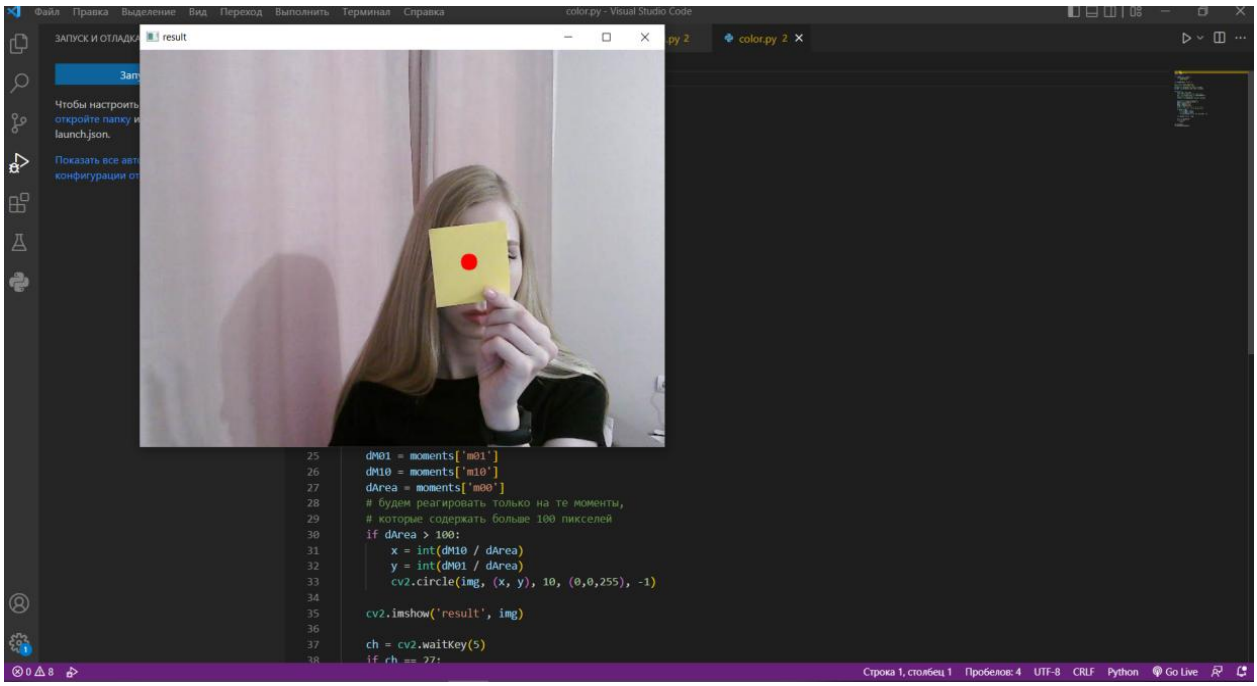

Рисунок 21 – Обнаружение объекта заданного цвета

3.2.3 Вывод координат искомого объекта заданного цвета

Для того чтобы лучше понимать, в какой части кадра находится объект, к действующей программе был добавлен вывод координат рядом с обнаруженным центром искомого цвета с помощью функции cv2.putText():

cv2.putText (img, "%d-%d" % (x,y), (x+10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, red, 2)

Также в программу была добавлена условная территория, на которой может находиться объект – выделение ее происходит с помощью функции cv2.rectangle и окрашивание контур в красный цвет:

cv2.rectangle(img, (100,0), (650,500), red, thickness=2, lineType=8, shift=0)

После чего вывод координат был сделан интуитивно понятным – если объект находится внутри прямоугольника – координаты выводятся зеленым цветом, иначе – красным. Разделив кадр на области, появляется возможность отслеживать положение объекта.

Пример вывода координат при обнаружении объекта в пределах границы представлен на рисунке 22.

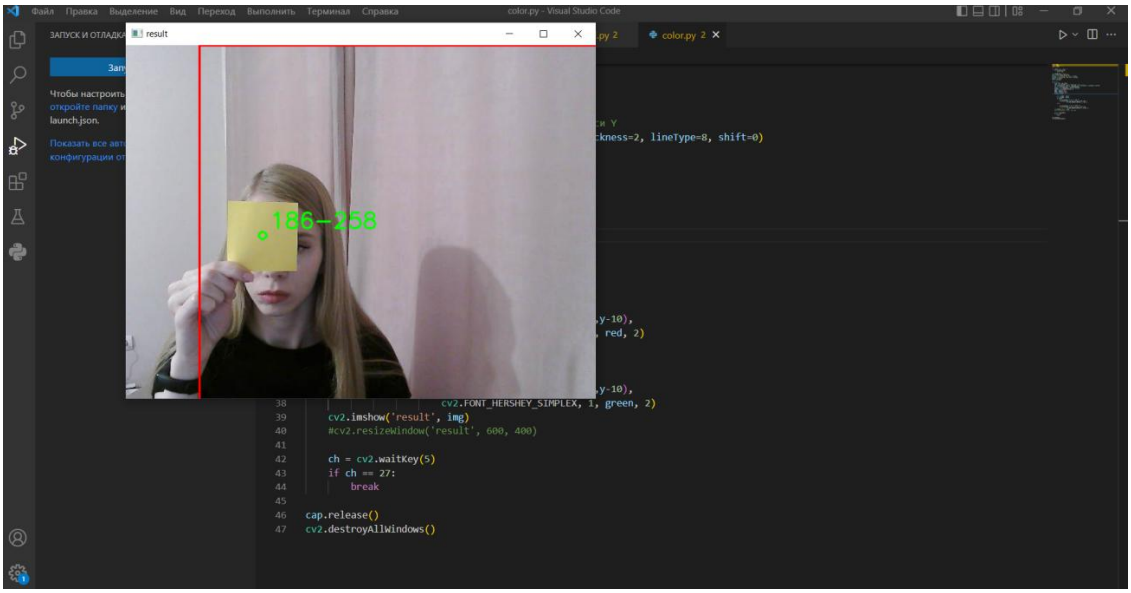

Рисунок 22 – Обнаружение объекта в пределах границы

Пример вывода координат при обнаружении объекта за пределами границы представлен на рисунке 23.

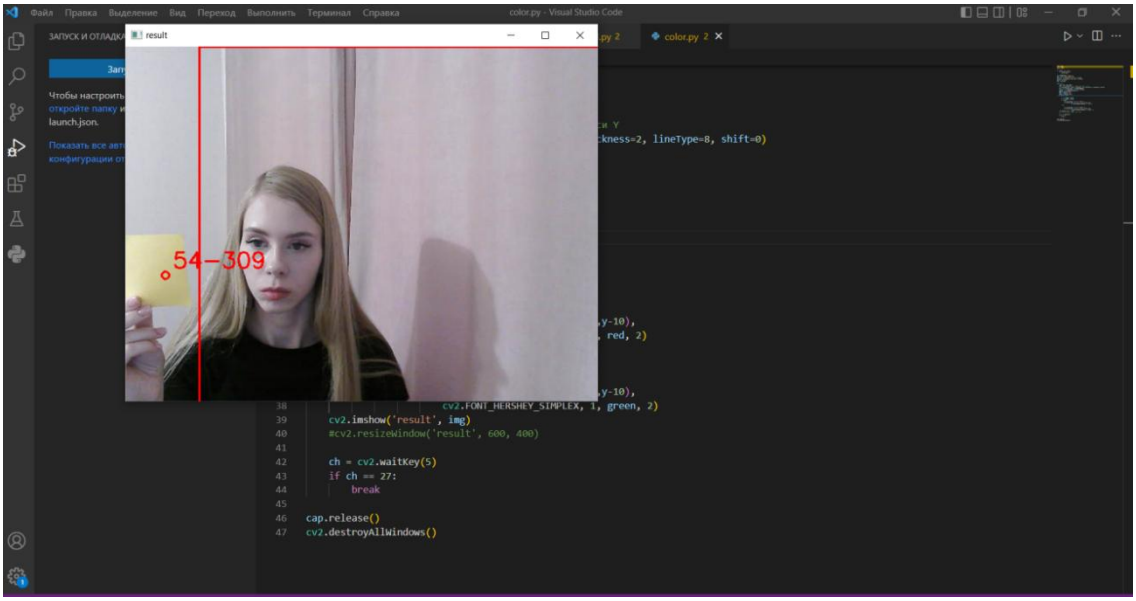

Рисунок 23 – Обнаружение объекта за пределами границы

Таким образом, в разделе представлена структурная схема и схема электрическая соединений разрабатываемой системы и перечень всех используемых элементов. Рассмотрены все этапы подготовки к работе одноплатного компьютера, в том числе шаги, необходимые для настройки программного обеспечения Raspberry Pi. Приведены основы работы с библиотекой компьютерного зрения OpenCV при использовании языка программирования Python, в том числе определение простейших геометрических фигур. Реализован алгоритм поиска объекта по заданному цвету, основанный на вычислении моментов изображения. Продемонстрирован результат работы программы, обнаруживающей объект заданного цвета в потоке видеокадров с последующим выведением его координат.

4 Разработка алгоритма детектирования движения в кадре

Следующим этапом в разработке итоговой программы является детектирование движения в кадре. Обнаружение движения — это программный алгоритм, предназначенный для обнаружения движущихся объектов в определенной области.

Существует несколько способов для определения характера движения, анализ которых позволит определиться, какой из них является наиболее подходящим для решения поставленной задачи.

4.1 Способы определения характера движения в системах видеонаблюдения

При анализе движения в общем случае используется сравнение нескольких последовательно отображаемых цифровых изображений с целью регистрации на них различных изменений. Например, определение факта перемещения (простые детекторы движения), выделение движущихся объектов и наблюдение за ними, обнаружение момента возникновения новых или исчезновения ранее существовавших объектов в кадре.

Выделение движущихся объектов может быть реализовано следующими способами:

— разность кадров,

— детектирование новых предметов,

— метод оптических потоков,

— корреляционное слежение.

Наиболее простым способом анализа движений является вычисление межкадровой разности. На рисунке 24 показан результат вычитания двух последовательных кадров. После такой операции хорошо выделяются контуры контрастирующих с фоном движущихся объектов. В левой части

рисунка показан предыдущий кадр, в центре – текущий кадр, а справа – результат вычисления межкадровой разности.

Рисунок 24 – Результат вычисления межкадровой разности

Аналогичный прием можно использовать для автоматического выделения в ходе видеонаблюдения новых объектов, ранее не участвовавших в наблюдаемом изображении. Для этого необходимо зафиксировать исходное изображение сцены, после чего каждое новое изображение сравнивать не с предыдущим, а с исходным. Если несколько кадров подряд зафиксирован новый объект, отсутствующий на исходном изображении – значит этот объект является новым и будет обнаружен.

На рисунке 25 показан результат вычитания текущего и базового исходного изображения.

Рисунок 25 – Детектирование новых объектов сцены

С практической точки зрения специфика задачи заключается в том, что одновременно в сцене наблюдения может присутствовать множество движущихся и неподвижных объектов, однако данный детектор должен выделять только те объекты, которые ранее находились в движении, а затем стабилизировали свое положение.

Как следствие, движение объектов перед камерой или движение камеры в неподвижной окружающей обстановке приводит к изменению изображения. Можно использовать эти изменения в качестве восстановления относительного движения и формы объекта.

Оптическим потоком называется характер видимого движения объектов изображения между двумя последовательными кадрами, вызванный движением объекта или камеры. Это двумерное векторное поле, где каждый вектор представляет собой вектор смещения, показывающий перемещение точек от первого кадра ко второму.

Оптический поток работает на нескольких предположениях:

— интенсивность пикселей объекта не меняется между последовательными кадрами;

— соседние пиксели имеют аналогичное движение.

На самом деле оптический поток является полезным понятием даже при условии деформации наблюдаемых поверхностей, а в частном случае движения твердого тела спектральный поток строго определен.

При определении поля передвижения каждой точке изображения необходимо указать вектор скорости. На данный момент точка Pi на изображении соответствует точке P0 на поверхности объекта. В этих двух точках они связаны уравнениями проектирования. Из центра оптической системы луч, проходящий через центр оптической системы, продолжается до пересечения с непрозрачной поверхностью (рисунок 26).

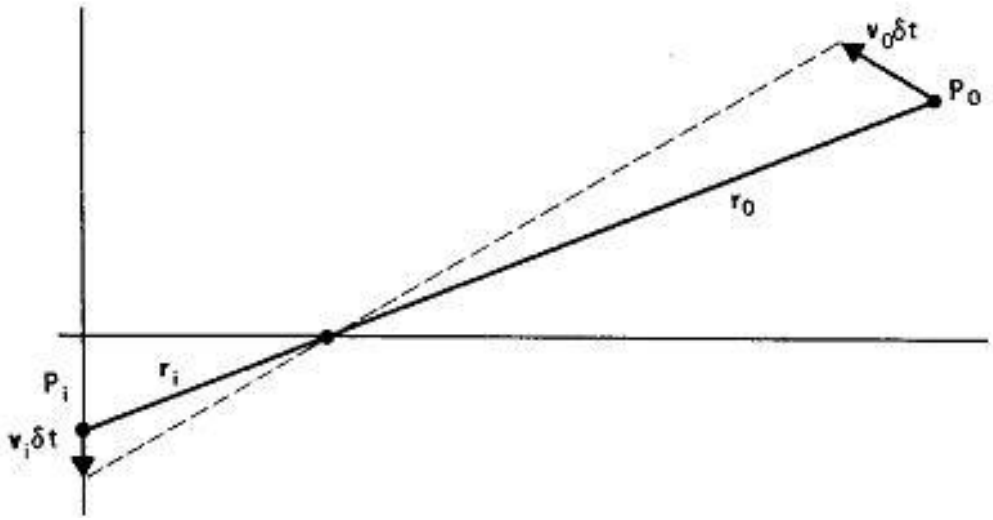

Рисунок 26 – Перемещение точки внешней среды, вызывающее перемещение соответствующей точки изображения

В точке объекта P0 камера движется со скоростью v0. В этом случае движение vi соответствует точке изображения Pi. В течение δt точка P0 переместится на расстояние $v_0\delta t$, а ее изображение P_i - на расстояние $v_i\delta t$.

Существует несколько методов вычисления оптического потока, которые можно объединить в несколько общих подходов:

— дифференциальный подход,

— корреляционный подход,

— частотный подход.

Дифференциальный подход основан на нахождении скоростей точек изображения в разностной схеме. Первым методом была вычисления производных первого порядка.

Однако дифференциальный подход к определению оптического потока может быть непрактичен, если на изображениях присутствуют шумы или они слишком малы для последовательности. Для этого были предложены корреляционные алгоритмы, которые используют наилучшее смещение между областями в последовательности изображений.

В большинстве случаев они основываются на максимизации функции сравнения (Sum-of-Squares Difference) или на минимизации SSD-функционала.

Частотный (или мощностный) подход основан на подсчете значения выходной мощности фильтров по скорости, базирующихся в основном на фильтрах Фурье.

При использовании данного подхода при фильтрах определенного вида результаты, эквивалентны результатам в дифференциальном и корреляционном подходах.

При корреляционном слежении за объектами следят путем сравнения изображения объекта, полученного с одного из предыдущих кадров видеопоследовательности (или некоторого базового «шаблона» объекта), с последующими изображениями видеопоследовательности (рисунок 27).

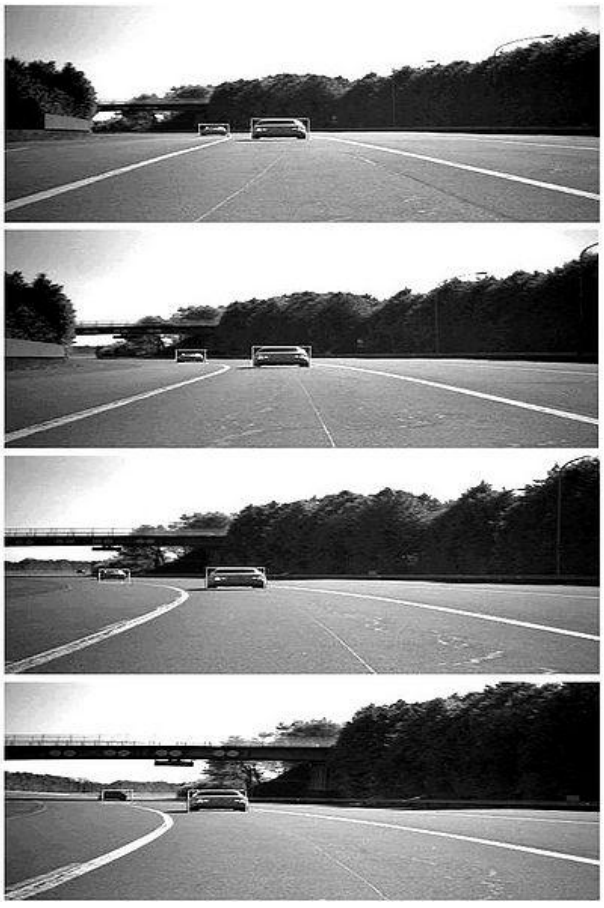

Рисунок 27 – Пример корреляционного слежения

Считается, что максимум корреляционной функции указывает на место прослеживаемого объекта в новом кадре [1].

4.2 Детектирование движения в кадре

Перед началом написания программы, следует оформить блок-схему основного алгоритма, детектирующего движение (рисунок 28).

Рисунок 28 – Блок-схема алгоритма детектирования движения в кадре

На вход поступают кадры, представляющие собой две последовательности байт в формате RGB. Далее происходит попиксельное вычисление межкадровых разностей. Для каждого пикселя изображения происходит вычисление среднего значения между тремя компонентами цвета. Среднее значение сравнивается с заданным порогом и в результате данного сравнения формируется двоичная маска. Полученное изображение преобразуется в черно-белое, производится удаление шумов и сглаживание. При наличии движения, происходит обведение в контур участка на изображении, где была вычислена межкадровая разность.

После оформления блок-схемы можно приступать к написанию программы. Сначала необходимо импортировать библиотеки для работы с компьютерным зрением и массивами.

```
import numpy as n
import cv2
```

Затем, необходимо произвести захват с веб-камеры и сохранить в переменные ширину и высоту экрана.

```
video = cv2.VideoCapture(0,cv2.CAP_DSHOW)
width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
height=int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

Последовательно считываются и сохраняются в переменные два кадра.

```
ret,frame1 = cap.read()
ret,frame2 = cap.read()
```

После этого с помощью оператора while проверяется, включена ли камера. Пока она включена, производится вычисление межкадровой разности с помощью функции absdiff:

```
diff = cv2.absdiff(frame1, frame2)
```

Затем изображение преобразуется в черно-белое с помощью функции cv2.COLOR_BGR2GRAY:

```
gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
```

После этого полученное изображение необходимо сгладить, убирать шумы и найти контуры интересующей движущейся области:

blur = cv2.GaussianBlur(gray, (5,5), 0)

thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)

dilated = cv2.dilate(thresh, None, iterations=3)

contours=cv2.findContours(dilated,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

Далее отрисовываются контуры объекта, который движется:

cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)

Полученное изображение сохраняется и выводится на экран. При отсутствии движения в кадре будет выведено лишь текущее изображение (рисунок 29).

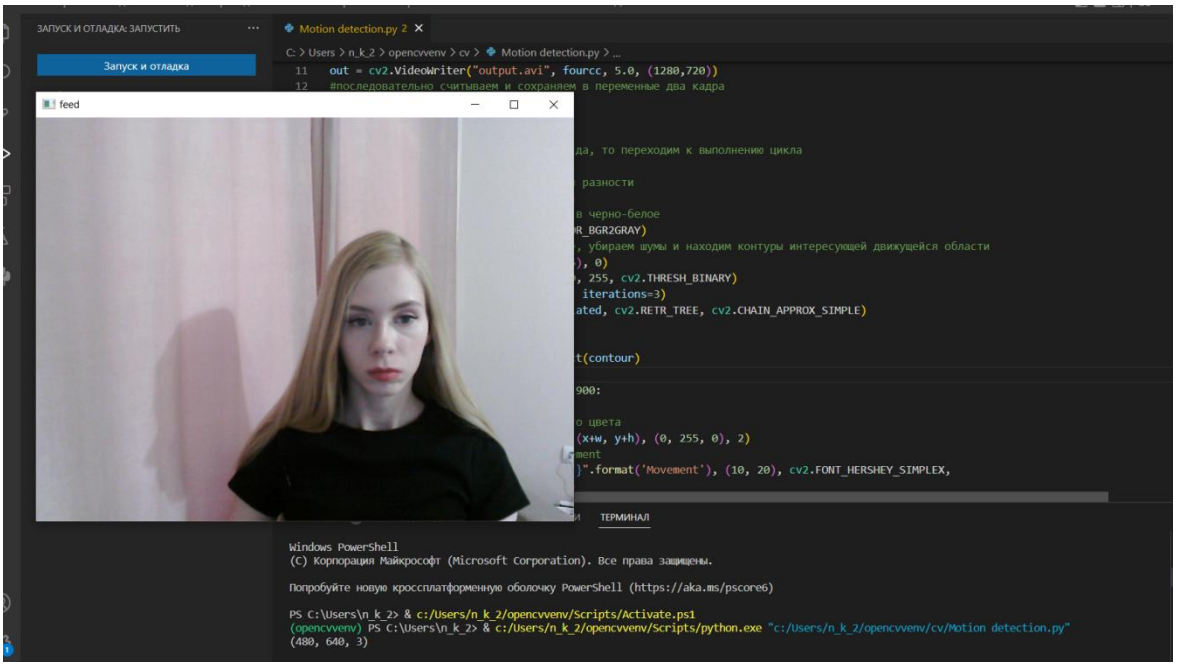

Рисунок 29 – Отсутствие движения в кадре

Однако как только программа обнаружит движение, этот предмет будет обведен в контур и появится текст, сообщающий о наличии движения в кадре (рисунок 30).

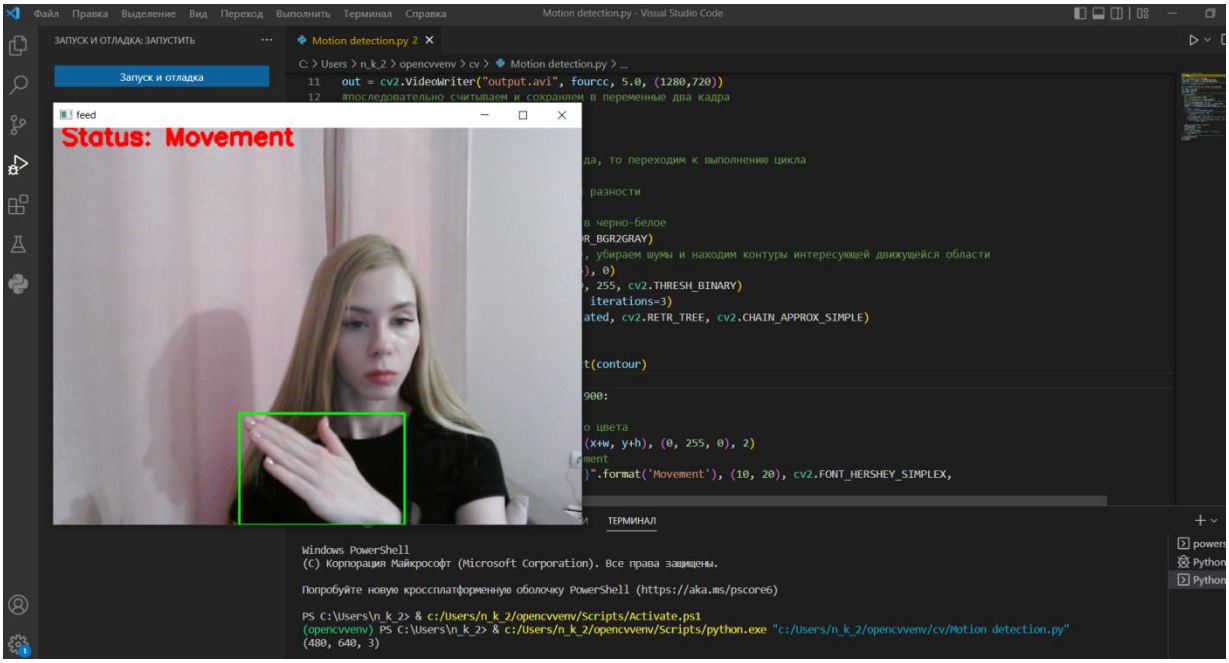

Рисунок 30 – Присутствие движения в кадре

4.3 Реализация вывода координат движущегося объекта

По аналогии с программой, описанной в третьем разделе, можно вычислить средние координаты движущегося объекта путем вычисления его моментов:

```
moments = cv2.moments(thresh, 1)
    m01 = moments['m01']
    m10 = moments['m10']
    Area = moments['m00']
    if dArea > 100:
        for contour in contours:
            (x, y, w, h) = cv2.boundingRect(contour)
            if cv2.contourArea(contour) < 3000:
                continue
            x = int(m10 / Area)
```

y = int(m01 / Area)

Результат работы программы продемонстрирован на рисунке 31.

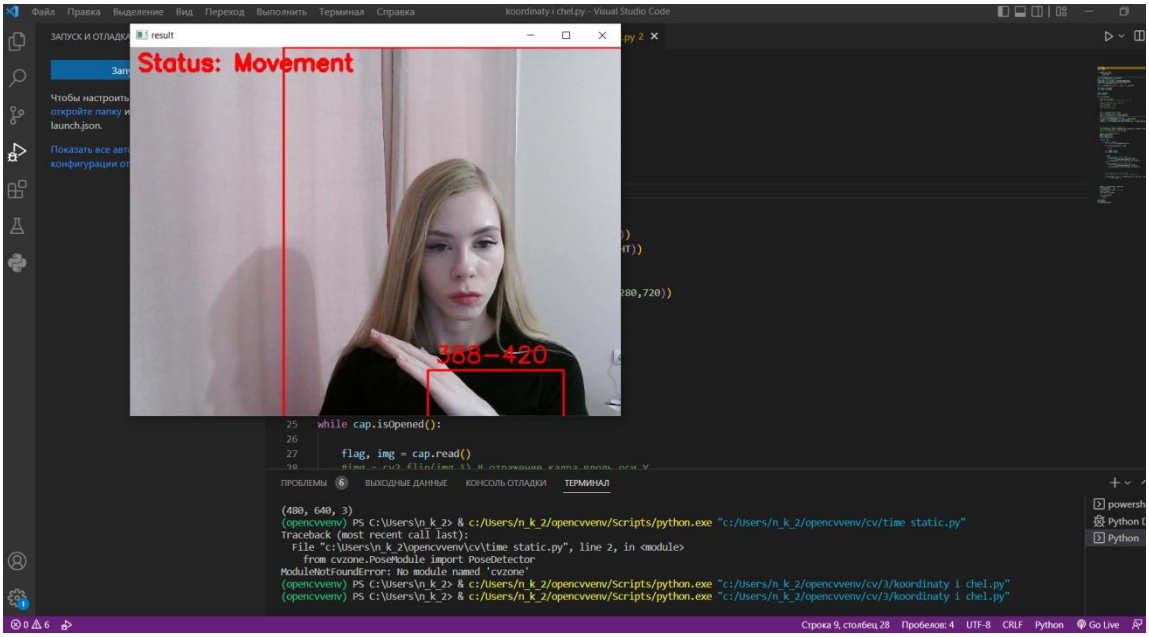

Рисунок 31 – Детектирование движения и координат объекта

Таким образом, в разделе были проанализированы основные методы детектирования новых объектов на изображении. Разработан алгоритм детектирования движения в кадре, в основе которого лежит метод межкадровой разности. Используя данный метод, библиотеку OpenCV и язык программирования Python, была реализована программа, считывающая любое движение в кадре, а также фиксирующая координаты найденного объекта.

5 Разработка алгоритма распознавания человеческой фигуры

После успешного определения движения в кадре, необходимо сузить круг детектируемых объектов, а именно — реализовать алгоритм, который будет считывать только движения фигуры человека.

5.1 Анализ алгоритмов распознавания человеческих фигур

На сегодняшний день существует ряд алгоритмов, позволяющих распознать фигуру человека. В библиотеке OpenCV самыми популярными и эффективными считаются:

— метод гистограммы ориентированных градиентов,

— каскадный классификатор,

— mediapipe.

В OpenCV реализован быстрый метод обнаружения человека, называемый HOG (гистограммы ориентированных градиентов). При использовании данного метода изображение предварительно обрабатывается, оно может быть любого размера, но единственное ограничение заключается в том, что анализируемые области имеют фиксированное соотношение сторон.

Далее необходимо рассчитать градиентные изображения. Чтобы вычислить дескриптор HOG, нужно сначала вычислить горизонтальный и вертикальный градиенты для вычисления гистограммы градиентов. В OpenCV расчет можно выполнить с помощью функции cartToPolar().

В градиентной картинке убрали множество незначащей информации (например, постоянное цветное изображение), но выделили контуры. Так что можно смотреть на градиентное изображение и все равно легко сказать, кто изображен в нём.

В каждой ячейке градиент вычисляется для каждого пикселя, и градиенты используются для заполнения гистограммы: значение - это угол градиента, а вес - величина градиента.

Гистограммы всех ячеек объединяются и передаются на дескриптор машинного обучения, чтобы решить, соответствуют ли ячейки текущего окна обнаружения человеку или нет.

Градиент в каждом пикселе имеет величину и направление. Для цветных изображений оценивается градиент трех. Площадь градиента в пикселе - это максимум величины градиентов трех каналов, а угол - это угловой коэффициент максимальной градации [16].

Поиск объектов с использованием каскадных классификаторов на основе признаков Хаара - это эффективный метод обнаружения объекта. Пол Виола и Майкл Джонс опубликовали статью «Быстрое обнаружение предметов с помощью усиленного каскада простых признаков» в 2001 году. Этот подход основывается на машинном обучении, когда каскадная функция учится на множестве положительных и отрицательных изображений. Потом для обнаружения объектов на изображениях его уже можно будет использовать.

По мере обучения классификатора обнаружению лиц, сначала нужно использовать множество положительных (изображений лиц) и отрицательных изображений (изображений без лиц). Затем следует извлечь из него признаки. В этом случае используется функция Хаара, которая показана на рисунке 32.

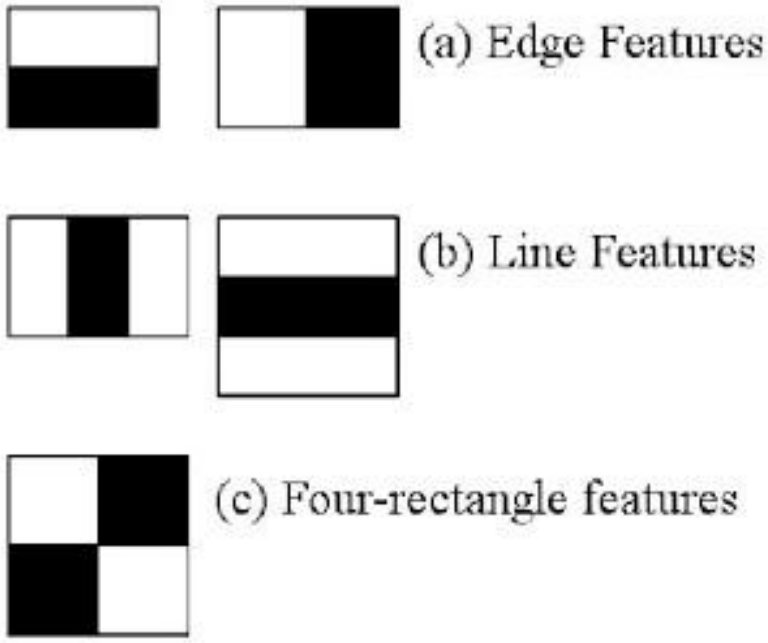

Рисунок 32 – Функция Хаара

Все функции представляют собой отдельные значения, полученные при сложении суммы пикселей под белым прямоугольником и суммы пикселей под черным. Пример использования функции Хаара на изображении лица человека представлен на рисунке 33.

Рисунок 33 – Применение функции Хаара

Далее полученный размер и расположение ядра используется для расчета множества функций. Расчет происходит путем нахождения суммы пикселей под черными и белыми прямоугольниками. Для упрощения этой задачи, как правило, вводятся интегральные изображения.

Каждая функция применяется ко всем тренировочным изображениям. Для каждой функции классификатор находит лучший порог, который классифицирует лица на положительные и отрицательные. Но, очевидно, будут ошибки или неправильная классификация, поэтому каждому изображению вначале присваивается одинаковый вес. После каждой классификации веса неправильно классифицированных изображений увеличиваются. Процесс повторяется несколько раз и рассчитываются новые коэффициенты ошибок. Процесс завершится, как только будет достигнута требуемая точность или частота ошибок, или будет найдено необходимое количество признаков.

Окончательный классификатор представляет собой взвешенную сумму этих слабых классификаторов. Он называется слабым потому, что сам по себе не может классифицировать изображение, но вместе с другими образует сильный классификатор. В документе говорится, что даже 200 функций обеспечивают обнаружение с точностью 95%. В их окончательной настройке было около 6000 функций.

В OpenCV данный классификатор представлен с тренером, а также с самим детектором. На данный момент библиотека содержит множество предварительно обученных классификаторов для распознавания лица, глаз, улыбки и т. д. Эти файлы хранятся в формате XML [24].

Сначала необходимо загрузить необходимые классификаторы XML. Затем загрузить входное изображение (или видео) в оттенках серого [4].

```
import numpy as np
import cv2 as c
```

Теперь объявляем переменные, в которых будут храниться каскады Хаара для определения лица и глаз.

```
face_cas = c.CascadeClassifier('frontalface.xml')
eye_cas = c.CascadeClassifier('haarcascade_eye.xml')
```

Считываем изображение из папки и сохраняем в переменную image. Затем превращаем ее в черно-белый формат.

```
image = c.imread('image123.jpg')
gray = c.cvtColor(image, c.COLOR_BGR2GRAY)
```

Далее идет поиск лица на изображении. Если лица найдены, то возвращается позиция обнаруженного лица в виде Rect(x,y,w,h). Как только эта область будет найдена, сразу же к ней применяется обнаружение глаз.

```
faces = face_cas.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
    image = c.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = image[y:y+h, x:x+w]
    eyes = eye_cas.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        c.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
```

Выводим получившееся изображение с нарисованным квадратом поверх найденного лица на экран. Ожидаем нажатия любой кнопки клавиатуры для закрытия приложения.

```
c.imshow('img', image)
c.waitKey(0)
c.destroyAllWindows()
```

Данный алгоритм наиболее эффективен для обнаружения фронтальных изображений лица и не подходит под задачу данной работы – обнаружению человеческой фигуры. Также каскады Хаара склонны к ложным срабатываниям — алгоритм Виолы-Джонса может легко сообщить о лице на изображении, когда лица нет.

Mediapipe уже довольно активно, а главное, эффективно используется для детектирования многочисленных лиц на фото, а класс PoseDetector

способен оценивать точки тела человека. Изображение RGB представляется в виде массива в математическом модуле Numpy. Оно обрабатывается и возвращает ориентиры точек наиболее заметного обнаруженного человека [19].

Модель ориентиров в MediaPipe Pose прогнозирует расположение 33 ориентиров позы (рисунок 34).

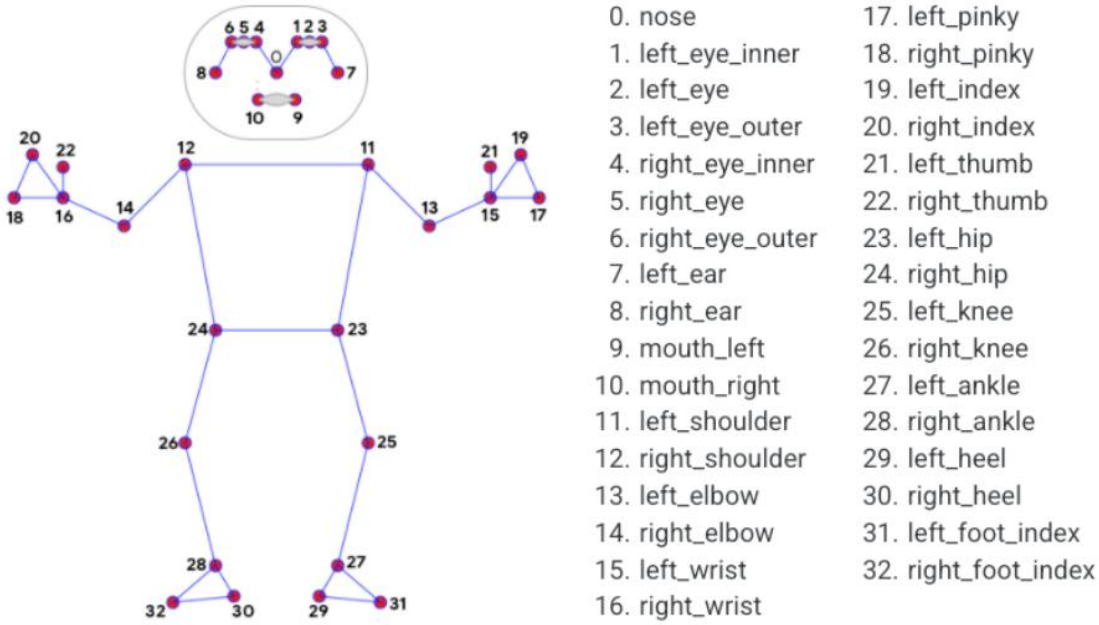

Рисунок 34 – Модель ориентиров в MediaPipe Pose

При инициализации класса Pose необходимо указать некоторые параметры, указанные ниже.

В параметре «STATIC_IMAGE_MODE» устанавливается тип входных данных. Если указано значение false, программа обрабатывает входные изображения как видеопоток. Она попытается обнаружить самого заметного человека на самых первых изображениях, а при успешном обнаружении дополнительно локализует ориентиры позы. Затем на последующих изображениях он просто отслеживает эти ориентиры, не вызывая другого обнаружения, пока не потеряет отслеживание, уменьшая вычисления и

задержку. Если установлено значение true, обнаружение людей запускает каждое входное изображение, что идеально подходит для обработки пакета статических, возможно, несвязанных изображений.

Параметр «MODEL_COMPLEXITY» отвечает за сложность модели ориентира позы: 0, 1 или 2. Точность ориентира, а также задержка вывода обычно увеличиваются со сложностью модели.

При установленном значении true параметра «SMOOTH_LANDMARKS», подключаются фильтры сглаживания, чтобы уменьшить дрожание, но игнорируются, если для static_image_mode также установлено значение true.

Параметр «ENABLE_SEGMENTATION» при установленном значении true, в дополнение к ориентирам позы также генерирует маску сегментации. По умолчанию false.

При установленном значении true параметра «SMOOTH_SEGMENTATION», подключаются фильтры маски сегментации для разных входных изображений в целях уменьшения дрожания. Игнорируется, если значение «enable_segmentation» равно false или «static_image_mode» равно true. По умолчанию true.

Параметр «MIN_DETECTION_CONFIDENCE» отвечает за минимальное значение достоверности ([0.0, 1.0]) из модели обнаружения человека, при котором обнаружение считается успешным. По умолчанию 0.5.

Параметр «MIN_TRACKING_CONFIDENCE» — минимальное значение достоверности ([0.0, 1.0]), которое считается успешно отслеженным, иначе обнаружение человека будет автоматически вызываться на следующем входном изображении. Установка более высокого значения может повысить надежность решения за счет более высокой задержки. Игнорируется, если «static_image_mode» имеет значение true, где обнаружение человека запускается на каждом изображении. По умолчанию 0.5 [18].

Для решения задачи данной работы можно воспользоваться каскадом Хаара, обученном на распознавание лиц. Однако при детектировании

движения человека в помещении недостаточно распознавать только лицо, а каскад Хаара, обученный на всю фигуру человека работает довольно медленно [2]. Поэтому было принято решение воспользоваться подходом к распознаванию людей, описанном в модуле cvzone. Это пакет компьютерного зрения, который позволяет легко запускать функции обработки изображений и искусственного интеллекта. В своей основе он использует библиотеки OpenCV и Mediapipe.

После выставления вышеперечисленных параметров, можно приступать к написанию итоговой программы.

5.2 Детектирование движения фигуры человека

За основу берется программа, которая детектирует движение в кадре за счет вычисления межкадровой разности.

В первую очередь необходимо импортировать класс PoseDetector из модуля cvzone.PoseModule:

```
from cvzone.PoseModule import PoseDetector
```

Далее следует объявить объект, принадлежащий классу PoseDetector:

```
det = PoseDetector()
```

После этого необходимо поместить в основной цикл программы переменную, которая будет отвечать за поиск человека на изображении и принимать координаты найденного человека:

```
frame = det.findPose(frame)
lmlist, bboxInfo = det.findPosition(frame)
```

Далее необходимо прописать проверку того, что фигура находится в пределах экрана и поставить ограничение на поиск движения именно в пределах данной области:

```
    if len(bboxInfo) != 0:
        if bboxInfo['bbox'][0] > 0 and bboxInfo['bbox'][0] < bboxInfo['bbox'][2]:
```

frame1 = frame1[bboxInfo['bbox'][1]:bboxInfo['bbox'][3], bboxInfo['bbox'][0]:bboxInfo['bbox'][2]]

frame2 = frame2[bboxInfo['bbox'][1]:bboxInfo['bbox'][3], bboxInfo['bbox'][0]:bboxInfo['bbox'][2]]

Результат выполнения программы представен на рисунке 35.

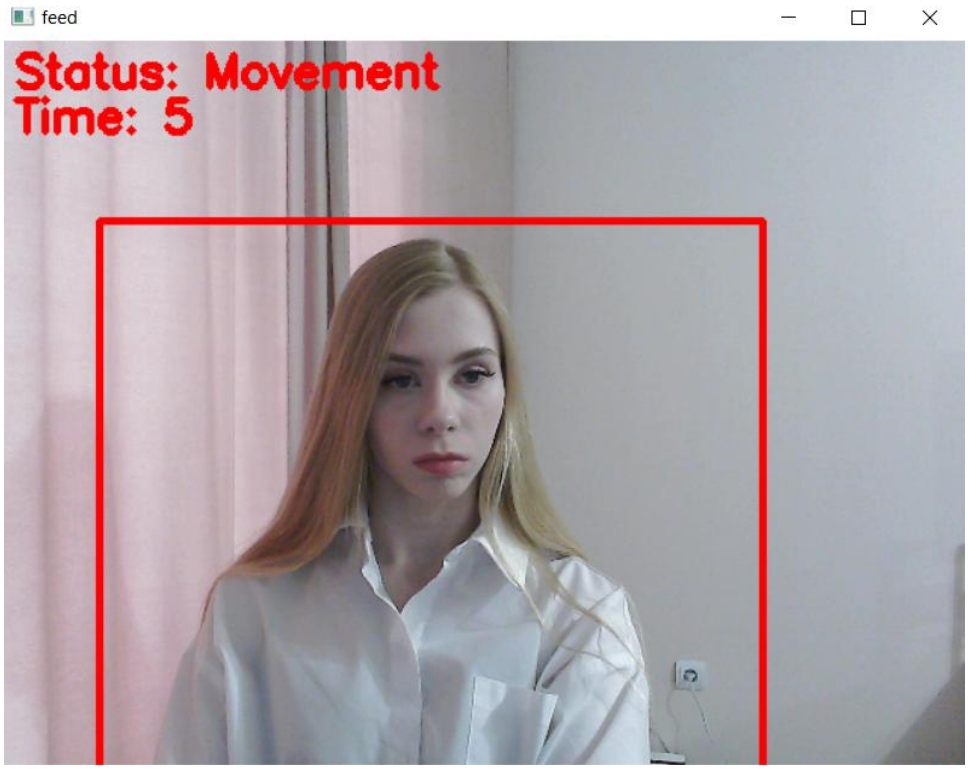

Рисунок 35 – Результат детектирования движения человека

Таким образом, в разделе были проанализированы современные алгоритмы распознавания человеческой фигуры - метод гистограммы ориентированных градиентов, каскадный классификатор и mediapipe. Разработана и протестирована программа для поиска фигуры человека в кадре, использующая модель ориентиров.

6 Фиксация интервала времени и экспорт данных

Последним этапом в разработке итоговой программы является фиксация интервала времени, в течение которого человек в кадре был неподвижен. Также предложен вариант экспорта данных в текстовый документ.

6.1 Фиксация интервала времени неподвижности объекта

Одной из самых важных частей описываемой работы является фиксация времени, в течение которого объект неподвижен. За основу взята программа, определяющая движение объекта в кадре.

В программе производится расчет времени с использованием вспомогательных переменных и функций модуля time, встроенных в язык Python. Метод основывается на фиксации последнего момента времени, когда объект перестал двигаться.

Программа для фиксации интервала времени, в течение которого объект неподвижен, будет включать в себя все пункты, описанные ранее, а именно: расчет моментов, вычисление межкадровой разности и вывод контура движущегося объекта.

Для решения задачи в первую очередь необходимо создать новые переменные, в которые будет записываться время и значения функций встроенного модуля time [6]:

```
timer=0
start_time=time.time()
start_time_show=time.time()
time_to_show=0
```

В цикле программы начинается детектирование движения, а также добавлено условие, по которому можно засечь и сохранить в переменную последний момент времени, когда объект не двигался:

```
if cv2.contourArea(contour) < 900:
        timer=time.time() #последний момент времени, когда объект не двигался
        #не рисуем, пропускаем
        continue
```

Задаем еще одно условие - если время, пройденное с последнего детектирования движения в кадре больше 1 секунды, и в данный момент объект снова двигается, тогда обнуляем это время и выводим на исходное изображение время, в течение которого объект был неподвижен:

```
    if time.time() - start_time_show  > 1:
        start_time_show = time.time()
        time_to_show=timer-start_time
      cv2.putText(frame1, f"Time: {int(time_to_show)}", (100, 200), cv2.FONT_HERSHEY_SIMPLEX,
                  1, (0, 0, 255), 3)
```

После этого программа снова переходит в фазу поиска.

6.2 Экспорт данных в текстовый документ

В ходе работы программы мы получаем некоторые значения, которые необходимо фиксировать в отдельном текстовом файле.

В Python есть встроенная функция open(). С ее помощью можно открыть любой файл на компьютере. Технически Python создает на его основе объект:

```
f = open(file_name, access_mode)
```

В параметр file_name записывается имя открываемого файла, а параметр access_mode отвечает за режим открытия файла. По умолчанию используется режим чтения (r), если другое не указано [7].

В данном случае необходимо использовать режим appending - он позволяет добавлять в файл новую информацию, не удаляя при этом предыдущие записи [28].

Воспользоваться данным приемом необходимо в той части программы, где считывается время, за которое фигура человека не двигалась:

```
if time.time() - start_time_show  > 1:
    start_time_show = time.time()
time_to_show=timer-start_time
with open('test.txt', 'a') as f:
    f.write(str(int(time_to_show)) + '\n')
```

Результат выполнения программы представлен на рисунке 36.

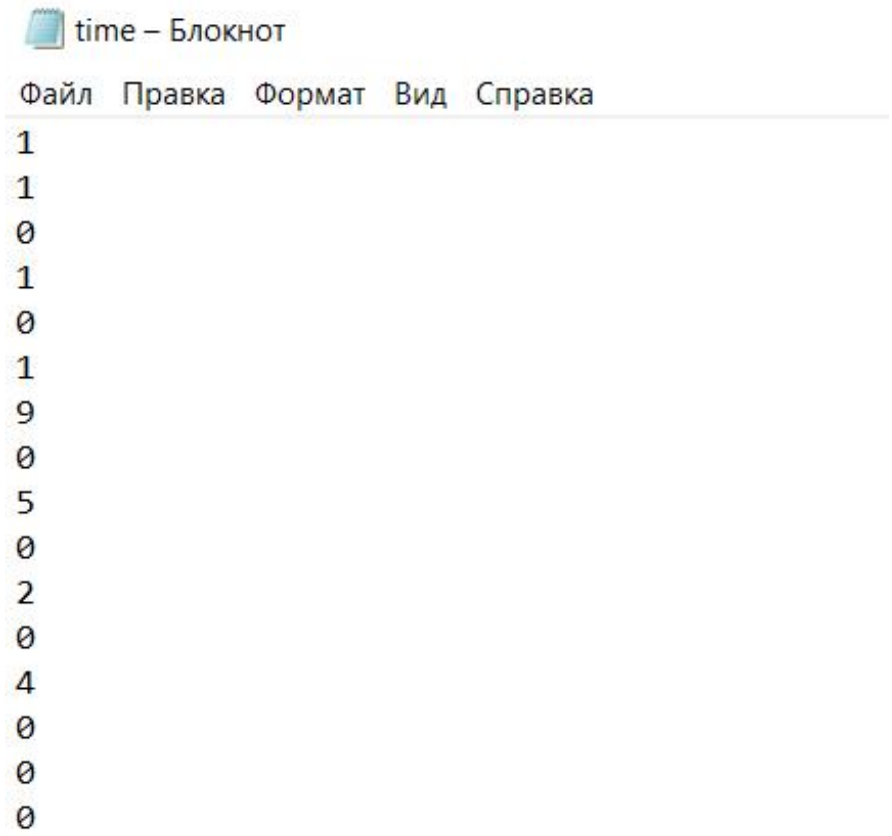

Рисунок 36 – Результат экспорта данных

6.3 Блок-схема итоговой программы

Итоговая программа включает в себя следующие ключевые элементы:

— ограничение поиска движения по силуэту человека,

— вычисление межкадровой разности для детектирования движения человека в кадре.

Блок-схема алгоритма, осуществляющего ограничение поиска движения по силуэту человека, представлена на рисунке 37.

Рисунок 37 – Блок-схема алгоритма ограничения поиска движения по силуэту человека

На данном этапе программа использует метод ориентиров позы человека для его обнаружения в кадре, после чего происходит ограничение поиска движения строго в границах найденной фигуры.

Блок-схема алгоритма вычисления межкадровой разности для детектирования движения человека в кадре представлена на рисунке 38.

Рисунок 38 – Блок-схема алгоритма вычисления межкадровой разности

На данном этапе происходит детектирование движения путем вычисления межкадровой разности с последующей фиксацией последнего момента времени, когда фигура человека находилась в движении. При результате, отличным от нуля, он выводится поверх текущего изображения и экспортируется в текстовый документ.

Блок-схема итоговой программы представлена на рисунке 39.

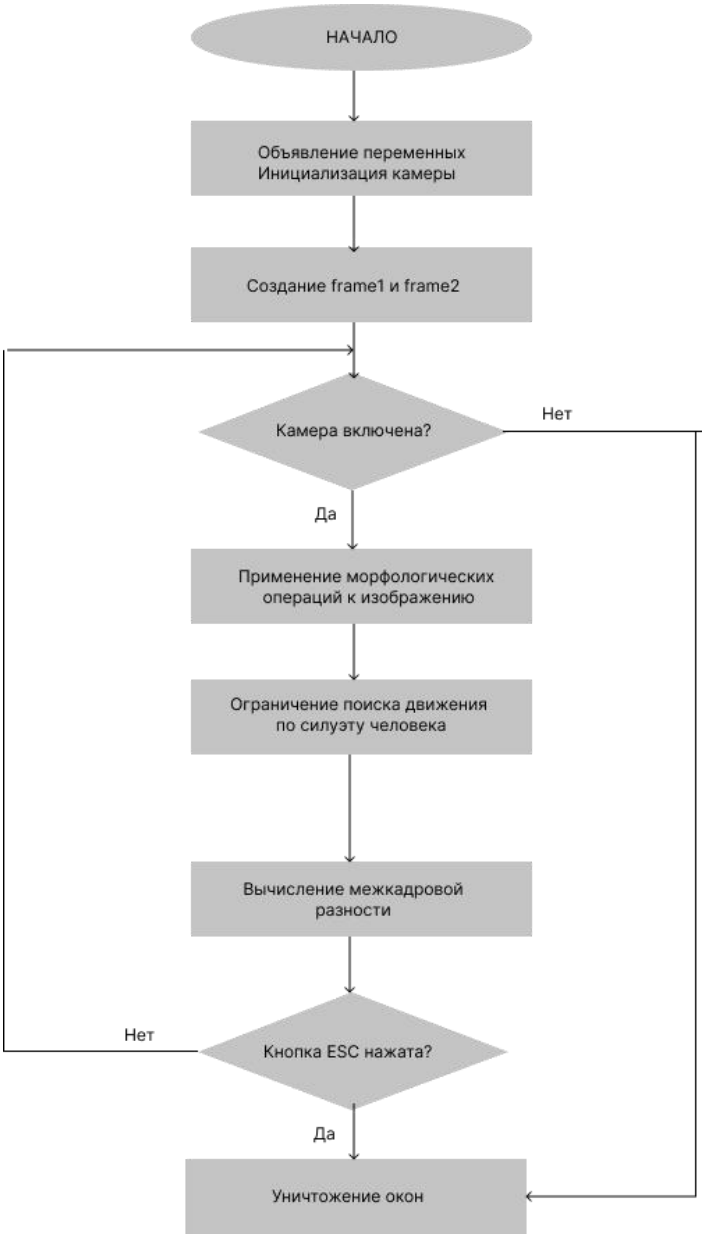

Рисунок 39 – Блок-схема итоговой программы

Таким образом, в данном разделе был разработан алгоритм, позволяющий рассчитать и зафиксировать время, за которое объект оставался неподвижен. С помощью встроенных функций в Python, нацеленных на работу с файлами, осуществлен экспорт значений времени в текстовый документ. Представлена блок-схема итоговой программы.

Заключение

В рамках выполнения выпускной квалификационной работы была создана система распознавания фигуры человека средствами компьютерного зрения. Программа внедрена в одноплатный компьютер Raspberry Pi 2 Model B. Готовый проект представляет собой систему, которая распознает фигуру человека в офисном помещении, а также фиксирует и экспортирует в текстовый документ интервал времени, в течение которого фигура была неподвижна.

В первом разделе проанализирована структура, характеристики и область применения встраиваемых систем.

Второй раздел посвящен выбору инструментов для выполнения выпускной квалификационной работы. Проанализирован рынок одноплатных компьютеров и рассмотрены современные библиотеки компьютерного зрения. Обоснован выбор каждого элемента системы.

В третьем разделе представлена структурная схема разрабатываемого устройства и перечень всех используемых элементов. Рассмотрены все этапы настройки ПО Raspberry Pi, необходимые для дальнейшей работы с системой. Приведены основы работы с библиотекой компьютерного зрения OpenCV при использовании языка программирования Python, в том числе определение простейших геометрических фигур. Реализован алгоритм поиска объекта по заданному цвету, основанный на вычислении моментов изображения. Продемонстрирован результат работы программы, обнаруживающей объект заданного цвета в потоке видеокадров с последующим выведением его координат.

В четвертом разделе были проанализированы основные методы обнаружения нового объекта в потоке видеокадров. Разработан алгоритм детектирования движения в кадре, в основе которого лежит метод межкадровой разности.

В пятом разделе проанализированы современные алгоритмы распознавания человеческой фигуры. Разработана программа, использующая модель ориентиров и прогнозирующая расположение 33 ориентиров позы человека. Приведен результат выполнения программы.

В шестом разделе описан алгоритм, позволяющий фиксировать время, в течение которого фигура человека была неподвижна, а также экспортировать данные в текстовый документ. Представлена блок-схема итоговой программы. Приведены результаты обнаружения.

Таким образом, результатом выполнения выпускной квалификационной работы является исправно работающая система распознавания фигуры человека, находящая свое применение в офисных помещениях.

Список используемой литературы

1. Анализ движения в задачах видеонаблюдения // wiki.technicalvision. URL:http://wiki.technicalvision.ru/index.php/Анализ_движения_в_задачах_видеонаблюдения (дата обращения: 26.11.2021)

2. В.Т. Фисенко, Т.Ю. Фисенко. Компьютерная обработка и распознавание изображений: учеб. пособие. - СПб: СПбГУ ИТМО, 2008. – 192 с.

3. ГОСТ 12.0.003-2015. Межгосударственный стандарт. Система стандартов безопасности труда. Опасные и вредные производственные факторы. Классификация. Введ. 2017-03-01

4. Коэльё Л. П., Ричерт В. Построение систем машинного обучения на языке Python. — Перевод с английского. — М.: ДМК Пресс, 2015. — с. — ISBN 978-5-9706-0330-7.

5. Л. Шапиро, Дж. Стокман. Компьютерное зрение = Computer Vision. — М.: Бином. Лаборатория знаний, 2006. — 752 с. — ISBN 5-94774-384-1.

6. Модуль time в Python // pythonru. URL: https://pythonru.com/osnovy/modul-time-v-python (дата обращения: 15.02.2022)

7. Основы работы с файлами в Python // Tproger. URL: https://tproger.ru/articles/files-in-python/ (дата обращения: 15.02.2022)

8. Петин В. А. Микрокомпьютеры Raspberry Pi. Практическое руководство. — СПб.: БХВ-Петербург, 2015. — 240 с.: ил. — (Электроника) ISBN 978-5-9775-3519-9

9. Распиновка разъемов GPIO, DSI, CSI, 3.5 аудио/видео, I2S, тестовых точек в RaspberryPi // pcminipro. URL: https://pcminipro.ru/stati/raspinovka-razemov-gpio-dsi-csi-3-5-audio-video-i2s-testovyh-tochek-v-raspberrypi/ (дата обращения: 10.04.2021).

10. Что такое TensorFlow? // ru.realiventblog. URL: https://ru.realiventblog.com/27-what-is-tensorflow-the-machine-learning-library-explained (дата обращения: 04.04.2021).

11. Шахин Гадир Сравнительный анализ библиотек компьютерного зрения // Colloquium-journal. 2019. №24 (48). URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-bibliotek-kompyuternogo-zreniya (дата обращения: 10.04.2021).

12. 20 Best Raspberry Pi Alternatives // ubuntupit. URL: https://www.ubuntupit.com/best-raspberry-pi-alternatives/ (дата обращения: 27.11.2020)

13. Beginners Guide to Installing Raspberry Pi OS on a Raspberry Pi // medium. URL: https://medium.com/@mkmety/beginners-guide-to-installing-raspberry-pi-os-on-a-raspberry-pi-6d437ce2f54b (дата обращения: 09.10.2021)

14. embedded system // TechTarget URL: https://www.techtarget.com/iotagenda/definition/embedded-system (дата обращения: 01.12.2020)

15. Feature Detection // OpenCV 2.4.13.7 documentation. URL: https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=houghcircles (дата обращения: 15.10.2021)

16. Histogram of Oriented Gradients explained using OpenCV // learnopencv. URL: https://learnopencv.com/histogram-of-oriented-gradients/ (дата обращения: 03.12.2021)

17. How Does Video Surveillance System Monitoring Work? // centralizedvision. URL: https://centralizedvision.com/2016/06/16/video-surveillance-system-monitoring/ (дата обращения: 21.03.2021)

18. Mediapipe 0.8.10 // pypi. URL: https://pypi.org/project/mediapipe/ (дата обращения: 07.12.2021)

19. MediaPipe Pose // github. URL: https://google.github.io/mediapipe/solutions/pose#enable_segmentation (дата обращения: 03.12.2021)

20. Morphological Transformations of Images using OpenCV | Image Processing Part-2 // Analitics Vidhya 2020 URL: https://medium.com/analytics-vidhya/morphological-transformations-of-images-using-opencv-image-processing-part-2-f64b14af2a38 (дата обращения: 07.09.2021).

21. OpenCV // opencv. URL: https://opencv.org/about/ (дата обращения: 04.04.2021)

22. OpenCV или MatLAB: Что лучше для компьютерного зрения? // kverner. URL: https://www.kverner.ru/opencv-ili-matlab-chto-luchshe-dlya-kompyuternogo-zreniya/optncv (дата обращения: 07.04.2021)

23. OpenCV шаг за шагом. Сравнение контуров через суммарные характеристики — моменты // robocraft. URL: https://robocraft.ru/computervision/867 (дата обращения: 26.11.2021)

24. OpenCV Haar Cascades // pyimagesearch. URL: https://pyimagesearch.com/2021/04/12/opencv-haar-cascades/ (дата обращения: 03.12.2021)

25. OpenCV, Машинное зрение на Python: Прямоугольные объекты. Часть 3 // линуксблог. URL: https://линуксблог.рф/opencv-mashinnoe-zrenie-na-python-pryamougolnye-obekty-chast-3/ (дата обращения: 13.09.2021)

26. Raspberry Pi 2 Model B Review // pcmag. URL: https://www.pcmag.com/reviews/raspberry-pi-2-model-b (дата обращения: 27.11.2020)

27. Raspberry Pi Camera Module Review and Tutorial Guide [Электронный ресурс]. URL: https://www.tweaktown.com/guides/5617/raspberry-pi-camera-module-review-and-tutorial-guide/index.html (дата обращения: 13.09.2021)

28. Reading and Writing Files in Python (Guide) // realpython. URL: https://realpython.com/read-write-files-python/ (дата обращения: 18.02.2022)

29. SINGLE BOARD COMPUTERS FOR EMBEDDED SYSTEMS // coghlincompanies URL: https://www.coghlincompanies.com/single-board-computers-for-embedded-systems/ (дата обращения: 17.05.2021)

30. The 12 Most Popular Computer Vision Tools in 2022 // vison URL: https://viso.ai/computer-vision/the-most-popular-computer-vision-tools/ (дата обращения: 04.04.2021)