ЯФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»

(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ПРОГНОЗИРОВАНИЯ ПОСЕЩАЕМОСТИ МАГАЗИНА

Выпускная квалификационная работа

обучающегося по направлению подготовки

02.03.03 Математическое обеспечение и администрирование информационных систем

очной формы обучения,

группы 07001402

Карпенко Эдуарда Борисовича

Научный руководитель

к.т.н., Великая Я.Г.

БЕЛГОРОД 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 3
ГЛАВА 1. ЗАДАЧИ И МЕТОДЫ ВИДЕОАНАЛИТИКИ 5
1.1 Задачи, решаемые с помощью видеоаналитики 5
1.2 Алгоритмы сжатия, применяемые к видео 13
ГЛАВА 2. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ПОИСКА И ПРОЦЕССОВ РАСПОЗНАВАНИЯ ОБЪЕКТОВ НА ВИДЕО 16
2.1 Обнаружение объектов методом вычитания фона 16
2.2 Постобработка переднего плана изображения 18
2.3 Сегментация и извлечение объекта 19
2.4 Особенности объектов 21
2.5 Классификация сегментированных объектов 23
ГЛАВА 3. РАЗРАБОТКА СИСТЕМЫ ПРОГНОЗИРОВАНИЯ ПОСЕЩАЕМОСТИ МАГАЗИНА 26
3.1 Предварительная обработка потока видео для задач определения параметров движения объектов 26
3.2 Гистограмма ориентированных градиентов (HOG) 31
3.3 Структура автоматизированной системы 40
3.4 Тестирование приложения 43
ЗАКЛЮЧЕНИЕ 47
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 48
ПРИЛОЖЕНИЕ 50

ВВЕДЕНИЕ

В современном мире задача трекинга объектов на видео является неотъемлемой частью большинства прикладных областей, таких как построение систем видеонаблюдения, отслеживания транспортного и людского трафиков, автоматического контроля. Каждая из этих задач имеет свою особенность, свои требования к надежности распознавания, к дальности действия и многому другому. Даже такой фактор, как желание самого объекта (в данном случае человека), быть распознанным – может значительно повлиять на работу того или иного алгоритма распознавания.

В связи с этим, современная жизнь диктует растущие требования для все более новых и совершенных методов локализации и распознавания людей. Интерес к ним довольно значителен, в виду их широкого практического применения в таких областях, как охранные системы, системы обеспечения безопасности в местах массового пребывания людей, антитеррористические системы и т. д.

Автоматический подсчет числа людей необходим не только в целях общественной безопасности. Его можно использовать для оценки числа посетителей, регулирования количества открытых касс, планирования мест установки рекламных носителей. Системы подсчета людей все чаще применяются организациями розничной торговли.

Таким образом целью данной дипломной работы является разработка автоматизированной системы прогнозирования посещаемости магазина. Соответственно, можно выделить следующие задачи, для достижения поставленной цели:

- рассмотреть существующие алгоритмы и методы поиска объектов на видео;

- выполнить предварительную обработку потока видео;

- реализовать алгоритм для реализации поиска и классификации объектов на изображении;

- осуществить автоматическую отправку данных о количестве объектов на кадре для обработки на хост;

- предоставить данные о посещаемости в удобной форме пользователю.

В первой главе пояснительной записки рассматриваются задачи видеоаналитики и способы их решения, а также алгоритмы сжатия видео, так как возникает потребность в обработке больших объемов данных.

Во второй главе представлен обзор существующих методов поиска и процесса распознавания объектов на видео, что включает в себя методы обнаружения объектов на изображениях, методы построения переднего плана и классификацию движущихся объектов.

В последней главе рассматривается непосредственно реализация приложения автоматизированной системы прогнозирования посещаемости магазина. Также, подробно рассмотрен алгоритм построения гистограммы ориентированных градиентов (HOG), на основе которого реализовано распознавание объектов на видео. Описана структура разработанного приложения.

Поставленные задачи по обработке видео будем решать на основе использования методов из библиотеки OpenCV на языке программирования Python.

Данная дипломная работа содержит 50 страниц, 26 рисунков, 11 литературных источников и 1 приложение.

ГЛАВА 1. ЗАДАЧИ И МЕТОДЫ ВИДЕОАНАЛИТИКИ

1.1 Задачи, решаемые с помощью видеоаналитики

Видеоаналитика является общим термином, используемым для описания компьютеризированной обработки и анализа видеопотоков. Компьютерный анализ видео в настоящее время реализуется в различных областях и отраслях, однако термин «видеоаналитика» обычно связан с анализом видеопотоков, захваченных системами наблюдения. Приложения видеоаналитики могут выполнять множество задач: от анализа в реальном времени видео для немедленного обнаружения интересующих событий, до анализа, предварительно записанного видео с целью извлечения событий и данных из записанного видео.

Опираться на видеоаналитику в автоматическом наблюдении за камерами и предупреждениях о событиях, представляющих интерес, во многих случаях гораздо эффективнее, чем полагаться на человека-оператора, который является дорогостоящим ресурсом с ограниченной возможностями и вниманием. Различные исследования и реальные инциденты показывают, что средний человек-оператор системы наблюдения, которому поручены наблюдательные видеоэкраны, не может оставаться внимательным и внимательным в течение более 20 минут. Более того, способность оператора отслеживать видео и эффективно реагировать на события значительно ухудшается с течением времени.

Кроме того, часто приходится проходить через записанное видео и извлекать определенные сегменты видео, содержащие событие, представляющее интерес. Эта потребность растет, поскольку использование видеонаблюдения становится все более распространенным, и количество записанного видео увеличивается. В некоторых случаях время является сущностью, и такой обзор должен проводиться эффективным и быстрым образом. Пользователи системы

видеонаблюдения также ищут дополнительные способы использования своего записанного видео, в том числе путем извлечения статистических данных для целей бизнес-аналитики. Анализ записанного видео - это потребность, для которой крайне неэффективно использовать человека-оператора из-за длительного процесса ручного перехода и наблюдения за записанным видео и связанной с ним рабочей силы для этой задачи.

Несмотря на то, что необходимость и преимущества систем наблюдения неоспоримы, а прилагаемые финансовые инвестиции в развертывание такой системы наблюдения являются значительными, фактическая выгода, получаемая от системы наблюдения, ограничена, если полагаться только на операторов-людей. Напротив, преимущества, получаемые от системы наблюдения, могут быть значительно увеличены при развертывании видеоаналитики [9].

Видеоаналитика - идеальное решение, которое отвечает потребностям операторов системы наблюдения, сотрудников службы безопасности и корпоративных менеджеров, поскольку они стремятся сделать практичное и эффективное использование своих систем наблюдения.

Простой функцией видеоаналитики является обнаружение движения с фиксированным фоном. Более технические функции могут включать в себя оценку egomotion (3D-движение камеры в среде) и отслеживание видео. Отсюда может прийти и анализ поведения. Технология, используемая для достижения видеоаналитики, включает датчики, камеры, методы сжатия изображений и методы подключения.

Видеоаналитика имеет множество преимуществ для бизнеса, особенно в сферах безопасности и общественной безопасности. Он часто используется для расширения этих двух секторов с помощью всеобъемлющих разведывательных, защитных и исследовательских возможностей. Данные, полученные с помощью видеоаналитики, предназначены для создания разумного интеллекта, чтобы помочь планировщикам городов или инфраструктуры, правоохранительным

органам или даже менеджерам и администраторам правильно и своевременно реагировать на текущие ситуации. Другие отрасли, которые использовали приложения для видеоаналитики, также включают образование, транспорт, банковские и финансовые учреждения, казино, а также игровые и государственные предприятия.

Существует также растущий спрос на видеоаналитику по причинам, не связанным с безопасностью. Одна из таких причин, которая набирает популярность, - это подсчет людей. Это приложение может помочь конечным пользователям измерять поток людей в определенных точках выхода и входа в здание, объект или дверь в режиме реального времени или во время периодической отчетности. Такие отрасли, как розничная торговля и развлечения, начали использовать видеоаналитику для этих приложений. Это помогает контролировать время пребывания в розничных магазинах. Видеоаналитика, однако, предназначена не только для делового мира. Другие полезные приложения могут включать обнаружение пламени и дыма и даже домашнюю автоматизацию.

Неожиданное увеличение популярности рынка видеоаналитики связано с увеличением угроз безопасности и необходимостью интеллектуального наблюдения. Рынок также обусловлен увеличением использования облачной вычислительной техники, растущий спрос на автоматизацию, потребность в инфраструктурной безопасности и обеспечение вычислительной среды. Технологические достижения, такие как простота установки, высокая качество видеозаписи, улучшенная удобство для пользователя и технология биометрического распознавания, также будут существенно влиять на рост рынка.

Задачи, решаемые с помощью видеоаналитики:

а) Поиск объектов и обнаружение людей.

Обнаружение и трекинг людей являются одними из основных целей использования видеоаналитики, как показано на рис. 1.1.

Рис. 1.1. Результат работы трекинга

Первый метод – прослеживание областей движущихся объектов, второй – выделение на изображении людей по данным нескольких наблюдений или метод «фильтра частиц».

Основным современным подходом к обнаружению объектов в сцене является распознавание по набору отобранных признаков, что представлено на рис. 1.2. Метод позволяет сократить количество признаков, необходимых для решения конкретной задачи распознавания, от нескольких сотен тысяч признаков (пикселов исходного изображения) до нескольких десятков или сотен признаков.

Разработанный метод состоит из 3 этапов. На первом этапе для каждого изображения из обучающей выборки производится расчет двумерных полей признаков (например, результатов обработки пространственными фильтрами, спектральных полей) и на этих полях реализуется предварительный отбор признаков по критерию соотношения общей и средней внутриклассовой дисперсии. Далее отбор осуществляется путем перебора различных комбинаций (методом последовательного присоединения и отбрасывания признаков) с использованием критерия

конкретной задачи распознавания, для решения которой отбираются признаки. На последнем этапе выбранные комбинации признаков тестируются на контрольной выборке изображений, и окончательно принимается решение о выборе набора признаков для использования при распознавании.

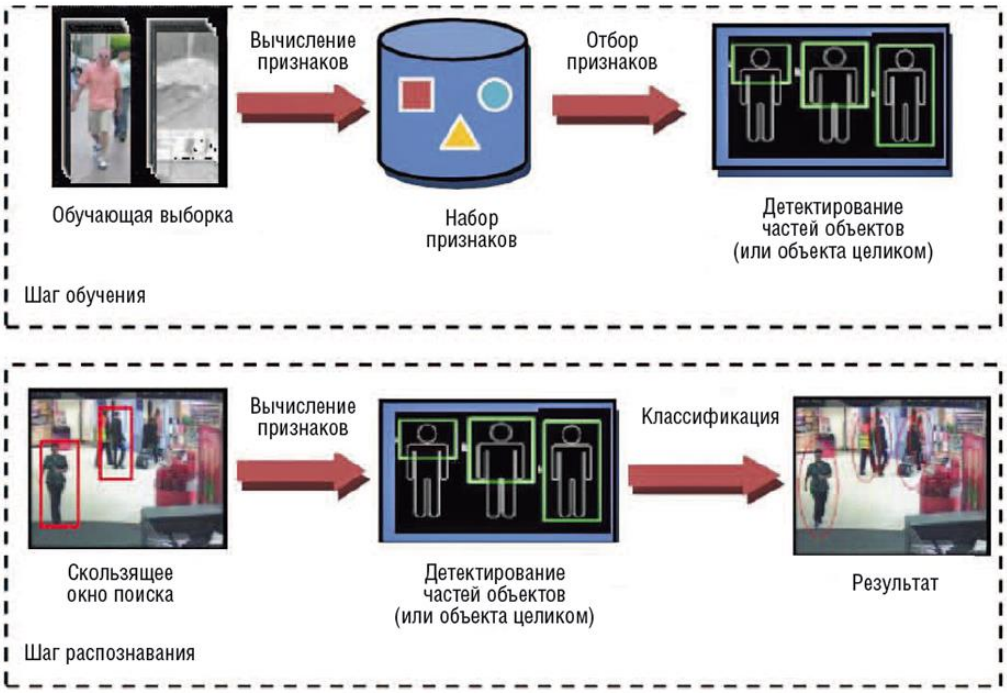

Рис. 1.2. Схема распознавания образов

б) Многокамерный трекинг.

Многокамерная видеоаналитика сопровождает объект несколькими камерами, что позволяет существенно сократить количество ложных срабатываний в месте перекрытия зон, контролируемых разными устройствами.

При межкамерном трекинге система классифицирует объект, выявляя его особенности, и пытается найти объекты с похожими характеристиками на соседних камерах. Безусловным плюсом является тот факт, что для анализа не требуется полностью декодировать видео, что снижает требования к ресурсам.

Потолочная камера, которая изображена на рис. 1.3, позволяет достоверно распознать человека, но размер контролируемой зоны ограничен

высотой подвеса камеры. Для идентификации используется ограниченный набор признаков, а при образовании плотных скоплений людей возможны ошибки [1].

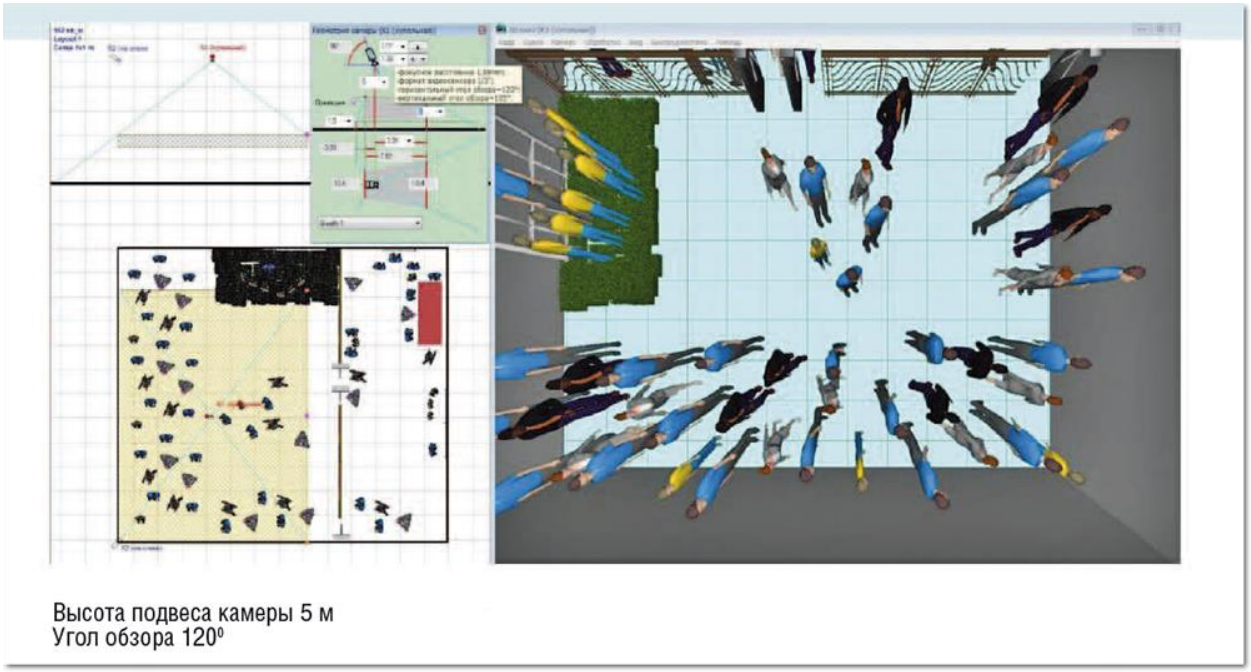

Рис. 1.3. Потолочная камера

в) Подсчет людей.

Подсчет числа людей необходим не только в целях общественной безопасности. Его можно использовать для оценки числа посетителей, регулирования количества открытых касс, планирования мест установки рекламных носителей. Системы подсчета людей все чаще применяются организациями розничной торговли — на смену прежним инфракрасным горизонтальным или термокамерам приходят видеосчетчики, которые в корне поменяли рынок систем подсчета посетителей, вытесняя все остальные решения [2].

Их достоинства — малый объем монтажных работ, отсутствие помех для персонала и покупателей, максимальная физическая защита от преднамеренного искажения данных, простота в настройке, возможности использования в смежных областях (безопасность, видеоаналитика). В каждом сегменте розничной торговли (магазины, торговые сети и пр.)

применяются свои системы видеоподсчета. Они могут использовать аналитическое ПО и программы для настройки оборудования, предусматривать интеграцию со сторонним программным обеспечением и внедрение дополнительных функций («траектория движения», поведение покупателей, «горячие» и «холодные» зоны, оптимальная выкладка товара). Однако во многих случаях достаточно дорогое аналитическое ПО не требуется, а возможность самостоятельной установки системы позволяет сэкономить значительные средства.

В зависимости от ракурса съемки и плотности толпы могут использоваться разные сценарии подсчета людей. Первый — плотная толпа. Такая ситуация характерна для мероприятий (митингов, концертов), а также часов пик, когда формируются плотные людские потоки. В этом случае применяются методы подсчета на основе занимаемой толпой площади с учетом перспективы и «пестроты» текстуры. При достаточной величине объектов и их движении можно задействовать методы на основе кластеризации траекторий независимо отслеживаемых точек (синхронные траектории с высокой степенью вероятности относятся к одному объекту).

Последние методы позволяют определять людей с точностью до 94%. При этом допускаются разные ракурсы и конфигурации сцены [3]. С помощью данного метода можно оценивать количество проходящих людей, но он плохо работает с малоподвижными людьми, к тому же для него характерна низкая скорость — для обработки одного кадра требуется более 10 сек. Если размер лица слишком мал, а объекты значительно перекрываются, анализ затрудняется, да и толпа уже не разбивается на несколько небольших групп.

Второй сценарий — разреженный поток людей, например, на улицах или в крупных магазинах. Для данной ситуации характерны слишком маленькие изображения лиц, неплотные группы людей и обширные перекрытия внутри групп. В этом случае применяются методы на основе вычитания фона, которые при невысокой скорости характеризуются

возможностью работы с неподвижными и малоподвижными компонентами объектов (их должно быть не более 10). Средняя ошибка составляет 1,2 человека (в сцене от 11 до 45 человек).

Если у изображения высокое разрешение, а перекрытия невелики, подсчет людей может вестись путем обнаружения с сопровождением между кадрами, как на примере рис. 1.4. Такой сценарий используется в магазинах при оценке количества покупателей и длины очереди, когда камеру можно разместить у входа в помещение или у кассы, однако его точность (достаточная для данного применения) зависит от ракурса и качества трекинга.

Рис. 1.4. Подсчет людей через обнаружение

Зенитное (потолочное) расположение камеры способствует большей точности, повышает скорость обработки информации, упрощает методы подсчета. В этом случае можно применять методы на основе оптического потока, а также поиска и сопровождения головы человека. Их заявленная точность превышает 95%, а ошибка (когда за людей принимаются другие объекты) составляет около 10–20% (в сторону завышения). Это достаточно хорошая точность. Проблемы могут возникнуть, если существенную часть проходящих в контролируемой зоне людей составляет персонал либо посетители с тележками, большими сумками или коробками, а кроме того, если рядом с камерой стоит охранник, кто-то курит, говорит по телефону или ждет кого-то.

1.2 Алгоритмы сжатия, применяемые к видео

Сжатие видео использует современные методы кодирования для уменьшения избыточности в видеоданных. Большинство алгоритмов сжатия видео и кодеков сочетают сжатие пространственного изображения и временную компенсацию движения. Сжатие видео является практической реализацией исходного кода в теории информации.

Большинство алгоритмов сжатия видео используют сжатие с потерями. Для несжатого видео требуется очень высокая скорость передачи данных. Хотя кодеки сжатия видео без потерь выполняют с коэффициентом сжатия 5-12, типичное видео с потерей сжатия MPEG-4 имеет коэффициент сжатия от 20 до 200. Как и при любом сжатии с потерями, существует компромисс между качеством видео, стоимость обработки сжатия и декомпрессии, а также системные требования. Высоко сжатое видео может представлять видимые или отвлекающие искажения.

Некоторые схемы сжатия видео, как правило, работают на квадратные группы соседних пикселей, которые часто называют макроблоками [4]. Эти пиксельные группы или блоки пикселей сравниваются от одного кадра к другому, а кодек сжатия видео передает только различия в этих блоках. В областях видео с большим движением сжатие должно кодировать больше данных, чтобы не отставать от большего количества пикселей, которые меняются. Обычно во время взрывов, пламени, стай животных и в некоторых панорамных снимках высокочастотная деталь приводит к снижению качества или увеличению переменного битрейта.

Видеоданные могут быть представлены как серия кадров неподвижного изображения. Последовательность кадров содержит пространственную и временную избыточность, которые алгоритмы сжатия видео пытаются устранить или закодировать в меньшем размере. Сходства

можно кодировать только путем хранения различий между кадрами или с использованием перцепционных особенностей человеческого зрения. Например, небольшие различия в цвете более трудно воспринимаются, чем изменения яркости. Алгоритмы сжатия могут усреднять цвет в этих похожих областях для уменьшения пространства, аналогично тем, которые используются при сжатии изображений JPEG. Некоторые из этих методов по своей сути являются потерями, в то время как другие могут сохранять всю соответствующую информацию из исходного несжатого видео.

Одним из самых мощных методов сжатия видео является межкадровое сжатие. Сжатие между кадрами использует один или несколько ранних или более поздних кадров в последовательности для сжатия текущего кадра, тогда как внутрикадровое сжатие использует только текущий кадр.

Самый мощный используемый метод работает, сравнивая каждый кадр в видео с предыдущим. Если в кадре есть области, где ничего не двигалось, система просто выдает короткую команду, которая копирует ту часть предыдущего кадра, в следующую. Если части рамы перемещаются простым способом, то компрессор дает команду, которая сообщает декомпрессу сдвинуть, повернуть, осветить или затемнить копию. Эта более длинная команда по-прежнему остается намного короче, чем внутрикадровое сжатие. Межкадровое сжатие хорошо работает для программ, которые будут просто воспроизводиться зрителем, но могут вызывать проблемы, если необходимо отредактировать видеопоследовательность.

Поскольку межкадровое сжатие копирует данные из одного кадра в другой, исходный кадр просто вырезается (или теряется при передаче), следующие кадры не могут быть восстановлены должным образом. Некоторые видеоформаты, такие как DV (digital video), сжимают каждый кадр независимо, используя внутрикадровое сжатие. Создание «разрезов» во внутрикадровом сжатии видео почти так же просто, как редактирование несжатого видео: можно найти начало и конец каждого кадра и просто копировать исходный бит в бит каждого кадра, который нужно сохранить, и

отбрасывает кадр. Другое различие между внутрикадровым и межкадровым сжатием заключается в том, что с внутрикадровыми системами каждый кадр использует аналогичный объем данных. В большинстве межкадровых систем определенные кадры (такие как «I-кадры») не разрешено копировать данные из других фреймов, поэтому они требуют гораздо больше данных, чем рядом с другими кадрами.

Существует возможность создания компьютерного видеоредактора, который обнаруживает проблемы, возникающие при редактировании фреймов, в то время как другие фреймы нуждаются в них. Это позволило использовать новые форматы, такие как HDV для редактирования [6]. Однако этот процесс требует гораздо большей вычислительной мощности, чем редактирование внутрикадрового сжатого видео с тем же качеством изображения.

Сегодня почти все широко используемые методы сжатия видео (например, стандарты, одобренные ITU-T или ISO) применяют дискретное косинусное преобразование (DCT) для уменьшения пространственной избыточности. DCT, который широко используется в этом отношении, был введен Н.Ахмедом , Т. Натараджаном и К.Р. Рао в 1974 г. Другие методы, такие как фрактальное сжатие, поиск наилучших совпадений и использование дискретного вейвлет-преобразования (DWT), были предметом некоторых исследований, но обычно не используются в практических продуктах (за исключением использования вейвлет-кодированиякак неподвижные кодировщики без компенсации движения). Интерес к фрактальному сжатию, по-видимому, ослабевает из-за недавнего теоретического анализа, свидетельствующего о сравнительном отсутствии эффективности таких методов.

ГЛАВА 2. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ПОИСКА И ПРОЦЕССОВ РАСПОЗНАВАНИЯ ОБЪЕКТОВ НА ВИДЕО

В этой статье рассматривается ряд методов, используемых в приложениях видеонаблюдения для обнаружения и классификации объектов. Более того, использование этих методов в беспроводной сети наблюдения способствуют снижению энергопотребления устройств, поскольку они уменьшают объем информации, передаваемой через сеть.

2.1 Обнаружение объектов методом вычитания фона

Процесс идентификации объекта или события в видео начинается с сегментации захваченного кадра, чтобы обнаружить интересующие области. Эти области интересов включают объекты без фона. Выполнение этого шага имеет решающее значение для достижения желаемой точности в конце идентификации объектов. Существуют различные методы, включая легкую обработку изображений, которые подходят для устройств с малыми возможностями обработки, а также более сложные статистические методы, требующие большей вычислительной мощности и памяти. В этом разделе рассмотрим наиболее популярные методы, которые являются предпочтительными для этапа извлечения идентификации объекта. Вычитание фона (BS) - это общее название самого популярного метода извлечения, который используются для обнаружения движущихся объектов в видеопотоках. BS широко используется в приложениях наблюдения, чтобы отличать объекты переднего плана от базовой модели до выполнения дополнительных задач, таких как классификация, локализация или отслеживание [7]. Процесс BS требует сохранения модели фона и использует

эту модель, чтобы извлечь пиксели переднего плана из видеокадра как показанном на рис. 2.1.

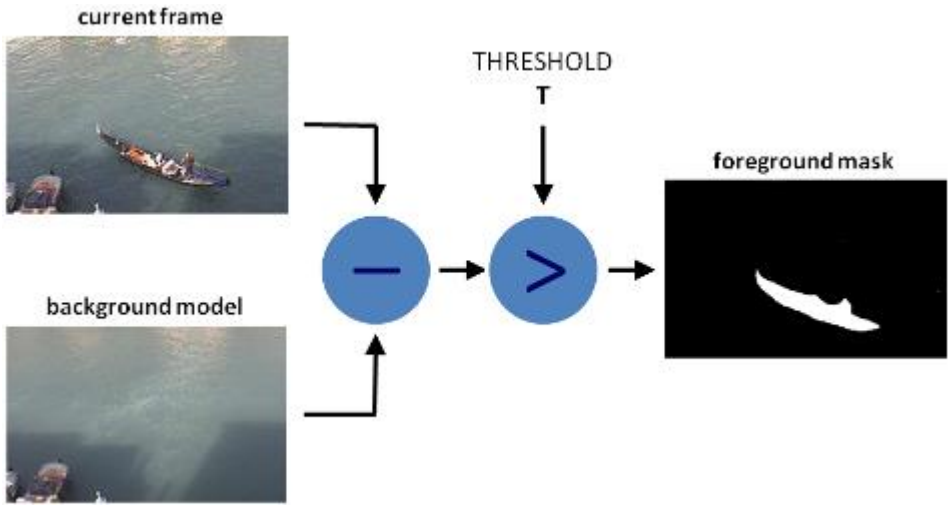

Рис. 2.1. Пример обработки изображения методом вычитания фона

Существуют различные подходы к построению этой фоновой модели, такие как сохранение неподвижной фоновой рамки, построение адаптивного фона, используя статистические методы или временное моделирование, чтобы извлечь на переднем плане разницу между фоновой моделью и текущим кадром видео. Самый простой способ вычитания фона – это вычислить разницу между значениями пикселей текущего кадра и стационарной системы отсчета, а затем применить порог к результату в чтобы обнаружить нефонические области. Этот способ не подходит для наружного применения в частности, потому что невозможно принять фоновую модель как статическую. Световые изменения, перемещение фоновых объектов, таких как листья или изменения угла обзора камеры требуют обновлять в фоновом режиме модель. По этой причине статистические методы, такие как применение гауссовского среднего или смесь гауссовых, используются эффективно для сохранения и обновления фона. Используя эти методы, фоновое значение пикселя вычисляется периодически, применяя статистические формулы, такие как усреднение или функции плотности вероятности. Временное различие кадров - еще один эффективный подход, который рассматривается как адекватный метод

фонового моделирования. Этот метод предлагает сохранить последние несколько кадров в качестве фоновой модели, что требует дополнительное пространство [7]. Среднее значение рассчитывается с использованием этих кадров, а обнаружение переднего плана согласно этому медианному значению. Этот подход очень успешный при обнаружении движущихся объектов, но он не работает, когда объект переднего плана прекращает движение. Одним из наиболее предпочтительных методов фонового моделирования является мультимодальный метод фонового моделирования, который имеет дело с несколькими фоновыми объектами в одном и том же местоположение в разные временные рамки. Пиксель моделируется взвешенной комбинацией нескольких функций плотности вероятности (PDF), а не одной PDF. Этот метод также называется смесью гауссовых. На практике количество PDF-файлов устанавливается в пределах от 3 до 5. Простые методы, такие как гауссовские среднее предложение обеспечивает приемлемую точность при достижении высокой частоты кадров и не очень требовательны к памяти.

2.2 Постобработка переднего плана изображения

Как было сказано выше, успех процесса идентификации тесно связан с успехом фазы извлечения объекта. Хотя методы BS выделяют передний план, который содержит объекты, представляющие интерес, этот результат все еще содержит шум, который снижает точность и эффективность процесса классификации. Чтобы предотвратить это - остатки и шум на переднем плане должны быть удалены перед классификацией. Помимо очистки шума, объекты переднего плана необходимо конкретизировать для лучшей классификации. С этой целью первичные морфологические операции эрозии и дилатации применяются к двоичному изображению. Применение операции открытия и закрытия поочерёдно. Выполнение этих операций на двоичном

изображении означает зондирование изображения с помощью структурирующего элемента, чтобы увеличить область переднего плана и сократить промежутки или наоборот. Медианный фильтр является одним из самых популярных нелинейных фильтров для удаления шумов из-за его хорошей де-шумовой мощности и вычислительной эффективности. Это широко используется для удаления соляных и перечных типов шумов в изображении. Идея медианного фильтра заключается в том, что он должен заменить значение пикселя медианой соседних значений, который образует окно. Применение этих морфологических операций на двоичном изображении переднего плана, которое извлекается методом BS, показано на рис. 2.2.

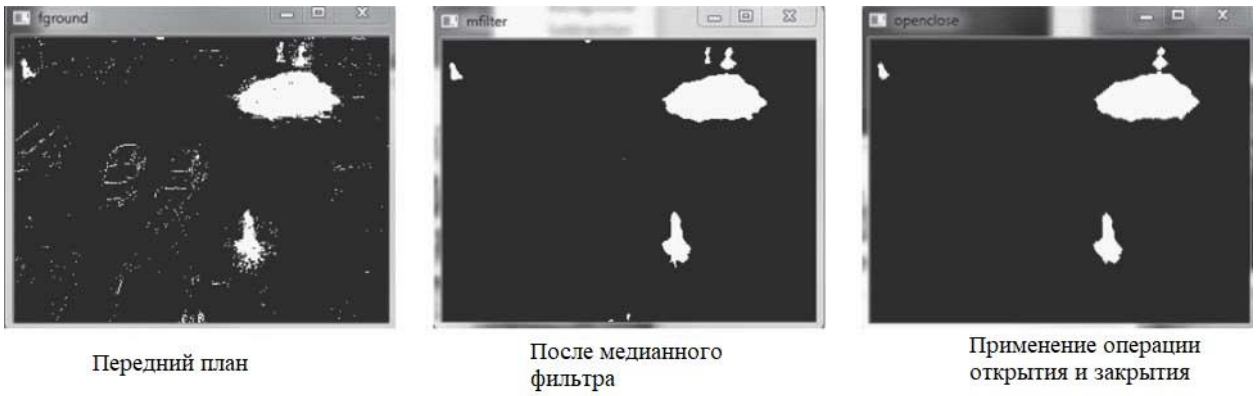

Рис. 2.2. Процесс постобработки

2.3 Сегментация и извлечение объекта

Сегментация изображений означает разбиение изображения на значимые области. Все пиксели в области аналогичны по некоторым характеристикам или вычисленному свойству, таким как цвет, интенсивность или текстура. Переднее изображение, которое извлекается с использованием BS, может содержать несколько различных объектов. Каждый из этих объектов обрабатывается отдельно во время классификации. Алгоритмы

сегментации применяются на изображениях переднего плана, чтобы извлекать области реальных объектов.

Принято классифицировать методы сегментации изображений на три класса: пороговое значение, обнаружение границ и извлечение области [9]. В подходе выбора порога, изображение разделяется на основе одного или более пороговых значений. Этот метод можно использовать в качестве простого подхода сегментации для разбиения изображения на фоновые и передние области или как более сложный подход для извлечения нескольких объектов с использованием нескольких пороговых значений.

Методы обнаружения кромок преобразуют изображения в крайние изображения, благодаря изменениям серого тона в изображениях. Так, существуют три основных метода обнаружения края в литературе: Робертс, Прэвит и Собель. Эти методы обнаруживают изменения градиента, вычисляя различия уровня серого между соседними пикселями. Сегментированное изображение переднего плана с использованием алгоритма обнаружения края показано на рис. 2.3.

Обработка переднего плана изображения

Сегментация

Рис. 2.3. Сегментирование изображения переднего плана

Другой эффективный метод сегментации изображений - это итеративная классификация пикселей, в которой решения, связанные с пикселями, выполняются в соответствии со своими соседями итеративно. Одним из применений этого метода является операция маркировки подключенных компонентов в двоичных изображениях. Для двоичного изображения, представленного в виде массива d-мерных пикселей или элементов изображения, маркировка подключенных компонентов представляет собой процесс назначения меток элементам BLACK-изображения таким образом, что смежные BLACK-элементы изображения назначаются на одну и ту же метку.

Даже просто удаление объектов может способствовать умению приложений наблюдения. Эти методы, введенные до настоящего времени, значительно облегчают работу операторов. Вместо того, чтобы иметь дело с непрерывными сырыми мультимедийными данными в реальном времени, операторы получают и анализируют лишь небольшую часть данных, которая состоит из угроз. Кроме того, для беспроводной сети мультимедийных датчиков значительное уменьшение количества передаваемых данных приводит к продлению срока службы устройства. Для целей классификации или отслеживания необходимы дальнейшие операции. Эти операции описаны в следующих разделах.

2.4 Особенности объектов

Для построения системы интеллектуального наблюдения следующий шаг - идентифицировать извлеченный объект. Для этой цели используются конкретные функции объекта. Эти функции помогают идентифицировать тип объектов. Эти функции могут быть простыми структурами, такими как точка,

угол и край, а также более сложные структуры, такие как текстура, blob или сам объект.

Особенности могут быть классифицированы как функции на основе формы, такие как соотношение сторон (ширина / высота), сложность формы (периметр / область), края, углы; функции на основе текстур, такие как функции Gabor; и движения, такие как скорость [11]. Стоит использовать отношение ширины / высоты ограничивающего прямоугольника объекта, а также скорость в их исследовании, чтобы классифицировать, является ли обнаруженный объект человеком, транспортным средством или животным. Можно также использовать соотношение ширины / высоты и ритм ходьбы, чтобы классифицировать объекты в качестве пешехода, транспортного средства или мотоцикла в системе уличного наблюдения. Также функции компактности и дисперсии способны отличать людей от транспортных средств для произвольной сцены. Поскольку функции на основе формы легко извлекаются, они предпочтительны в приложениях с ограничением мощности, таких как системы беспроводного видеонаблюдения.

Хотя функции на основе формы очень эффективны в процессах идентификации объектов, на большинство функций влияют изменения освещения, масштабирование, положение и угол камеры или направление объекта. Для извлечения более стабильных и инвариантных признаков предлагается несколько методов. Одним из таких методов является преобразование инвариантных показателей масштаба (SIFT. В методе SIFT изображение преобразуется в локальные векторы признаков, применяя поэтапную технологию фильтрации. Ускоренные надежные характеристики (SURF) - аналогичный подход, который является более эффективным с точки зрения извлечения признаков. SURF использует приближение матрицы Гессиана на интегральном изображении, а не на исходном изображении, чтобы обнаружить точки признаков. Гистограммы дескрипторов ориентированного градиента (HOG) также являются инвариантными по масштабам чертами, которые первоначально были предложены для

обнаружения людей на изображениях. Метод HOG основан на идее, что объект может быть достаточно хорошо охарактеризован распределением локальных градиентов интенсивности или направлений краев.

Помимо распознавания объектов, они также выполняют запросы, основанные на локальных художественных дескрипторах объектов. Например, для отслеживания нескольких типов объектов в видео используют функции SIFT и методы сопоставления подобия для соответствия объекту между видеокадрами. Поскольку требуется время, память устройства и вычислительная мощность для извлечения и хранения, использование локальных функций в ограниченных средах, таких как беспроводные мультимедийные датчики, является сложной задачей.

2.5 Классификация сегментированных объектов

Распознавание объектов - это наиболее важный этап, который реализует цели построения системы интеллектуального видеонаблюдения. На этом этапе тип или идентичность объекта переднего плана в видео определяется с помощью его извлеченных признаков. Существуют различные категории классификации, такие как минимальное расстояние, максимальное количество правдоподобия, нечеткие множества, нейронные сети и деревья принятия решений. Эти классификаторы в основном классифицируются на основе наличия данных обучения или распространения собранной информации.

Одним из наиболее популярных подходов к классификации является использование наборов данных, содержащих достаточное количество выборок для каждого типа объекта, для обучения системы наблюдения. Система может определять класс нового образца с использованием статистических методов, таких как максимальное правдоподобие, средства и

т.д. Этот тип классификации называется контролируемой классификацией. Также, используются различные методы контролируемой классификации для целей распознавания или отслеживания объектов в реальном времени.

Когда данные обучения недоступны или достаточны для принятия решения, предпочтительным является метод неконтролируемой классификации. Неконтролируемые методы классификации включают использование методов кластеризации, таких как метод центроида и метод группового усреднения. Используя эти методы, определяются спектрально разделяемые классы.

Когда распределение собранных данных вписывается в определенный шаблон, такой как гауссовское распределение, можно угадать параметры классификации, такие как средняя векторная или ковариационная матрица. В этих условиях классы объектов определяются применением статистических методов, которые используют эти параметры. Эти типы классификаторов называются параметрическими классификаторами. Если невозможно сопоставить распределение данных по любому шаблону, то предположение параметра не может быть выполнено. Классификация выполняется с применением методов кластеризации, таких как k-Nearest Neighborhood (kNN) или окно Parzen. Эти типы классификаторов называются непараметрическими классификаторами.

Одним из популярных методов классификации является kNN, который является непараметрическим (не делает никаких предположений о распределении данных) и ленивым обучением (не использует методы обучения для обобщения) для классификации. Классификация основана на расстоянии между дескрипторами признаков и дескрипторами образцов. Еще один эффективный классификатор - это вектор-машина поддержки (SVM), которая также является непараметрическим и контролируемым алгоритмом классификации. Классификация основана на гиперплоскости с максимальным расстоянием до точек данных обучения. Эти классификаторы, показанные на рис. 2.4, являются предпочтительными для приложений

классификации в режиме реального времени в средах с ограниченными ресурсами из-за более низкой вычислительной сложности, простоты и меньших требований к объему хранилища.

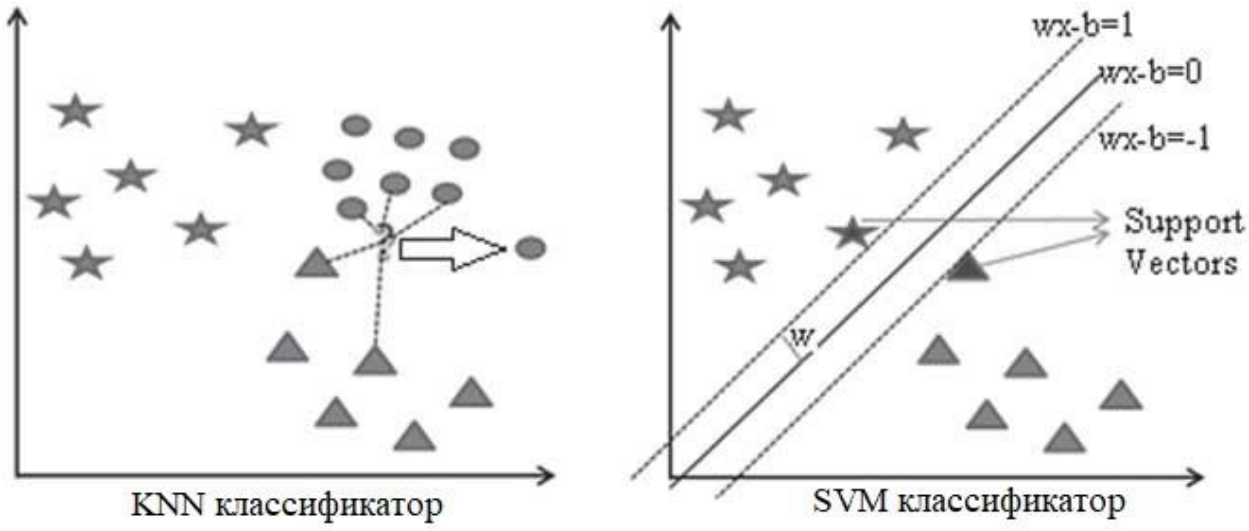

Рис. 2.4. Методы классификации

Искусственная нейронная сеть (ANN) также является популярным классификатором и имеет вычислительную модель, подобную человеческому мозгу. Классификатор имеет многоуровневую сетевую архитектуру, и каждый слой состоит из узлов (нейронов), соединенных друг с другом несколькими входами и выходами (ребрами). Во время тренировочной фазы края взвешиваются для получения правильной выходной метки. После этого сеть формирует функцию, а затем, используя эту функцию, сеть может прогнозировать вывод для данного входа. Из-за сложности вычислений классификаторы типа ANN не подходят для приложений с ограниченными ресурсами.

ГЛАВА 3. РАЗРАБОТКА СИСТЕМЫ ПРОГНОЗИРОВАНИЯ ПОСЕЩАЕМОСТИ МАГАЗИНА

В результате данной дипломной работы должно быть приложение, включающее в себя программу-детектор людей, получающую на вход видеопоток. Затем, данные, полученные в результате работы алгоритма поиска людей на видео, будут отправляться на хост для дальнейшего анализирования.

Разработку системы прогнозирования можно условно разделить на следующие этапы:

- предварительная обработка видеопотока и определение параметров движения;

- реализация алгоритма для распознавания и подсчета людей;

- передача данных на хост, для дальнейшего анализирования и представления результата.

3.1 Предварительная обработка потока видео для задач определения параметров движения объектов

OpenCV имеет возможность для интеграции в нее предварительно подготовленной модели HOG + Linear SVM, которая может использоваться для обнаружения пешеходов как в изображениях, так и на видео. Описание принципа работы метода HOG представлено в главе 3.2.

Листинг 3.1. Предварительная обработка кадров

```
# import the necessary packages
from __future__ import print_function
from imutils.object_detection import non_max_suppression
from imutils import paths
import numpy as np
```

```
import argparse
import imutils
import cv2
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--images", required=True, help="path to images directory")
args = vars(ap.parse_args())
# initialize the HOG descriptor/person detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```
Конец листинга.

Сначала выполняется импорт необходимых пакетов. Импортируем print_function, чтобы гарантировать, что код совместим как с Python 2.7, так и с Python 3 (этот код также будет работать для OpenCV 2.4.X и OpenCV 3). Также, импортируем функцию non_max_suppression из пакета imutils. Суть алгоритма подавления немаксимальности состоит в том, чтобы взять несколько перекрывающихся ограничивающих прямоугольников и свести их только к одному ограничивающему прямоугольнику, его результат представлен на рис.3.1. Это помогает уменьшить количество ложных срабатываний, сообщаемых детектором конечных объектов.

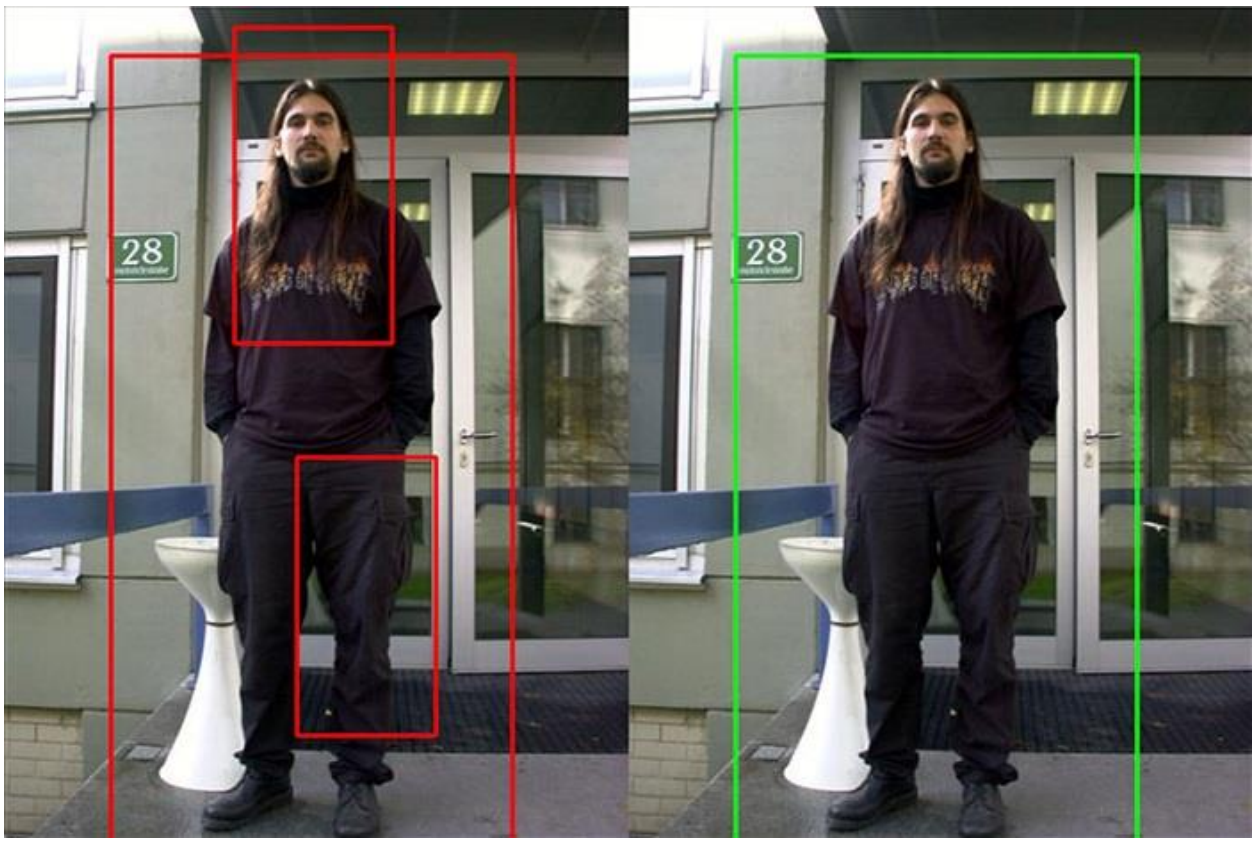

Рис.3.1. Результат выполнения функции non_max_suppression

Затем выполняется обработка аргументов командной строки. Нужен только один параметр, - изображения, который представляет собой путь к каталогу, содержащий список изображений, на которых будет выполняться обнаружение пешеходов. Далее инициализируется детектор пешеходов. Сначала вызываем hog = cv2.HOGDescriptor(), который инициализирует дескриптор гистограммы ориентированных градиентов. Затем вызываем setSVMDetector, чтобы установить метод опорных векторов, который был предварительно обучен детектором пешеходов и загруженным функцией cv2.HOGDescriptor _getDefaultPeopleDetector ().

Листинг 3.2. Обнаружение объектов

```
# loop over the image paths
for imagePath in paths.list_images(args["images"]):
        # load the image and resize it to (1) reduce detection time
        # and (2) improve detection accuracy
        image = cv2.imread(imagePath)
        image = imutils.resize(image, width=min(400, image.shape[1]))
        orig = image.copy()
        # detect people in the image
        (rects, weights) = hog.detectMultiScale(image, winStride=(4, 4),
```

```
        padding=(8, 8), scale=1.05)
# draw the original bounding boxes
for (x, y, w, h) in rects:
        cv2.rectangle(orig, (x, y), (x + w, y + h), (0, 0, 255), 2)
# apply non-maxima suppression to the bounding boxes using a
# fairly large overlap threshold to try to maintain overlapping
# boxes that are still people
rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
pick = non_max_suppression(rects, probs=None, overlapThresh=0.65)

# draw the final bounding boxes
for (xA, yA, xB, yB) in pick:
        cv2.rectangle(image, (xA, yA), (xB, yB), (0, 255, 0), 2)
# show some information on the number of bounding boxes
filename = imagePath[imagePath.rfind("/") + 1:]
print("[INFO] {}: {} original boxes, {} after suppression".format(
        filename, len(rects), len(pick)))
# show the output images
cv2.imshow("Before NMS", orig)
cv2.imshow("After NMS", image)
cv2.waitKey(0)
```

Конец листинга.

В листинге 3.2 представлен цикл, который обрабатывает все кадры из директории, в которой они хранятся. Также, изображение обрабатывается, после чего, его размер изменяется до максимальной ширины в 400 пикселей. Причина, по которой необходимо уменьшить размеры изображения, следующие:

1) уменьшение размера изображения гарантирует, что потребуется меньшее количество раздвижных окон в пирамиде изображения, что сокращает время обнаружения объекта и увеличивает общую пропускную способность обнаружения.
2) Изменение размера изображения также улучшает общую точность обнаружения пешеходов, т. е. меньше ложных срабатываний.

Фактически обнаружение пешеходов в изображениях обрабатывается, делая вызов метода detectMultiScale дескриптора hog.detectMultiScale и создает пирамиду изображения с масштабом 1,05 и размером шага скольжения (4, 4) пикселей как в направлении x, так и в направлении y соответственно.

Размер скользящего окна фиксирован на уровне 64 x 128 пикселей, как это предлагает оригинальный стандарт гистограммы ориентированных

градиентов для обнаружения человека. Функция detectMultiScale возвращает ограничивающий прямоугольник (x, y) -координаты каждого человека на изображении и весовое значение, возвращаемое SVM для каждого обнаружения.

Больший размер шкалы будет оценивать меньше слоев в изображении, что может ускорить выполнение алгоритма. Однако наличие слишком большого масштаба может привести к тому, что пешеходы не будут обнаружены. Мало того, что это может быть вычислительно в ущерб производительности, оно также может значительно увеличить количество ложных срабатываний, обнаруженных детектором пешехода. Тем не менее, масштаб является одним из самых важных параметров для настройки при обнаружении пешеходов.

Также, в листинге 3.2 представлено рисование исходных ограничивающих прямоугольников на изображении. Однако для некоторых изображений можно заметить, что для каждого человека обнаружены множественные перекрывающиеся ограничивающие поля (как показано на рисунке 3.1 выше).

В этом случае есть два варианта. Можно обнаружить, что один ограничивающий блок полностью содержится внутри другого. Или можно применять функцию non_max_suppression и подавлять ограничивающие прямоугольники, которые перекрываются со значительным порогом.

После применения функции non_max_suppression рисуются финальный вариант ограничивающего поля, отображается некоторая основная информация об изображении и количестве ограничивающих прямоугольников.

3.2 Гистограмма ориентированных градиентов (HOG)

Дескриптор функции HOG, используемый для обнаружения пешеходов, вычисляется на патче 64 × 128 изображения. Конечно, изображение может быть любого размера. Обычно патчи в нескольких масштабах анализируются во многих местах изображения. Единственное ограничение состоит в том, что анализируемые патчи имеют фиксированное соотношение сторон. В данном случае патчи должны иметь соотношение сторон 1: 2. Например, они могут быть 100 × 200, 128 × 256 или 1000 × 2000, но не 101 × 205.

Чтобы проиллюстрировать этот момент, покажем большое изображение размером 720 × 475. Теперь выберем патч размером 100 × 200 для вычисления дескриптора функции HOG. Этот патч обрезается из изображения и изменяется до 64 × 128, как показано на рис. 3.2. Теперь можно рассчитать дескриптор HOG для этого патча изображения.

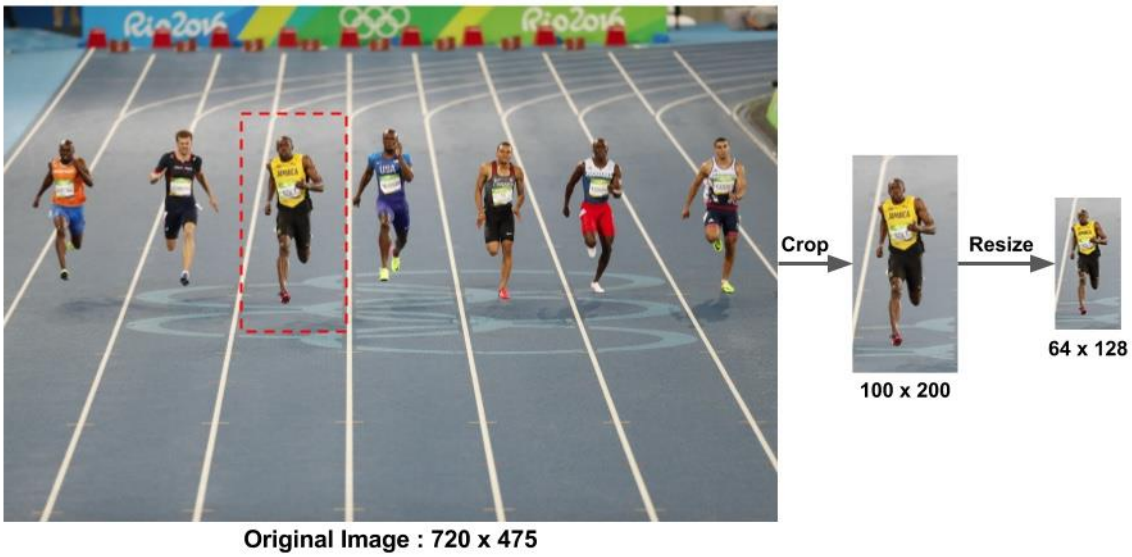

Рис. 3.2. HOG-обработка

В документе Далала и Триггса также упоминается гамма-коррекция как шаг предварительной обработки, но прирост производительности незначительный, и поэтому пропускаем этот шаг.

Чтобы вычислить дескриптор HOG, нужно сначала вычислить горизонтальный и вертикальный градиенты. В конце концов, нужно рассчитать гистограмму градиентов. Это легко достигается путем фильтрации изображения с помощью следующих ядер, представленных на рис. 3.3.

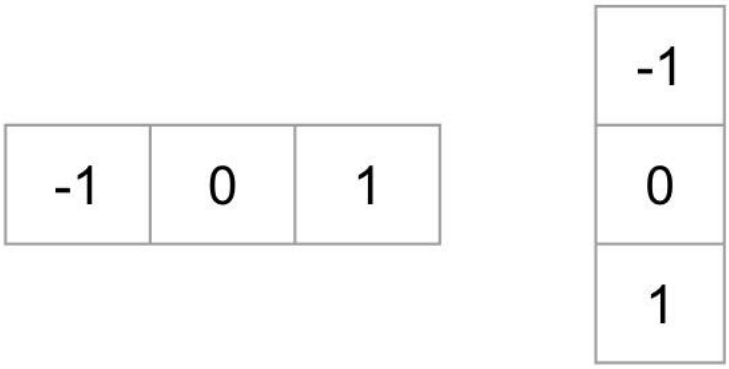

Рис. 3.3. Градиентное ядро

Также можем добиться тех же результатов, используя оператор Sobel в OpenCV с размером ядра 1 [7], что представлено в листинге 3.3.

Листинг 3.3. Пример использования оператора Sobel

```
# Read image
im = cv2.imread('bolt.png')
im = np.float32(im) / 255.0
# Calculate gradient
gx = cv2.Sobel(img, cv2.CV_32F, 1, 0, ksize=1)
gy = cv2.Sobel(img, cv2.CV_32F, 0, 1, ksize=1)
```

Конец листинга.

Затем можем найти величину и направление градиента, используя следующую формулу:

$$g = \sqrt{g_x^2 + g_y^2}$$
$$\theta = \arctan \frac{g_y}{g_x} \quad . \qquad (3.1)$$

На рис. 3.4 показаны градиенты:

Рис. 3.4. Абсолютные значения и величина градиента

Следует обратить внимание, что x-градиент срабатывает по вертикальным линиям, а y-градиент срабатывает по горизонтальным линиям. Величина градиентных пожаров наблюдается только там, где есть резкое изменение интенсивности. Ни один из них не срабатывает, когда область гладкая.

Изображение градиента удалило много несущественной информации (например, постоянный цветной фон), но выделил контуры. Другими словами, можно посмотреть изображение градиента и все еще легко сказать, что на картинке есть человек.

На каждом пикселе градиент имеет величину и направление. Для цветных изображений оцениваются градиенты трех каналов (как показано на рисунке выше). Величина градиента в пикселе - это максимальная величина градиентов трех каналов, а угол - угол, соответствующий максимальному градиенту.

На следующем этапе изображение делится на 8 × 8 ячеек, и гистограмма градиентов вычисляется для каждой ячейки 8 × 8, как показано на рис. 3.5.

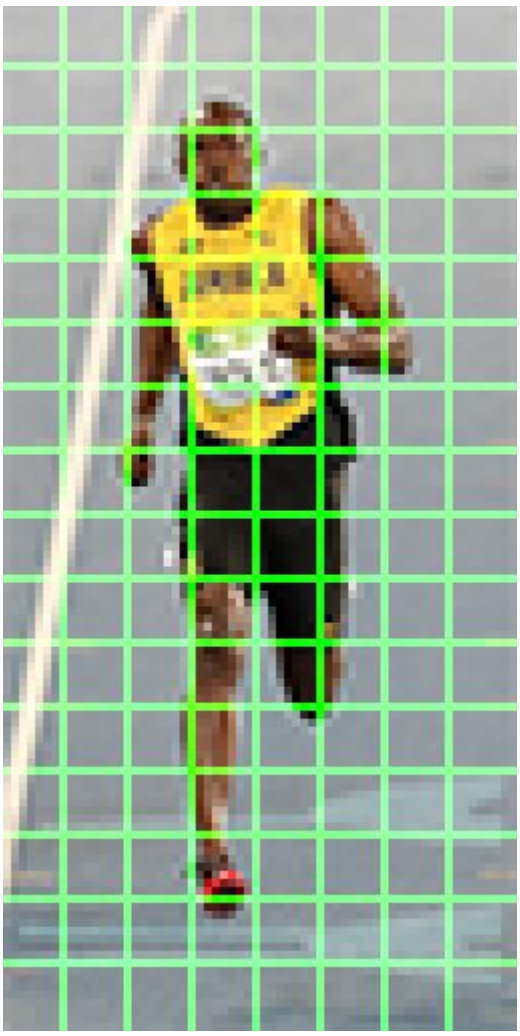

Рис. 3.5. Разбиение изображения на ячейки

Одной из важных причин использования дескриптора функции для описания патча изображения является то, что он обеспечивает компактное представление. Шаблон изображения 8 × 8 содержит 8x8x3 = 192 пиксельных значения. Градиент этого патча содержит 2 значения (величина и направление) на пиксель, который добавляет до 8x8x2 = 128 чисел. В конце этого параграфа увидим, как эти 128 чисел представлены с использованием гистограммы 9-бина, которую можно сохранить в виде массива из 9 чисел. Мало того, что представление более компактно, вычисление гистограммы поверх патча делает это представление более надежным для шума. У отдельных градиентов может быть шум, но гистограмма поверх 8 × 8 патчей делает представление менее чувствительным к шуму.

Первоначально HOG использовался для обнаружения пешеходов. 8 × 8 ячеек на фотографии пешехода размером до 64 × 128 достаточно велики, чтобы захватывать интересные функции (например, лицо, верх головы и т. д.).

Гистограмма представляет собой, по существу, вектор (или массив) из 9 ячеек (чисел), соответствующих углам 0, 20, 40, 60 ... 160.

Давайте посмотрим на один патч 8 × 8 на изображении и посмотрим на рис.3.6, как выглядят градиенты.

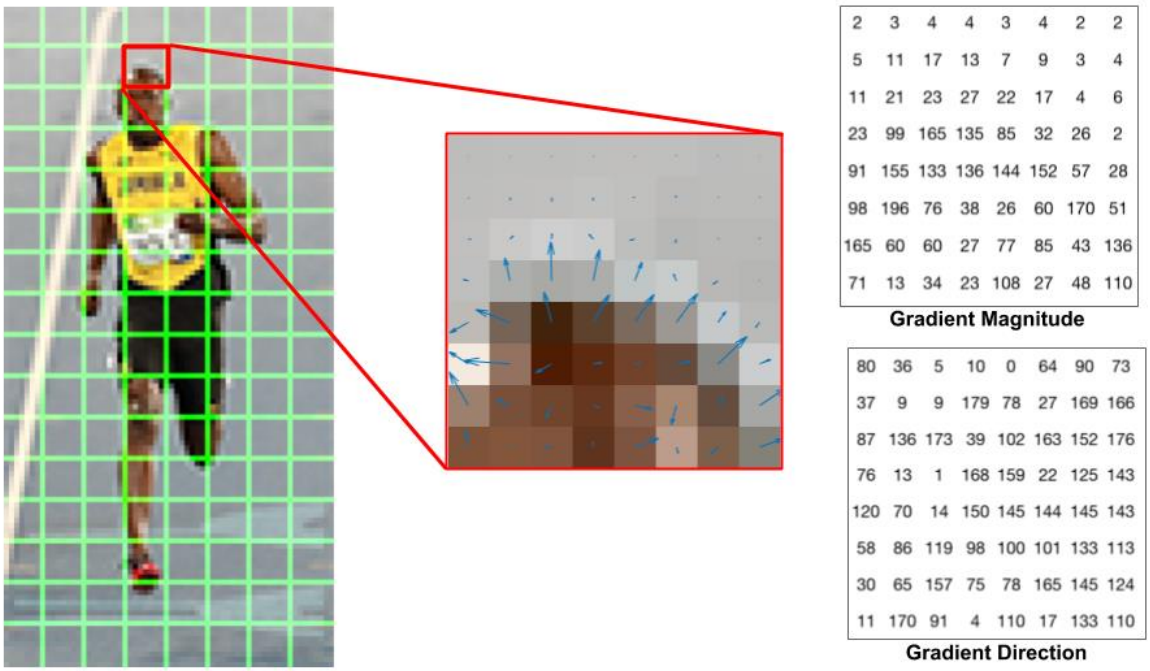

Рис. 3.6. Патч RGB и градиенты, представленные стрелками

Справа видим необработанные числа, представляющие градиенты в ячейках 8 × 8 с одной незначительной разницей - углы между 0 и 180 градусами вместо 0 до 360 градусов. Они называются «без знаковыми» градиентами, потому что градиент и его отрицательные значения представлены одинаковыми числами. Другими словами, стрелка градиента и одна противоположная ей противоположность на 180 градусов считаются одинаковыми. Эмпирически было показано, что без знаковых градиентов работают лучше, чем подписанные градиенты для обнаружения пешеходов.

Следующий шаг - создать гистограмму градиентов в этих ячейках 8 × 8. Гистограмма содержит 9 блоков, соответствующих углам 0, 20, 40 ... 160. На рис. 3.6 и рис.3.7 показан процесс. Смотрим на величину и направление градиента того же пятна 8 × 8, что и на предыдущем рисунке. Буфер выбирается в зависимости от направления, и голос (значение, которое идет в корзину) выбирается на основе величины. Давайте сначала сосредоточимся на пикселе, окруженном синим. Он имеет угол (направление) 80 градусов и величину 2. Таким образом, он добавляет 2 к 5-му бункеру. Градиент в пикселе, окруженном красным, имеет угол 10 градусов и величину 4. Поскольку 10 градусов находится на полпути между 0 и 20, голосование по пикселю равномерно распределяется по двум ячейкам, что наглядно показано на рис. 3.7.

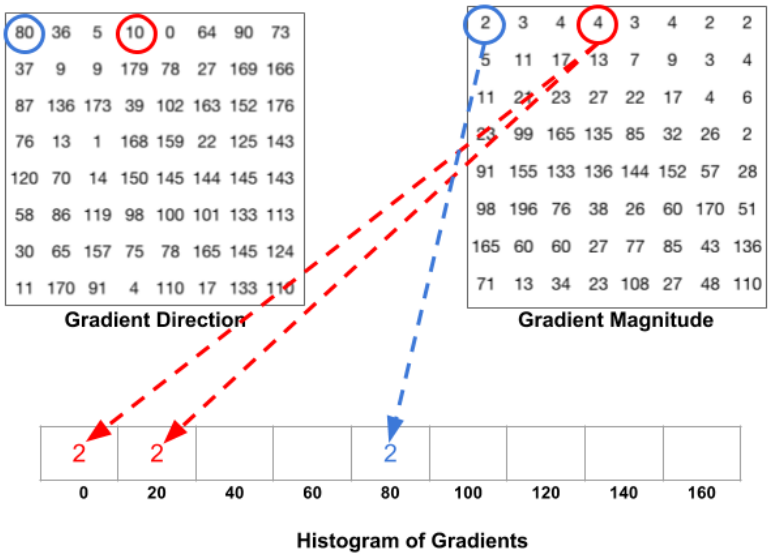

Рис. 3.7. Создание гистограммы градиентов

Есть еще одна деталь, о которой нужно знать. Если угол больше 160 градусов, он составляет от 160 до 180, а угловые обертывания составляют 0 и 180 эквивалентов. Таким образом, в приведенном ниже примере пиксель с углом 165 градусов вносит вклад пропорционально лотку на 0 градусов и лотку на 160 градусов.

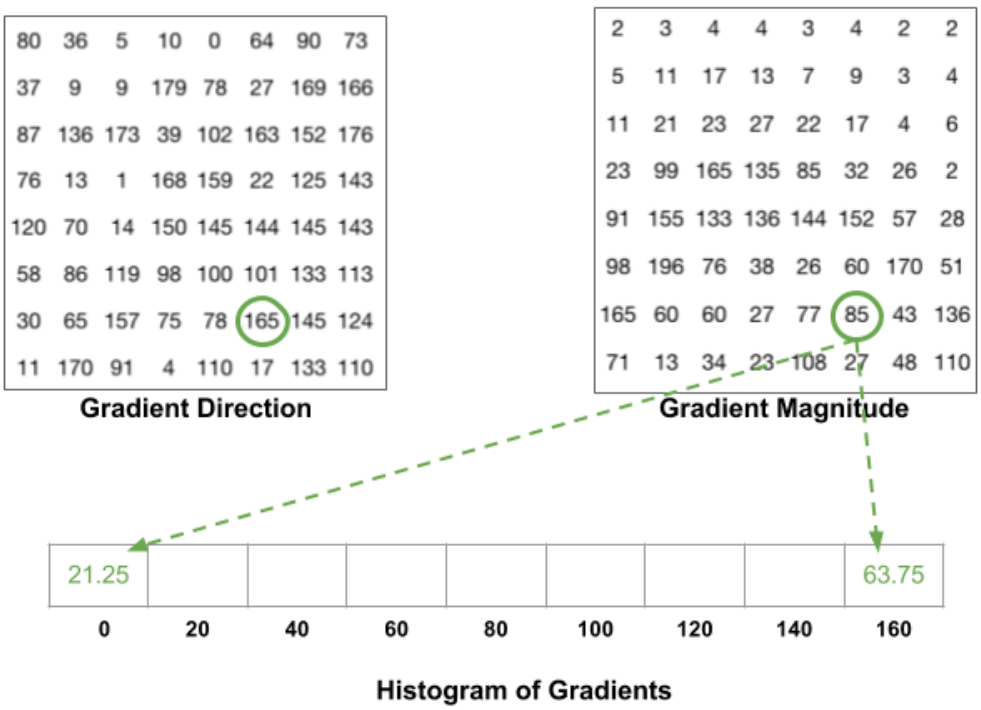

Рис. 3.8. Создание гистограммы градиентов

Вклады всех пикселей в ячейки 8 × 8 складываются для создания 9-битной гистограммы. Для патча выше это выглядит как представлено на рис.3.9:

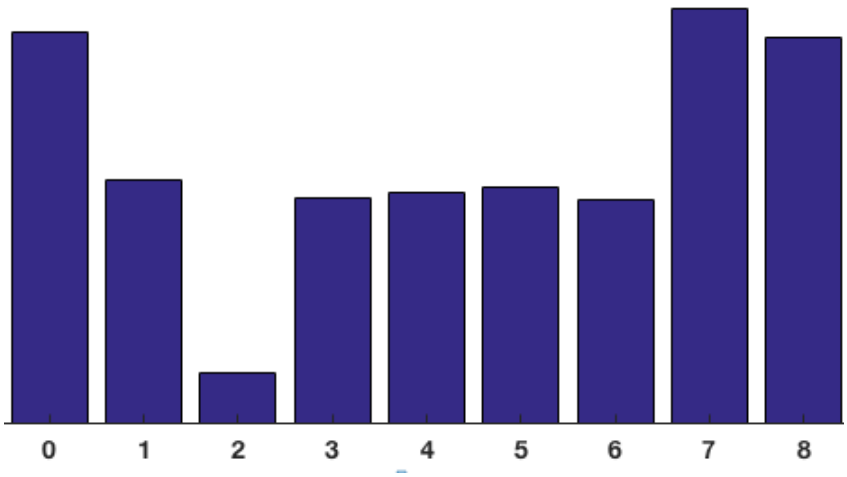

Рис. 3.9. Битная гистограмма

В данном представлении ось *y* равна 0 градусам. Можно увидеть, что гистограмма имеет большой вес около 0 и 180 градусов, тем самым показывая, что в градиентах патча указываются либо вверх, либо вниз.

Следующий шаг – это нормализация блока. На предыдущем шаге создали гистограмму, основанную на градиенте изображения. Градиенты

изображения чувствительны к общему освещению. Если сделаем изображение более темным, разделив все значения пикселей на 2, величина градиента изменится на половину, и поэтому значения гистограммы будут изменяться наполовину. В идеале необходимо, чтобы дескриптор не зависел от изменений освещения. Другими словами, нужно «нормализовать» гистограмму, чтобы на них не влияли изменения освещения.

Прежде чем разобрать, как нормализуется гистограмма, посмотрим, как нормализован вектор длины 3.

Предположим, у есть вектор цвета RGB [128, 64, 32]. Длина этого вектора равна $\sqrt{(128^2 + 64^2 + 32^2)} = 146{,}64$. Это также называется нормой L2 вектора. Разделение каждого элемента этого вектора на 146,64 дает нормированный вектор [0,87, 0,43, 0,22]. Теперь рассмотрим другой вектор, в котором элементы в два раза превышают значение первого вектора 2 x [128, 64, 32] = [256, 128, 64]. Можем самостоятельно разобраться, что нормализация [256, 128, 64] приведет к [0,87, 0,43, 0,22], что совпадает с нормализованной версией исходного RGB-вектора. Можно увидеть, что нормализация вектора удаляет масштаб.

Теперь, когда знаем, как нормализовать вектор, приступим к нормализациии более крупного блока размером 16 × 16. Блок 16 × 16 имеет 4 гистограммы, которые могут быть объединены, чтобы сформировать вектор элемента 36 x 1, и его можно нормализовать только так, как нормализуется вектор 3 × 1. Затем окно перемещается на 8 пикселей, и в этом окне вычисляется нормализованный вектор 36 × 1, и процесс повторяется.

На заключительном шаге, чтобы вычислить конечный вектор признаков для всего патча изображения, векторы 36 × 1 объединены в один гигантский вектор. Вычислим теперь размер этого вектора.

Сколько позиций блоков 16 × 16 у нас есть? Есть 7 горизонтальных и 15 вертикальных положений, составляющих в общей сложности 7 x 15 = 105 позиций.

Каждый блок 16 × 16 представлен вектором 36 × 1. Поэтому, когда объединяем их всех в один вектор gaint, получаем вектор размером 36 × 105 = 3780.

Дескриптор HOG патча изображения обычно визуализируется путем построения 9 × 1 нормализованных гистограмм в ячейках 8 × 8? Как представлено на рис. 3.10.

Рис. 3.10. Построение нормализированных гистограмм

Можем заметить, что доминирующее направление гистограммы отражает форму человека, особенно вокруг туловища и ног.

3.3 Структура автоматизированной системы

Всю автоматизированную систему прогнозирования можно разделить на три блока:

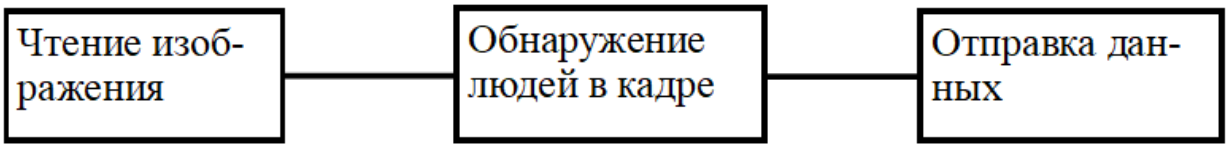

Рис.3.11. Условная схема работы программы

Теперь подробнее о каждом этапе. Для вызова модели обнаружения людей в OpenCV используется cv2.HOGDescriptor_getDefaultPeopleDetector().

Следующая функция - detector(). Здесь и происходит вся «магия», она получает разделение изображения RGB в трех цветовых каналов. Чтобы избежать проблем с производительностью, изменяем размер изображения с помощью imutils, а затем вызываем detectMultiScale () метод из нашего объекта HOG. Метод обнаружения-многомасштабный позволяет проанализировать изображение и узнать, присутствует ли человек на кадре, используя результат классификации из нашего SVM.

Анализ HOG будет генерировать некоторые захваченные ячейки (обнаруженные объекты), но иногда эти поля перекрываются, вызывая ложные срабатывания или ошибки обнаружения. Чтобы предотвратить эти недоразумения, будем использовать утилиту исключения non maxime из библиотеки imutils для удаления перекрывающихся прямоугольников - как показано на рис. 3.12:

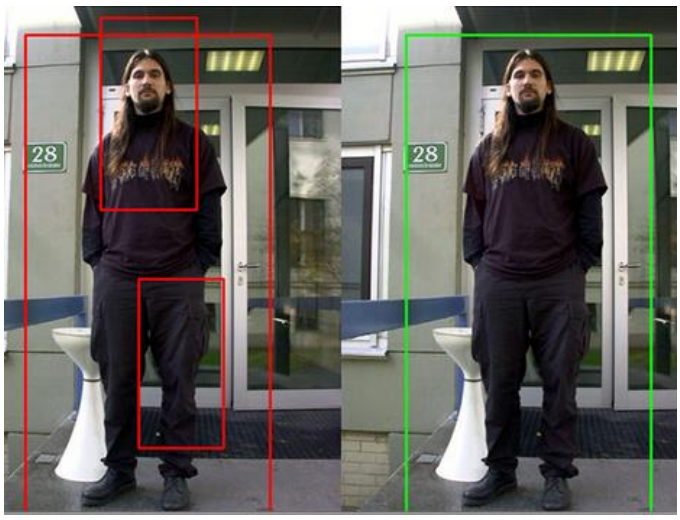

Рис. 3.12. Пример ложного срабатывания

Теперь, в этой части нашего кода, необходимо определить функцию для чтения изображения из локального файла и обнаружения любых объектов в нем. Для этого просто вызываем функцию detector () и добавляем простой цикл для рисования круглых коробок для детектора. Он возвращает количество обнаруженных участков и изображение с раскрашенным детектированием. Затем просто заново создаем результат в новом окне ОС.

Листинг 3.4. Функции чтения файлов и обнаружения объектов

```
def localDetect(image_path):
    result = []
    image = cv2.imread(image_path)
    if len(image) <= 0:
        print("[ERROR] could not read your local image")
        return result
    print("[INFO] Detecting people")
    result = detector(image)
    # shows the result
    for (xA, yA, xB, yB) in result:
        cv2.rectangle(image, (xA, yA), (xB, yB), (0, 255, 0), 2)
    cv2.imshow("result", image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
return (result, image)
```

Подобно предыдущей функции, эта функция вызовет метод detector () и поля для рисования. Изображение будет извлечено непосредственно с веб-камеры с использованием метода VideoCapture () из OpenCV. Тыакже немного изменили исходную часть кода OpenCV, чтобы получать изображения с камеры и отправить результаты на хост каждые «n» секунд. Функция convert_to_base64() преобразует наше изображение в базовую 64 строку, эта строка очень важна для просмотра наших результатов с использованием кода JavaScript внутри виджета HTML Canvas.

Листинг 3.5. Получение изображение с камеры

```
def cameraDetect(token, device, variable, sample_time=5):

    cap = cv2.VideoCapture(0)
    init = time.time()

    # Allowed sample time for host is 1 dot/second
    if sample_time < 1:
        sample_time = 1

    while(True):
        # Capture frame-by-frame
        ret, frame = cap.read()
        frame = imutils.resize(frame, width=min(400, frame.shape[1]))
        result = detector(frame.copy())

        # shows the result
        for (xA, yA, xB, yB) in result:
            cv2.rectangle(frame, (xA, yA), (xB, yB), (0, 255, 0), 2)
        cv2.imshow('frame', frame)

        # Sends results
        if time.time() - init >= sample_time:
            print("[INFO] Sending actual frame results")
            # Converts the image to base 64 and adds it to the context
            b64 = convert_to_base64(frame)
            context = {"image": b64}
            sendData(token, device, variable,
                     len(result), context=context)
            init = time.time()

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # When everything done, release the capture
    cap.release()
    cv2.destroyAllWindows()

def convert_to_base64(image):
```

```
    image = imutils.resize(image, width=400)
    img_str = cv2.imencode('.png', image)[1].tostring()
    b64 = base64.b64encode(img_str)
            return b64.decode('utf-8')
```

Конец листинга.

Метод, представленный в листинге 3.6, предназначен для получения аргументов, вставленных через терминал, и запускает процедуру, которая ищет людей в локально сохраненном файле изображения или через веб-камеру.

Листинг 3.6. Обнаружение объектов в локально сохраненном файле

```
def detectPeople(args):
    image_path = args["image"]
    camera = True if str(args["camera"]) == 'true' else False

    # Routine to read local image
    if image_path != None and not camera:
        print("[INFO] Image path provided, attempting to read image")
        (result, image) = localDetect(image_path)
        print("[INFO] sending results")
        # Converts the image to base 64 and adds it to the context
        b64 = convert_to_base64(image)
        context = {"image": b64}

        # Sends the result
        req = sendToHost(TOKEN, DEVICE, VARIABLE,
                         len(result), context=context)
        if req.status_code >= 400:
            print("[ERROR] Could not send data to host")
            return req

    # Routine to read images from webcam
    if camera:
        print("[INFO] reading camera images")
        cameraDetect(TOKEN, DEVICE, VARIABLE)
```

Конец листинга.

3.4 Тестирование приложения

В качестве примера рассмотрим кадр, представленный на рис. 3.13.

Рис. 3.13. Оживленное движение пешеходов

После того, как программа обработала этот кадр, распознанные люди были выделены зелеными прямоугольниками, как представлено на рис. 3.14, а результаты были переданы на хост, для анализирования и дальнейшего отображения в удобной пользователю форме.

Рис. 3.14. Результат обработки кадра

Для удобства работы с полученными после обработки видеопотока данными, результаты выводятся в формах, представленных на рис. 3.15.

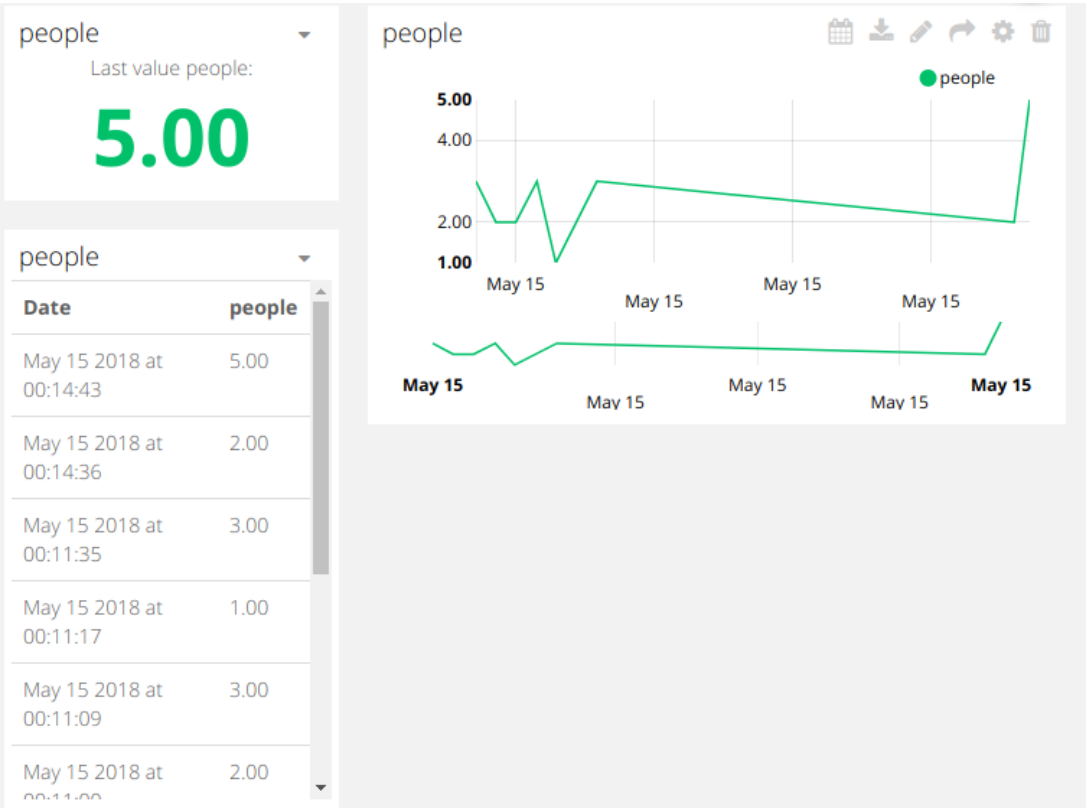

Рис. 3.15. Представление обработанной информации пользователю

Рассмотрим еще несколько вариантов, представленных на рис. 3.16 – 3.18, чтобы убедиться в правильности работы программы.

Рис. 3.16. Пример входного кадра

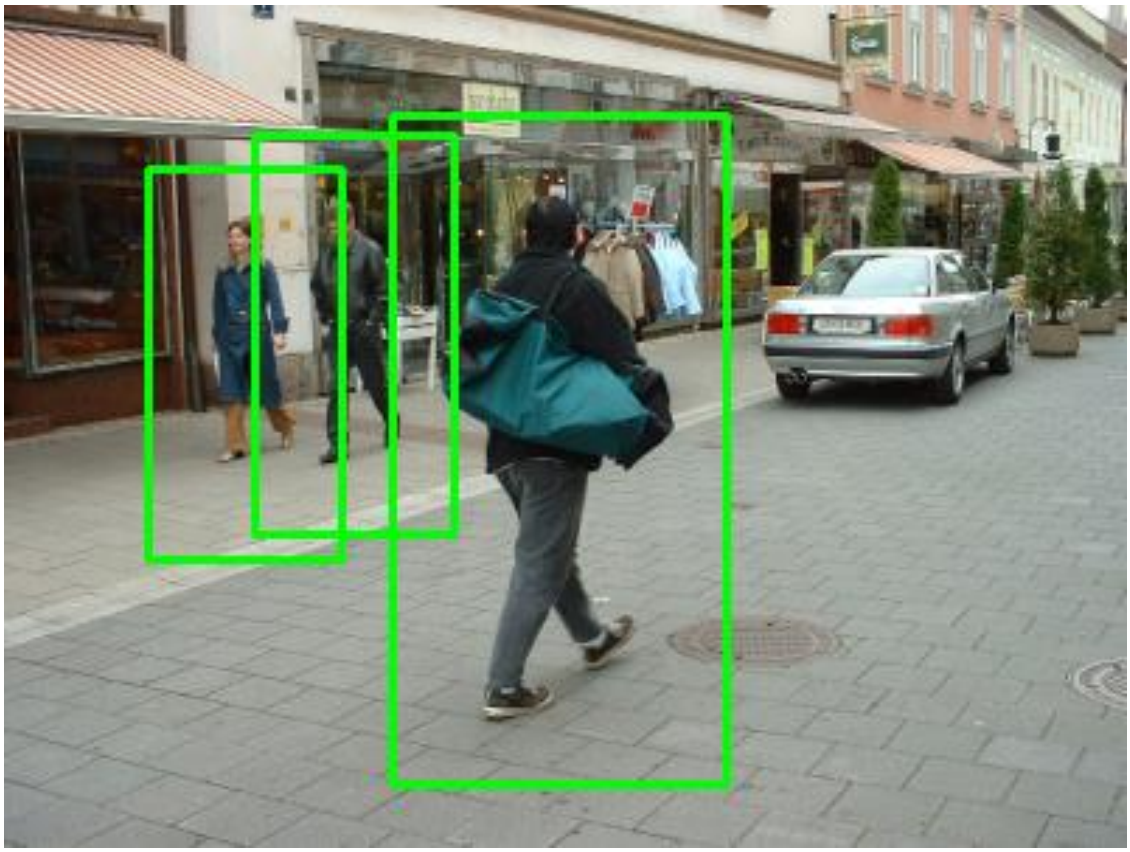

Рис. 3.17. Результат обработанного изображения

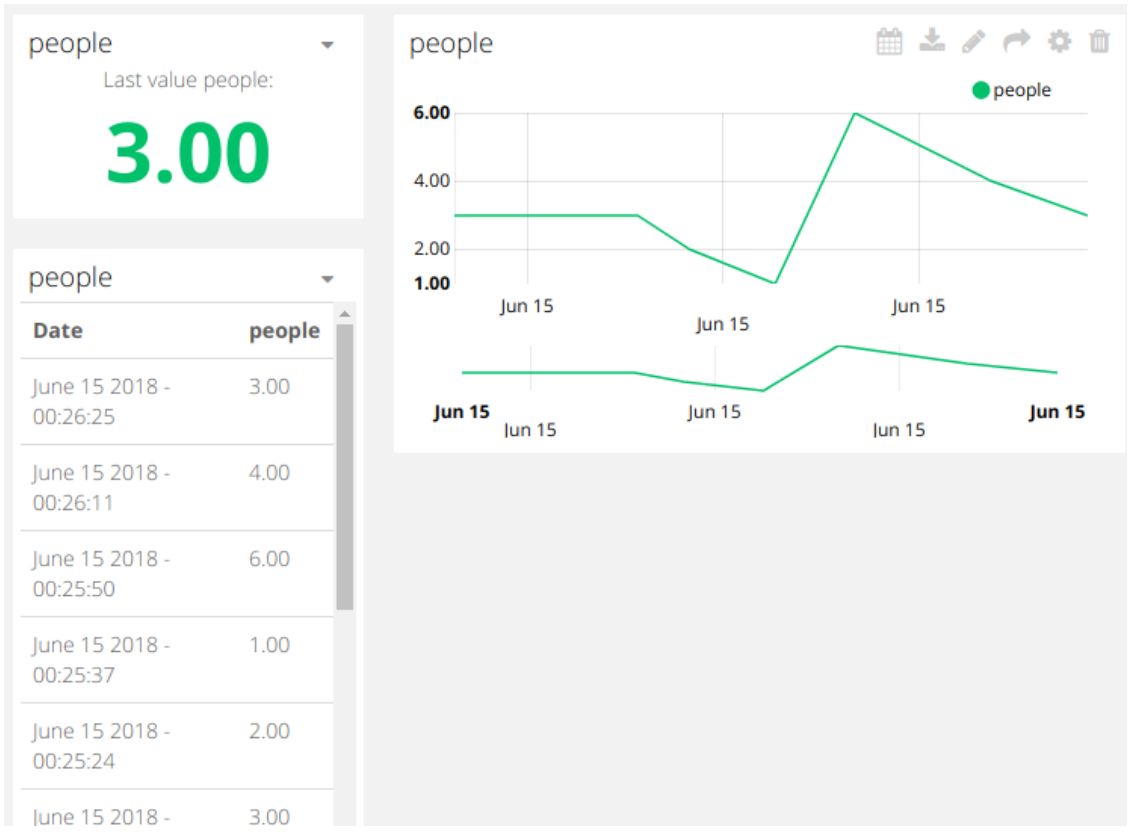

Рис. 3.18. Представление обработанной информации пользователю

ЗАКЛЮЧЕНИЕ

Для написания данной дипломной работы были рассмотрены методы распознавания объектов на изображении, а также методы их выделения и классификации.

Также, для реализации поставленных задач была решена проблема предварительной обработки потока видео для задач определения параметров движения объектов. Было определено, что для решения нашей задачи подходит выделение на изображении людей с помощью построения гистограммы ориентированных градиентов (HOG).

Полученные после обработки видео данные отправляются на хост, для дальнейшего анализа и предоставления в наглядном виде пользователю. Для этого были созданы формы с графиками, количеством распознанных людей и временем их появления в кадре.

Задачи, поставленные нами во введении, можно считать успешно выполненными. Более того, данную автоматизированную систему можно применять не только для прогнозирования числа покупателей магазина, но и в схожих ситуациях, в которых необходимо вести подсчет людей. В дальнейшем планируется дорабатывать систему до возможности распознавания других объектов, таких как автомобильный транспорт (грузовые, автобусы, легковые автомобили), животных и т.д. Это позволит более продуктивно работать с одним и тем же фрагментом видео, ввиду того, что система будет распознавать не только людей, но и другие типы объектов.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Орлов, С. Видеоаналитика: задачи и решения [Электронный ресурс] / С. Орлов // Журнал сетевых решений: электронный журнал. – 2014. – Режим доступа: http://www.osp.ru/lan/2014/06/13041879/ – (Дата обращения: 16.02.2018).

2. Сакович, И.О. Обзор основных методов контурного анализа для выделения контуров движущихся объектов [Электронный ресурс] / И.О. Сакович // Инженерный журнал: наука и инновации – электронный журнал. – 2014. – №12. – Режим доступа: http://engjournal.ru/catalog/it/hidden/1280 – (Дата обращения: 28.03.2018).

3. Тропченко, А.Ю. Методы вторичной обработки изображений и распознавания объектов [Текст] / А.Ю. Тропченко. – Санкт-Петербург: СПбГУ ИТМО, 2012. – 52 с.

4. Алгоритмы выделения контуров изображений [Электронный ресурс]. Режим доступа: https://habrahabr.ru/post/114452/. – (Дата обращения: 19.03.2018).

5. Стругайло, В.В. Обзор методов фильтрации и сегментации цифровых изображений / В.В. Стругайло// Научное издание МГТУ Им. Н.Э. Баумана «Наука и образование». – 2012.-№5. –С. 271–281.

6. Идентификация и распознавание объектов [Электронный ресурс]. Режим доступа: http://datalink.ua/news/17-01-2013-13-36-identifikaciya-iraspoznavanie-obektov/. - (Дата обращения: 19.03.2018).

7. OpenCV шаг за шагом. Обработка изображения – операторы Собеля и Лапласа [Электронный ресурс]. Режим доступа: http://robocraft.ru/blog/computervision/460.html/. – (Дата обращения: 08.04.2018).

8. Определение координат и параметров движения объекта на основе обработки изображений / Л.А. Мартынова [и др.] // Компьютерная оптика. – 2012.–№ 2. –С. 266–273.

9. Эффективность видеоаналитики и правильное размещение камер [Электронный ресурс]. http://www.aktivsb.ru/statii/effektivnost_videoanalitiki_i_pravilnoe_razmeshchenie_kamer.html. - (Дата обращения: 12.12.2016).

10. Алгоритмы выделения контуров для сегментации изображений / R. Muthukrishnan, M. Radha // International Journal of Computer Science & Information Technology. – 2011.-№3.-С. 259–267.

11. Ярышев, С.Н. Цифровые методы обработки видеоинформации и видеоаналитика[Текст] / С.Н. Ярышев – Санкт-Петербург: СПбГУ ИТМО, 2011. – 83 с.

ПРИЛОЖЕНИЕ

Листинг программы автоматического прогнозирования покупателей магазина

```
from imutils.object_detection import non_max_suppression
import numpy as np
import imutils
import cv2
import requests
import time
import argparse
import time
import base64

# Opencv pre-trained SVM with HOG people features
HOGCV = cv2.HOGDescriptor()
HOGCV.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

#1
def convert_to_base64(image):
    image = imutils.resize(image, width=400)
    img_str = cv2.imencode('.png', image)[1].tostring()
    b64 = base64.b64encode(img_str)

    return b64.decode('utf-8')


def detector(image):
    '''
    @image is a numpy array
    '''

    clone = image.copy()

    (rects, weights) = HOGCV.detectMultiScale(image, winStride=(4,
4),padding=(8, 8), scale=1.05)

    # draw the original bounding boxes
    for (x, y, w, h) in rects:
        cv2.rectangle(clone, (x, y), (x + w, y + h), (0, 0, 255), 2)

    # Applies non-max supression from imutils package to kick-off
overlapped
    # boxes
    rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
    result = non_max_suppression(rects, probs=None,
overlapThresh=0.65)

    return result


def buildPayload(variable, value, context):
    return {variable: {"value": value, "context": context}}


def sendResult(token, device, variable, value, context={},
industrial=True):
```

```
    # Builds the endpoint
    url = URL_INDUSTRIAL if industrial else URL_EDUCATIONAL
    url = "{}/api/v1.6/devices/{}".format(url, device)

    payload = buildPa      yload(variable, value, context)
    headers = {"X-Auth-Token": token, "Content-Type":
"application/json"}

    attempts = 0
    status = 400

    while status >= 400 and attempts <= 5:
        req = requests.post(url=url, headers=headers, json=payload)
        status = req.status_code
        attempts += 1
        time.sleep(1)

    return req


def argsParser():
    ap = argparse.ArgumentParser()
    ap.add_argument("-i", "--image", default=None,
                    help="path to image test file directory")
    ap.add_argument("-c", "--camera", default=False,
                    help="Set as true if you wish to use the camera")
    args = vars(ap.parse_args())

    return args


def localDetect(image_path):
    result = []
    image = cv2.imread(image_path)
    image = imutils.resize(image, width=min(400, image.shape[1]))
    clone = image.copy()
    if len(image) <= 0:
        print("[ERROR] could not read your local image")
        return result
    print("[INFO] Detecting people")
    result = detector(image)

    # shows the result
    for (xA, yA, xB, yB) in result:
        cv2.rectangle(image, (xA, yA), (xB, yB), (0, 255, 0), 2)

    cv2.imshow("result", image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    cv2.imwrite("result.png", np.hstack((clone, image)))
    return (result, image)


def cameraDetect(token, device, variable, sample_time=5):

    cap = cv2.VideoCapture(0)
    init = time.time()
```

```
    # Allowed sample time for host is 1 dot/second
    if sample_time < 1:
        sample_time = 1

    while(True):
        # Capture frame-by-frame
        ret, frame = cap.read()
        frame = imutils.resize(frame, width=min(400, frame.shape[1]))
        result = detector(frame.copy())

        # shows the result
        for (xA, yA, xB, yB) in result:
            cv2.rectangle(frame, (xA, yA), (xB, yB), (0, 255, 0), 2)
        cv2.imshow('frame', frame)

        # Sends results
        if time.time() - init >= sample_time:
            print("[INFO] Sending actual frame results")
            # Converts the image to base 64 and adds it to the context
            b64 = convert_to_base64(frame)
            context = {"image": b64}
            sendToHost(token, device, variable,
                       len(result), context=context)
            init = time.time()

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # When everything done, release the capture
    cap.release()
    cv2.destroyAllWindows()


#1

def detectPeople(args):
    image_path = args["image"]
    camera = True if str(args["camera"]) == 'true' else False

    # Routine to read local image
    if image_path != None and not camera:
        print("[INFO] Image path provided, attempting to read image")
        (result, image) = localDetect(image_path)
        print("[INFO] sending results")
        # Converts the image to base 64 and adds it to the context
        b64 = convert_to_base64(image)
        context = {"image": b64}

        # Sends the result
        req = sendResult(TOKEN, DEVICE, VARIABLE,
                         len(result), context=context)
        if req.status_code >= 400:
            print("[ERROR] Could not send data to host")
            return req

    # Routine to read images from webcam
    if camera:
```

```
        print("[INFO] reading camera images")
        cameraDetect(TOKEN, DEVICE, VARIABLE)


def main():
    args = argsParser()
    detectPeople(args)


if __name__ == '__main__':
    main()
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*