ФЕДЕРАЛЬНОЕ 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 ПРИКЛАДНОЙ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

РАЗРАБОТКА ИНФОРМАЦИОННОЙ ТЕХНОЛОГИИ РАСПОЗНАВАНИЯ ЛИЦ ПО МЕТОДУ ВИОЛЫ-ДЖОНСА

Выпускная квалификационная работа
обучающегося по направлению подготовки 09.03.03 «Прикладная информатика» заочной формы обучения, группы 07001361
Дергалева Михаила Александровича

Научный руководитель:
к. г. н., доцент
Петина М. А.

БЕЛГОРОД 2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Развитие современных компьютерных технологий происходит в экспоненциальном масштабе. Уже невозможно представить нашу жизнь без компьютера и сети Интернет. В настоящее время очень активно развивается и меняется такая научная дисциплина как компьютерное зрение. Компьютерное зрение (computer vision) — совокупность программно-технических средств, обеспечивающих считывание информации в цифровой форме видеоизображений, их обработку и выдачу результата в форме, пригодной для его практического использования в реальном масштабе времени.

На сегодняшний день нет общепринятой формулировки проблемы компьютерного зрения. А что даже еще важнее, нет стандартной формулировки того, как должна решаться проблема компьютерного зрения. Вместо этого, существует большое количество методов, позволяющих решать различные, строго определённые задачи компьютерного зрения, где методы часто зависимы от задач и редко могут быть обобщены для широкого круга применения. Многие методы и приложения все ещё находятся в стадии фундаментальных исследований, но постепенно всё большее число методов находит применение в коммерческих продуктах, где они часто составляют часть большей системы, которая может решать сложные задачи.

В большинстве практических применений компьютерного зрения компьютеры предварительно запрограммированы для решения отдельных задач, однако, методы, основанные на знаниях, становятся всё более общими.

Распознавание образов считается сферой, использующей разнообразные методы получение информации из видеопотока, и в основном, базирующиеся на статистическом подходе. Существенная часть этой области посвящена фактическому использованию этих методов.

На сегодняшний день алгоритм Виолы-Джонса является самым востребованным ввиду своей высокой скорости работы и высокой точности срабатывания.

Целью данной работы является создание программного средства распознавания лиц по методу Виолы-Джонса.

Поставленная цель достигается решением следующих задач:

1) Обзор современных методов распознавания лиц.

2) Определение недостатков существующих методов.

3) Разработка алгоритма поиска контура и распознавания лица в видеопотоке на основе метода Виолы-Джонса.

4) Программная реализация разработанных алгоритмов с применением библиотеки OpenCV.

5) Тестирование разработанной программной системы.

В качестве предмета исследования выступает процесс распознавания лиц. Объектом является метод Виолы-Джонса.

В первой главе проводится обзор основных методов распознавания лиц, рассмотрены недостатки современных методов, обоснована актуальность проблемы и описание её решения с помощью библиотеки OpenCV.

Во второй главе был разработан и описан алгоритм распознавания лиц на основе методов Виолы-Джонса.

В третьей главе был программно-реализован алгоритм распознавания лиц с помощью библиотеки OpenCV. Проведен обзор работоспособности разработанной программной системы, на основе вычислительных и визуальных экспериментов по времени вычислений и качественному показателю распознанных и не распознанных лиц.

Выпускная квалификационная работа состоит из 63 страниц и содержит в себе 19 рисунков, 20 формул, 5 таблиц, 5 приложений и 40 источников литературы.

ГЛАВА 1 ОСНОВНАЯ ЧАСТЬ

1.1 Характеристика предприятия ООО «Триатрон»

Общество с ограниченной ответственностью ООО «Триатрон» создано в соответствии с действующим законодательством РФ для извлечения прибыли в качестве основной цели своей деятельности;

Полное фирменное наименование: ООО «Триатрон» (Общество с ограниченной ответственностью ООО «КОМПАНИЯ ТРИАТРОН»);

Сокращенное наименование: ООО «Триатрон»;

Местом нахождения является место его государственной регистрации по месту нахождения его постоянно действующего исполнительного органа (директора): а адресом (местом нахождения) постоянно действующего исполнительного органа (директора) общества является адрес: 127238, г.Москва, Ильменский пр., д.5, тел.: (495) 784-70-96;

ООО «Триатрон» имеет 6 филиалов по России (Белгород, Воронеж, Орел, Тамбов, Курск, Обнинск).

Адрес в г. Белгород: 308009, г. Белгород, ул. Преображенская д.106, тел.: (4722)32-35-19;

ИНН: 7743923957;

КПП: 774301001;

ОКПО: 91492954;

ОГРН: 1117746314070;

Банк: «СДМ-Банк» (ПАО) г.Москва;

Расчетный счет: 40702810700030000548;

Корпоративный счет: 30101810845250000685;

БИК: 044525685;

ОКФС: 16 - Частная собственность

ОКОГУ: 4210014 - Организации, учрежденные юридическими лицами или гражданами, или юридическими лицами и гражданами совместно

ОКОПФ: 12165 - Общества с ограниченной ответственностью

ОКТМО: 45340000

ОКАТО: 45277577" - Москва, Административные округа г Москвы, Северный, Районы Северного административного округа, Западное Дегунино.

Исследуемое по организационно-правовой форме является обществом с ограниченной ответственностью, в отношении которого его учредитель имеет обязательные права. Размер уставного капитала составляет 15000 (пятнадцать тысяч) рублей. Общество имеет гражданские права и несет гражданские обязанности, необходимые для осуществления любых видов деятельности, не запрещенных федеральными законами, если это не противоречит предмету и целям деятельности.

Компания «Триатрон» на протяжении более 15 лет занимается поставками отечественных и импортных электронных компонентов на рынке России и стран СНГ. За это время была сформирована сеть постоянных клиентов в Москве, регионах, ближнем и дальнем зарубежье. Поставщиками являются ведущие производители электронных компонентов торговых марок - JAMICON, SAMSUNG, CCO, FENGHUA, DEGSON и многие другие.

Для быстрого обеспечения клиентов необходимыми электронными компонентами, на складе аккумулируются большие промышленные партии товаров. При тесном и долгосрочном сотрудничестве с зарубежными партнерами достигнуты короткие сроки изготовления и поставки, это позволило работать не только со склада, но и выполнять программные и срочные заказы. Прямые контакты с производителями позволили достичь самых низких цен и наилучших условий поставки. Ценовая политика фирмы ориентирована на индивидуальный подход к каждому клиенту, действует система скидок. Все производители, поставляющие в адрес ООО «Триатрон» электронные компоненты, имеют сертификаты качества ISO9002, что подтверждает и надежность, и высокие эксплуатационные характеристики

товаров. Поставляется широкий ассортимент электронных компонентов, который постоянно растет. Высокопрофессиональные сотрудники фирмы оказывают техническую и информационную консультацию для клиентов. Постоянным клиентом оказывается содействие в доставке товаров. Предлагаются различные способы доставки.

Принципы работы:

— долгосрочное и взаимовыгодное партнерство, основанное на обоюдном доверии и порядочности.

— репутация нашей компании является прямым результатом намерений и поступков всех наших сотрудников и акционеров.

— персональная ответственность за качество предлагаемых нами товаров и услуг.

Преимущества работы с компанией:

— широкий ассортимент товаров;

— оптимальные цены;

— регулярные поступления товаров на склад;

— оптимальное соотношение цена и качество товара;

— гарантия качества на все товары 12 месяцев;

— предоставление всех необходимых бухгалтерских документов, сертификатов;

— гибкая открытая система скидок;

— граница опта — минимальная партия 3000 рублей;

— доставка в регионы через транспортные компании, экспресс почтой;

— быстрое оформление и формирования заказа;

— заявки принимаются в любом виде;

— оплата в любой форме (наличный, безналичный, выписка счета, перевод на карту);

— возможность товарного кредита (действует для постоянных клиентов);

— клиенту предоставляется персональный менеджер;

— реальное наличие на складе;

— склад и офис находятся рядом;

— предоставляются образцы для тестирования.

Таким образом, была составлена общая характеристика предприятия, определены принципы работы общества с ограниченной ответственностью, обозначены преимущества работы с фирмой (как поставщиков, так и клиентов).

От уровня организационной структуры предприятия непосредственно зависит успех в достижении целей избранной стратегии. Просчеты в организационных структурах очень часто приводили даже компании к тяжелому кризису. Поэтому выбор и устроение организационной структуры в соответствии с внутренними и внешними факторами, которые определяют деятельность ООО «Триатрон», стоящие перед ним стратегические цели, является одной из наиболее важных и ответственных задач маркетинга. Каждый из существующих видов организационных структур имеет свои недостатки преимущества, которые должны обязательно учитываться.

1.2 Обоснование проблематики

Задача распознавания лиц имеет множество применений в различных областях. Среди них - интеллектуальные системы безопасности и контроля доступа, организация видеоконференций, системы машинного зрения в робототехнике, биометрия и т. п.

Техника идентификации личности, основанная на распознавании лица, отличается от техник распознавания по другим биометрическим показателям (радужная оболочка глаза, отпечаток пальца). Преимуществом здесь является то, что физический контакт и устройством не производится. Это делает данную технологию наиболее приемлемой для массового применения.

За последние годы были предложены множество алгоритмов обработки, распознавания и локализации лиц – цепи Маркова, как нейронные сети, и т. д. Все системы распознавания лиц можно условно разделить на две широкие

категории: использующие (2D) двухмерные изображения и (3D) трехмерные изображение лиц.

При использовании баз данных 2D лиц на качество распознавания влияют положение лиц на изображении и условия освещения, при этом 3D изображения разработаны для снятия этих ограничений. Получение трехмерных изображений требует наличия профессиональных устройств. Также при сканировании такие системы требуют, чтобы исследуемый объект оставался неподвижным несколько секунд. Данное требование является недопустимым для систем, требующих работы в реальном режиме времени.

Сложность задачи распознавания лиц можно обосновать следующими причинами: лицо человека – это динамический объект, имеющий высокую степень изменяемости во внешнем виде (по форме и цвету кожи); различные параметры освещения, определенные типом и направлением источника света, фрагментарное перекрытие лиц другими объектами сцены; необходимость локализации и распознавания лиц, имеющих произвольные положения в пространстве. Однако существующие системы локализации и распознавания лиц не всегда учитывают данные особенности, что не позволяет достичь приемлемого уровня распознавания на изображениях и видеопоследовательностях.

1.3 Обзор современных методов распознавания лиц

В сущности, работу любого алгоритма распознавания можно описать диаграммой на рисунке 1.1:

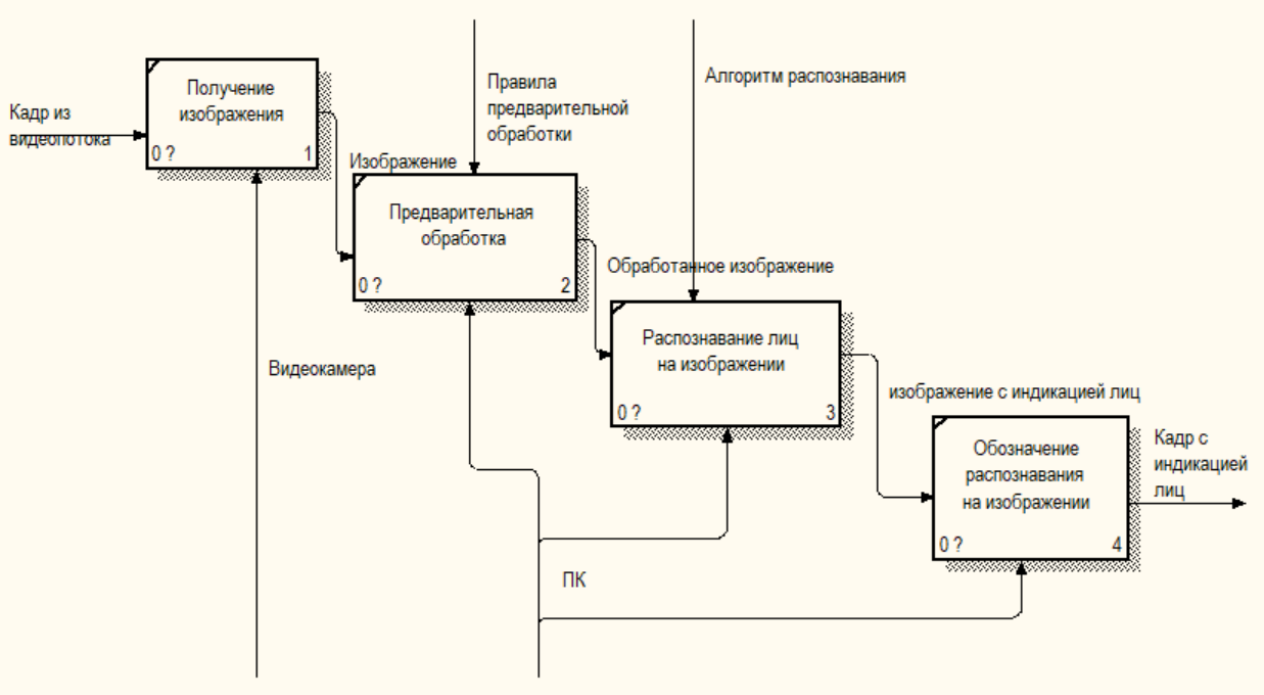

Рисунок 1.1 – Декомпозиция общей диаграммы распознавания лиц из видеопотока

Контекстная диаграмма представлен в приложении А.

Существующие алгоритмы обнаружения лиц можно разбить на четыре категории:

- эмпирический метод;
- метод характерных инвариантных признаков;
- распознавание с помощью шаблонов, заданных разработчиком;
- метод обнаружения по внешним признакам, обучающиеся системы.

Эмпирический подход базирующийся на «знаниях сверху-вниз» (knowledge based top-down methods) предполагает реализацию алгоритма, реализующего определенный набор правил, которому должен соответствовать участок изображения, чтобы можно было признать его лицом человека. Создание таких правил является попыткой формализовать эмпирические знания о том, как именно должно выглядеть лицо человека на изображениях. В итоге по ним определяется: есть лицо на участке изображения или нет. [5]

Самые простые правила:

- присутствует значительная разница в яркости между центральной частью и верхней частью лица;

- яркость и цвет центральной части лица являются однородными;

- резко отличаются по яркости относительно остальной части лица, два симметрично расположенных глаза, нос и рот

Для сглаживания помех применяется метод сильного уменьшения изображения подвергает изображение уменьшению в размерах. Это также уменьшает вычислительные операции (рисунок 1.2). Метод способствует более простому выявлению зоны равномерного распределения яркости (зона предполагаемого нахождения лица), чтобы в последующем выполнить проверку на наличие сильно отличающихся по яркости областей внутри: именно такие области можно с разной долей вероятности отнести к «лицу». [7]

Рисунок 1.2 - Метод Yang & Huang

Метод построения гистограмм пользуется вертикальной и горизонтальной гистограммами (рисунок 1.3). В областях-кандидатах происходит поиск черт лица. При получении гистограммы определенной формы можно определить вероятность наличия лиц.

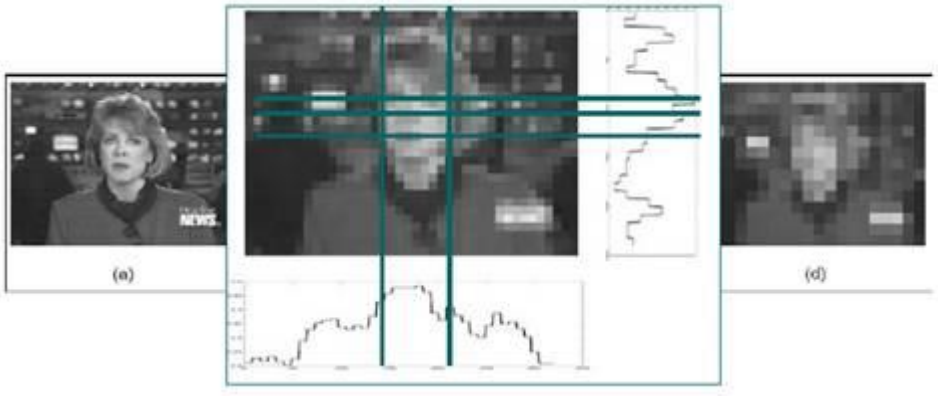

Рисунок 1.3 - Метод Kotropoulos & Pita

Данный подход использовался на заре развития компьютерного зрения ввиду малых требований к вычислительной мощности процессора для обработки изображения.

Описанные методы имеют неплохие показатели по определению лиц на изображениях на однородном фоне. Их легко реализовать с помощью машинного кода, что позволило разработать множество подобных алгоритмов. Недостатком является их абсолютная непригодность при наличии сложного заднего фона и чувствительность к наклону и повороту головы. [8]

Методы характерных инвариантных признаков, базирующиеся на знаниях снизу-вверх (Feature invariant approaches) образуют вторую группу методов определения объектов. Здесь виден другой подход к проблеме: не происходит формализации протекающих в человеческом мозге процессов в явном виде. Сторонники этого подхода стараются найти инвариантные особенности, выявить неявные закономерности и свойства объектов, независимо от угла наклона и положения.

Основные этапы алгоритмов этой группы методов:

- обнаружение: границы лица, форма, яркость, текстура, цвет;

- детектирование на изображении явных признаков лица: глаз, носа, рта;

- объединение всех найденных инвариантных признаков и их верификация.

В сложных сценах предполагается поиск правильных геометрический расположения форм лица. Для этого применяется гауссовский производный фильтр с множеством различных масштабов и ориентаций. Следом случайным перебором выполняется поиск соответствия выявленных черт лица, их взаимное расположение. [17]

Суть метода группировки признаков с применением второй производной гауссовского фильтра для поиска интересующих областей изображения (рисунок 1.4). После этого группируются края вокруг каждой такой области при помощи порогового фильтра.

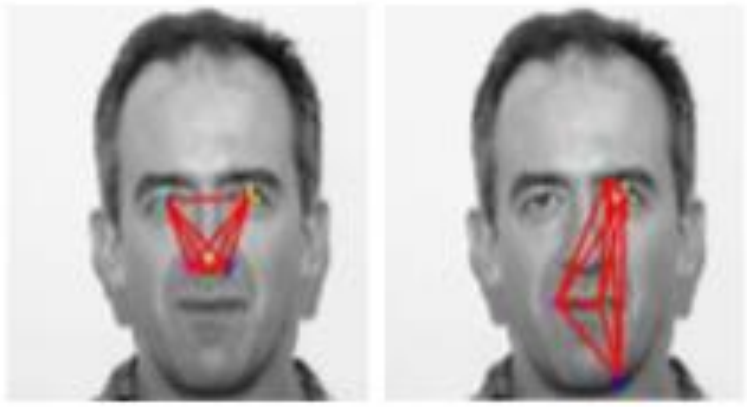

Рисунок 1.4 - Верные и случайные срабатывания

Для комбинирования найденных признаков используется оценка при помощи байесовской сети. Происходит выборка черт лица. В этой группе методов есть ряд недостатков. Большое влияние имеет сложный задний фон изображения, при котором могут возникать проблемы с обнаружением. При небольшом загромождении лица другими объектами, засветке или возникновении шумов процент достоверного распознавания также сильно падает. Основа рассмотренных подходов — эмпирика, является одновременно их сильной и слабой стороной. В данном случае обнаружение объектов на изображении относится к задачам высокой сложности из-за следующих факторов: большая изменчивость объекта распознавания, зависимость от освещения, условий съемки.

Применение эмпирических правил позволяет свести задачу распознаванию объектов на изображении до определенного количества относительно простых проверок. Однако данные методы первой категории пока очень далеки по эффективности от уже давно успешно функционирующего инструмента – человеческого зрения, поскольку исследователи, решившие встать на этот, сталкиваются с рядом серьезных трудностей. Во-первых, процессы, протекающие в человеческом мозге далеко не полностью изучены, и набор эмпирических знаний, который на данный момент доступны на сознательном уровне, далеко не исчерпывает весь спектр подсознательных

инструментов. Во-вторых, невозможно перенести неформальный человеческий опыт в набор определенных правил, т.к. в ряде случаев это может привести к большому количеству ложных срабатываний, либо наоборот – обнаружение вообще не произойдет. [9]

Распознавание с помощью шаблонов, заданных разработчиком (Template Matching Methods). Шаблоны задают некий стандартный образ изображения лица, например, путем описания свойств отдельных областей лица и их возможного взаимного расположения. Обнаружение лица с помощью шаблона заключается в проверке каждой из областей изображения на соответствие заданному шаблону. Особенности подхода:

- два вида шаблонов:
  а) недеформируемые;
  б) деформируемые;
- шаблоны заранее запрограммированы, необучаемы;
- используется корреляция для нахождения лица на изображении.

Метод детектирования лица при помощи трехмерных форм предполагает использование шаблонов в виде пар отношений яркостей в двух областях. Для детекции лица требуется просканировать все изображение на сравнение с заданным шаблоном. Причём делать это необходимо с различным масштабом (рисунок 1.5).

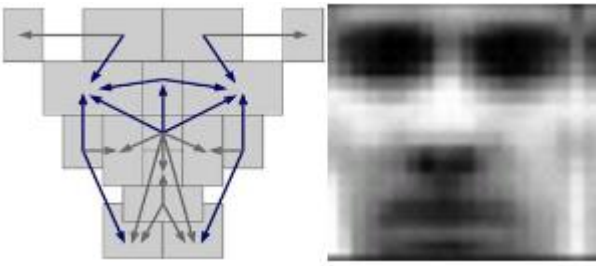

Рисунок 1.5 - Метод детектирования лица при помощи трехмерных форм

Модели распределения опорных точек являются статистическими моделями, которые представляют объекты, форма которых может измениться. Их полезная особенность для метода — способность выделить форму

переменных объектов в пределах учебного набора с небольшим количеством параметров формы. Эта компактная и точная параметризация может использоваться для разработки эффективных систем классификации.

Среди достоинств распознавания с помощью шаблонов можно выделить относительную простоту реализации и хорошие результаты срабатывания для изображений с несложным задним фоном. Основной недостаток этого метода – необходимость калибровки шаблона вблизи с изображением лица. Целесообразность метода не считается высокой из-за сложности вычисления шаблонов для различных поворотов лица и ракурсов.

Обычно поиск лиц на изображениях с помощью методов, основанных на построении математической модели изображения лица, заключается в полном переборе всех прямоугольных фрагментов изображения всевозможных размеров и проведения проверки каждого из фрагментов на наличие лица. Поскольку схема полного перебора обладает такими безусловными недостатками, как избыточность и большая вычислительная сложность, авторами применяются различные методы сокращения количества рассматриваемых фрагментов. [7]

Основные принципы методов:

- Схоластика: каждый сканируется окном и представляется векторами ценности.
- Блочная структура: изображение разбивается на пересекающиеся или непересекающиеся участки (рисунок 1.6) различных масштабов и производится оценка с помощью алгоритмов оценки весов векторов.

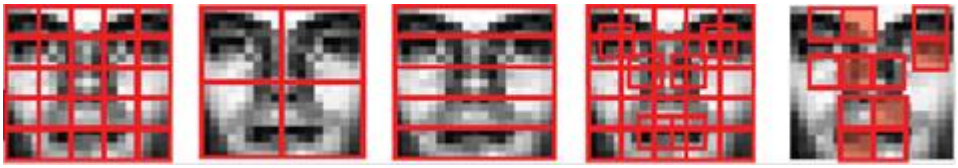

Рисунок1.6 - Примеры разбиения изображения на участки

Для обучения алгоритмов требуется библиотека вручную подготовленных изображений лиц и «не лиц», любых других изображений.

Стоит отметить что важнейшей задачей является выделить сильные классификаторы. Именно они будут иметь наивысший приоритет для проверки найденных признаков в изображении. Количество же более слабых классификаторов стоит уменьшать за счёт похожести друг на друга, а также удалении классификаторов, возникших за счёт шумовых выбросов.

Перечислим основные методики выполнения этих задач:

- Искусственные нейронные сети (Neural network: Multilayer Perceptions);
- Метод главных компонент (Principal Component Analysis (PCA));
- Сравнение шаблонов (Template Matching).
- Скрытые Марковские модели (Hidden Markov model);
- Метод гибкого сравнения на графах (Elastic graph matching)
- Совмещение ФА и метода главных компонент (Mixture of PCA, Mixture of factor analyzers);
- Разреженная сеть окон (Sparse network of windows (SNoW));
- Активные модели (Active Appearance Models);
- Адаптированное улучшение и основанный на нём Метод Виолы-Джонса и др.

В методе Виолы-Джонса используется алгоритм сканирования окна. Выглядит он следующим образом: на исследуемом изображении устанавливается окно сканирование в начальное положение, выбраны используемые признаки. Окно последовательно двигается по изображению с шагом в 1 ячейку (допустим, размер самого окна есть 24*24 ячейки).

Производится последовательное сканирование для различных масштабов. Масштабируется само сканирующее окно. Все найденные признаки попадают к классификатору, который «выносит вердикт». Вычислять все признаки на маломощных ПК просто нереально, поэтому классификатор должен реагировать на строго определенный набор признаков. Совершенно логично, что надо обучить классификатор по данному набору. Это производится автоматически. [18]

1.4 Недостатки существующих систем

Недостатки методов характерных инвариантных признаков: при несущественном перекрытии лица другими предметами, появлении шумов или засветке процент корректного распознавания существенно снижается. Также сильное влияние оказывает сложный задний фон.

Недостатки методов распознавание с помощью шаблонов: существенным недостатком является то, что требуется откалибровать шаблон вблизи с изображением лица. Целесообразность использования также под сомнением т.к. необходимо вычислять шаблоны для различных ракурсов и поворотов лица.

Недостатки нейронных сетей: при добавлении нового эталонного лица в БД требуется полное переобучение сети для всего имеющегося набора (процедура в зависимости от размера выборки может достигать нескольких дней). Проблемы обучения математического характера: попадание в локальный оптимум, выбор оптимального шага оптимизации, переобучение и т. д. Трудно формализуемый этап выбора архитектуры сети (количество нейронов, слоев, характер связей). Исходя из выше изложенного, можно заключить, что нейронная сеть – «черный ящик» с трудно интерпретируемыми результатами работы.

Недостатки методов главных компонентов (Principal Component Analysis, PCA): метод требует для своего применения идеальных условий таких как: строго единые параметры освещенности, нейтральное выражение лица, отсутствие помех на лице вроде очков и бород.

Недостатки метода сравнения шаблонов (Template Matching): недостатком является то, что требуется много ресурсов как для сравнения участков изображений, так и для хранения. Используется простейший алгоритм сравнений. Это накладывает ограничения на исходные изображения - должны быть сняты в строго установленных условиях. Не допустимы заметные изменения ракурса, освещения, эмоционального выражения. [21]

Недостатки скрытых Марковских моделей (CMM, HMM):

- необходимость подбирать параметры модели для каждой отдельной базы данных;
- не обладает различающей способностью, то есть алгоритм обучения только максимизирует отклик каждого изображения на свою модель, но не минимизирует отклик на другие модели.

Недостатки метода гибкого сравнения на графах (Elastic graph matching):

- высокая вычислительная сложность процедуры распознавания;
- низкая технологичность при запоминании новых эталонов;
- линейная зависимость времени работы от размера базы данных лиц.

В итоге среди основных недостатков существующих систем можно выделить следующие:

1) влияние окружающей среды и света;
2) потребление большого количества ресурсов;
3) проблематичное обслуживание;
4) необходимость эталонного вида распознаваемого объекта;
5) медленная скорость распознавания.

1.5 Библиотека OpenCV

OpenCV (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков. Может свободно использоваться в академических и коммерческих целях — распространяется в условиях BSD. [3]

Применяется для утверждения общего стандартного интерфейса компьютерного зрения для приложений в этой области. Для способствования росту числа таких приложений и создания новых моделей использования PC. Также чтобы сделать платформы Intel привлекательными для разработчиков

таких приложений за счёт дополнительного ускорения OpenCV с помощью Intel Performance Libraries (Сейчас включают IPP (низкоуровневые библиотеки для обработки сигналов, изображений, а также медиа-кодеки) и MKL (специальная версия LAPACK и FFTPack)). OpenCV способна автоматически обнаруживать присутствие IPP и MKL и использовать их для ускорения обработки. [4]

В версии 2.2 библиотека была реорганизована. Вместо универсальных модулей cxcore, cvaux, highGUI и других было создано несколько компактных модулей с более узкой специализацией.

Основные модули:

- opencv_core — основная функциональность. Включает в себя базовые структуры, вычисления (математические функции, генераторы случайных чисел) и линейную алгебру, DFT, DCT, ввод/вывод для XML и YAML и т. д.;
- opencv_imgproc — обработка изображений (фильтрация, геометрические преобразования, преобразование цветовых пространств и т. д.);
- opencv_highgui — простой UI, ввод/вывод изображений и видео;
- opencv_ml — модели машинного обучения (SVM, деревья решений, обучение со стимулированием и т. д.);
- opencv_features2d — распознавание и описание плоских примитивов (SURF (англ.)русск., FAST и другие, включая специализированный фреймворк);
- opencv_video — анализ движения и отслеживание объектов (оптический поток, шаблоны движения, устранение фона);
- opencv_objdetect — обнаружение объектов на изображении (нахождение лиц с помощью алгоритма Виолы-Джонса (англ.), распознавание людей HOG и т. д.);
- opencv_calib3d — калибровка камеры, поиск стерео-соответствия и элементы обработки трёхмерных данных;
- opencv_flann — библиотека быстрого поиска ближайших соседей (FLANN 1.5) и обертки OpenCV;

- opencv_contrib — сопутствующий код, ещё не готовый для применения;
- opencv_legacy — устаревший код, сохранённый ради обратной совместимости;
- opencv_gpu — ускорение некоторых функций OpenCV за счет CUDA, создан при поддержке NVidia. [24]

Библиотека довольно популярна, на текущий момент имеет чуть более 5 миллионов скачиваний (не считая загрузок напрямую из репозитория), и недавно была предложена в качестве основы для стандарта Khronos по компьютерному зрению. [33]

ГЛАВА 2 РАЗРАБОТКА АГЛОРИТМА РАСПОЗНАВАНИЯ ЛИЦ НА ОСНОВЕ МЕТОДА ВИОЛЫ-ДЖОНСА

2.1 Основные принципы

Метод был разработан в 2001 году Полом Виолой и Майклом Джонсом и до сих пор является основополагающим для поиска объектов на изображении в реальном времени.

Основные принципы, на которых основан метод:

- использование изображений в интегральном представлении, что позволяет вычислять быстро необходимые объекты;
- используются признаки Хаара, по которым происходит поиск нужного объекта (в данном случае, лица и его черт);
- используется бустинг (от англ. boost – улучшение, усиление) для выбора наиболее подходящих признаков для искомого объекта на данной части изображения;
- все признаки, поступающие на вход классификатора, дают результат «верно» либо «ложь»;
- используются каскады признаков для быстрого отбрасывания окон, где не найдено лицо.

Результаты поиска очень быстры, хотя самообучение классификаторов происходит крайне медленно. Поэтому этот метод был выбран для распознавания объектов на изображении. Метод Виолы-Джонса на сегодняшний день является одним из лучших по соотношению показателей эффективность распознавания/скорость работы. Также детектор имеет очень низкую вероятность ложного срабатывания. Алгоритм хорошо выполняет работу и распознает черты лица даже под небольшим углом, примерно до 30 градусов. Если угол наклона имеет большее значение, процент обнаружений

резко падает. В стандартной реализации при отсутствии нужного классификатора это не позволяет детектировать повернутое лицо человека под произвольным углом, что сильно затрудняет использование этого алгоритма в производственных системах с учетом растущих потребностей. Требуется подробный разбор принципов, на которых основан алгоритм Виолы-Джонса. В общем виде данный метод ищет лица и черты лица по общему принципу сканирующего окна. [34]

2.2 Принцип сканирующего окна

Принцип сканирующего окна в общем виде производит обнаружение лица на изображении следующим образом:

1) Есть изображение с искомыми объектами. Оно представлено двумерной матрицей пикселей w*h, где каждый имеет значение:

- от 0 до 255, если это черно-белое изображение;
- от 0 до 255^3, если это цветное изображение (палитра R, G, B).

2) В качестве результата своей работы, алгоритм должен определить черты лиц и пометить их. Поиск производится в активной области изображения с помощью прямоугольных признаков Хаара, которые и описывают найденное лицо и его черты:

$$rectangle_i = \{x,y,w,h,a\}, \quad (2.1)$$

где x, y – координаты центра i-го прямоугольника;

w – ширина;

h – высота;

a – угол наклона прямоугольника к вертикальной оси изображения.

Иначе говоря, применительно к рисункам и фотографиям используется метод сканирующего окна (scanning window): каждый фрагмент изображения сканируется окном. При каждом перемещении окна для каждого положения применяются классификаторы. Система обучения классификаторов полностью автоматизирована и не требует вмешательства человека, поэтому этот подход работает быстро. Задача поиска и нахождения лиц на изображении с помощью

данного метода часто оказывается очередным шагом на пути к распознаванию характерных черт, к примеру, верификации человека по распознанному лицу или распознавания мимики лица. [34]

2.3 Интегральное представление изображений

Для выполнения каких-либо действий с данными используется интегральное представление изображений по методу Виолы-Джонса. Также такое представление можно встретить и в других методах, таких как SURF, вейвлетах-преобразования и многих других. Интегральное представление быстро позволяет рассчитать общую яркость произвольного прямоугольника на данном изображении. Причем время расчета всегда константное.

Интегральное представление изображения – это матрица, которая совпадает по размерам с исходным изображением. Каждый ее элемент хранит сумму интенсивностей всех пикселей, находящихся левее и выше данного элемента. Элементы матрицы рассчитываются по следующей формуле:

$$L(x, y) = \sum_{i=0,j=0}^{i\le x,j\le y} * I(i, j), \quad (2.2)$$

где I(i,j) — яркость пикселя исходного изображения.

Каждый элемент матрицы L представляет собой сумму пикселей в прямоугольнике от (0,0) до (x,y), т.е. значение каждого пикселя (x,y) равно сумме значений всех пикселов левее и выше данного пикселя (x,y). Расчет матрицы выполняется линейное время, пропорциональное числу пикселей в изображении, поэтому интегральное изображение просчитывается за один проход.

Расчет матрицы возможен по формуле 2.3:

$$L(x,y) = I(x,y) – L(x-1,y-1) + L(x,y-1) + L(x-1,y). \quad (2.3)$$

По такой интегральной матрице можно быстро вычислить сумму пикселей произвольного прямоугольника и произвольной площади.

Пусть в прямоугольнике ABCD (рисунок 2.1) есть интересующий нас объект D:

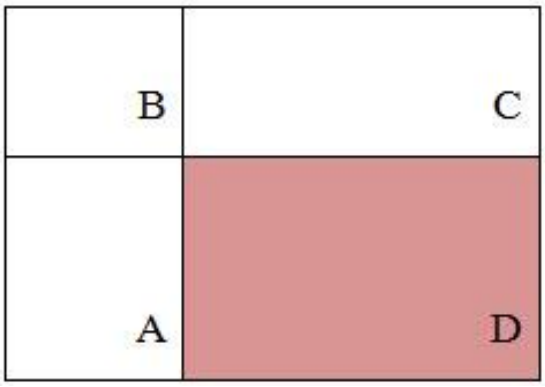

Рисунок 2.1 – Прямоугольник ABCD

Из рисунка понятно, что сумму внутри прямоугольника можно выразить через суммы и разности смежных прямоугольников по формуле 2.4:

$$S(ABCD) = L(A) + L(C) — L(B) — L(D). \quad (2.4)$$

Эта формула используется для вычисления суммы пикселей прямоугольника.

2.4 Признаки Хаара

Признак — отображение f: X => D_f, где D_f — множество допустимых значений признака. Если заданы признаки $f_1,\ldots,f_n$, тогда вектор признаков x = $(f_1(x),\ldots,f_n(x))$ называется признаковым описанием объекта x∈X. Признаковые описания допустимо отождествлять с самими объектами. При этом множество X = D_{f1}* …* D_{fn} называют признаковым пространством. [1]

Признаки делятся на следующие типы в зависимости от множества D_f:

- бинарный признак, D_f = {0,1};
- номинальный признак: D_f — конечное множество;
- порядковый признак: D_f — конечное упорядоченное множество;
- количественный признак: D_f — множество действительных чисел.

Естественно, бывают прикладные задачи с разнотипными признаками, для их решения подходят далеко не все методы.
В стандартном методе Виолы – Джонса используются прямоугольные признаки, они называются примитивами Хаара.

В расширенном методе Виолы – Джонса, использующемся в библиотеке OpenCV используются дополнительные признаки (приложение Б).

Вычисляемым значением этих признаков будет F = X-Y, где X – сумма значений яркостей точек закрываемых светлой частью признака, а Y – сумма значений яркостей точек закрываемых темной частью признака. Для их вычисления используется понятие интегрального изображения, рассмотренное выше. Признаки Хаара дают точечное значение перепада яркости по оси X и Y соответственно. [33]

2.5 Сканирование окна

Алгоритм сканирующего окна с признаками Хаара выглядит следующим образом:

- имеется исследуемое изображение, окно сканирования на начальной позиции, используемые признаки установлены;
- на каждом шаге окно сканирование перемещается по изображению с шагом в 1 пиксель (например, размер окна может быть 28*28 пикселей);
- при сканировании области в окне вычисляются около 200 тыс. различных вариантов расположения признаков за счет изменения положения и масштаба в окне сканирования;
- сканирование осуществляется последовательно при различных масштабах;
- само изображение не масштабируется, делает это сканирующее окно (изменяется размер ячейки);
- все признаки, которые были найдены на участке изображения, передаются классификатору, который выносит решение.

Рисунок 2.2 - Визуализация алгоритма сканирующего окна

Процесс поиска поочередно производит вычисление всех признаков, чего просто невозможно достичь на обычных домашних компьютерах. Следовательно, классификатору нужно реагировать на строго определенное множество признаков. Выглядит логичным, что для нахождения лиц классификатор нужно научить определять признаки, характерные только для них. Это достигается автоматическим обучением. [35]

2.6 Модель машинного обучения

Машинное обучение – процесс получения автоматическим модулем новых знаний. Есть признанное определение данному процессу: машинное обучение — это наука, изучающая компьютерные алгоритмы, автоматически улучшающиеся во время работы. Рисунок 2.3 иллюстрирует процесс обучения машины.

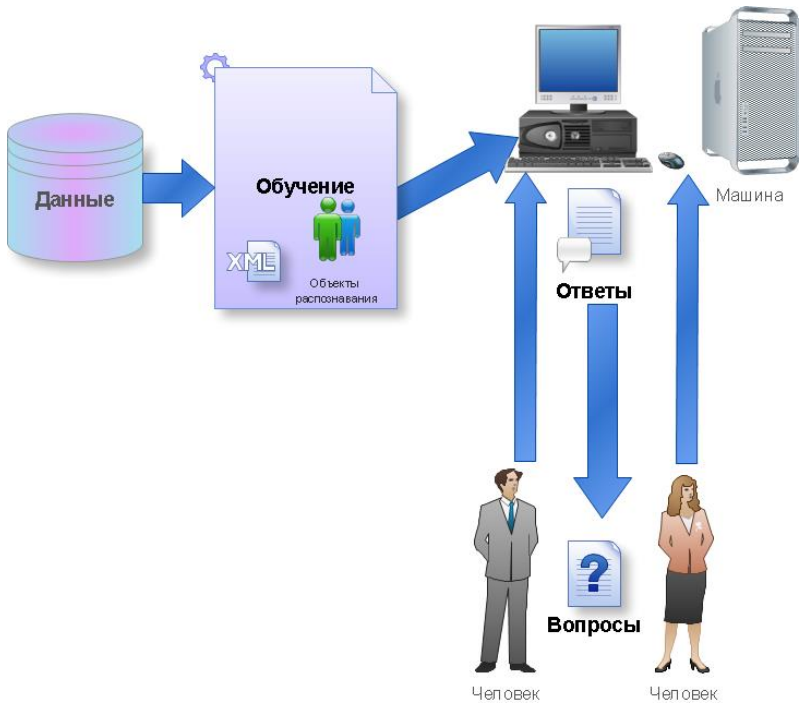

Рисунок 2.3 – Модель машинного обучения

Данный процесс входит в концепцию и технологию под названием Data mining (извлечение информации и интеллектуальный анализ данных), куда входят помимо машинного обучения такие дисциплины, как «Теория баз данных», «Искусственный интеллект», «Алгоритмизация», «Распознавание образов и прочие».

Машинное обучение в методе Виолы-Джонса решает такую задачу как классификация путем обучения каскадного классификатора. [37]

2.7 Обучение классификатора в методе Виолы-Джонса

В контексте алгоритма, имеется множество изображений, которые разделены на классы. Они задаются множеством изображений для определенного класса, к которому они относятся (например, это может быть класс «фронтальное положение глаз»). Такое множество называется обучающей выборкой. Для других объектов не определена классовая принадлежность. Требуется построить алгоритм, который способен классифицировать произвольный объект из исходного множества.

Классифицировать объект — значит, указать номер (или наименование класса), к которому относится данный объект.

Классификация объекта — номер или наименование класса, выдаваемые алгоритмом классификации в результате его применения к данному конкретному объекту.

Классификатор (classifier) — в задачах классификации — это аппроксимирующая функция, выносящая решение, к какому именно классу данный объект принадлежит.

Обучающая выборка – конечное число данных.

В машинном обучении задача классификации относится к разделу обучения с учителем, когда классы поделены. Распознавание образов по сути своей и есть классификация изображений и сигналов. В случае алгоритма Виолы-Джонса для идентификации и распознавания лица классификация является двухклассовой.

Постановка классификации выглядит следующим образом: есть X – множество, в котором хранится описание объектов, Y – конечное множество номеров, принадлежащих классам. Между ними есть зависимость – отображение Y*: X => Y. Обучающая выборка представлена формулой:

$$X_m = \{(x_1,y_1), \ldots, (x_m,y_m)\}. \quad (2.5)$$

Конструируется функция f от вектора признаков X, которая выдает ответ для любого возможного наблюдения X и способна классифицировать объект $x \in X$. Данное простое правило должно хорошо работать и на новых данных. [33]

2.8 Применяемый в алгоритме бустинг

Для решения проблемы сложного обучения существует технология бустинга.

Бустинг — комплекс методов, способствующих повышению точности аналитических моделей. Эффективная модель, допускающая мало ошибок классификации, называется «сильной». «Слабая» же, напротив, не позволяет надежно разделять классы или давать точные предсказания, делает в работе

большое количество ошибок. Поэтому бустинг (от англ. boosting – повышение, усиление, улучшение) означает дословно «усиление» «слабых» моделей – это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов.

Бустинг представляет собой жадный алгоритм построения композиции алгоритмов (greedy algorithm) — это алгоритм, который на каждом шагу делает локально наилучший выбор в надежде, что итоговое решение будет оптимальным. Бустинг над решающими деревьями считается одним из наиболее эффективных методов с точки зрения качества классификации. Во многих экспериментах наблюдалось практически неограниченное уменьшение частоты ошибок на независимой тестовой выборке по мере наращивания композиции. Более того, качество на тестовой выборке часто продолжало улучшаться даже после достижения безошибочного распознавания всей обучающей выборки. Это перевернуло существовавшие долгое время представления о том, что для повышения обобщающей способности необходимо ограничивать сложность алгоритмов. На примере бустинга стало понятно, что хорошим качеством могут обладать сколь угодно сложные композиции, если их правильно настраивать. [5]

Объяснение математического бустинга звучит следующим образом: наряду с множествами X и Y вводится вспомогательное множество R, называемое пространством оценок. Рассматриваются алгоритмы, имеющие вид суперпозиции $a(x) = C(b(x))$, где функция $b: X \to R$ называется алгоритмическим оператором, функция $C: R \to Y$ –решающим правилом.

Многие алгоритмы классификации имеют именно такую структуру: сначала вычисляются оценки принадлежности объекта классам, затем решающее правило переводит эти оценки в номер класса. Значение оценки, как правило, характеризует степень уверенности классификации.

Алгоритмическая композиция – алгоритм $a: X \to Y$ вида

$$a(x) = C(F(b_1(x), \ldots, b_T(x)), x \in X, \qquad (2.6)$$

составленный из алгоритмических операторов b_t :X→R, t=1,..., T, корректирующей операции F: R^T→R и решающего правила C: R→Y. Базовыми алгоритмами обозначаются функции $a_t(x) = C(b_t(x))$, а при фиксированном решающем правиле C — и сами операторы $b_t(x)$.

Суперпозиции вида $F(b_1,..., b_T)$ являются отображениями из X в R, то есть, опять же, алгоритмическими операторами.

В задачах классификации на два непересекающихся класса в качестве пространства оценок обычно используется множество действительных чисел. Решающие правила могут иметь настраиваемые параметры. Так, в алгоритме Виолы-Джонса используется пороговое решающее правило, где, как правило, сначала строится оператор при нулевом значении, а затем подбирается значение оптимальное. Процесс последовательного обучения базовых алгоритмов применяется, пожалуй, чаще всего при построении композиций. [35]

Критерии останова могут использоваться различные, в зависимости от специфики задачи, возможно также совместное применение нескольких критериев:

- построено заданное количество базовых алгоритмов T;
- достигнута заданная точность на обучающей выборке;
- достигнутую точность на контрольной выборке не удаётся улучшить на протяжении последних нескольких шагов при определенном параметре алгоритма.

Развитием данного подхода явилась разработка более совершенного семейства алгоритмов бустинга AdaBoost (adaptive boosting – адаптированное улучшение), предложенная Йоавом Фройндом (Freund) и Робертом Шапиром (Schapire) в 1999 году [9], который может использовать произвольное число классификаторов и производить обучение на одном наборе примеров, поочередно применяя их на различных шагах.

Рассматривается задача классификации на два класса, Y = {−1,+1}. К примеру, базовые алгоритмы также возвращают только два ответа −1 и +1, и

решающее правило фиксировано: C(b) = sign(b). Искомая алгоритмическая композиция имеет вид:

$$a(x) = C(F(b_1(x), \dots, b_T(x)) = \text{sign}(\sum_{t=1}^{T} a_1 b_1(x)),\ x \in X. \quad (2.7)$$

Функционал качества композиции Q_t определяется как число ошибок, допускаемых ею на обучающей выборке:

$$Q(b, W^1) = Q_T = \sum_{t=1}^{1} [y_i \sum_{t=1}^{T} a_t b_t(x_i)] < 0, \quad (2.8)$$

где $W^l = (w_1, \dots, w_l)$ – вектор весов объектов.

Визуализация алгоритма представлена в приложении Б.

Плюсы AdaBoost:

- хорошая обобщающая способность. В реальных задачах практически всегда строятся композиции, превосходящие по качеству базовые алгоритмы. Обобщающая способность может улучшаться по мере увеличения числа базовых алгоритмов;

- простота реализации;

- собственные накладные расходы бустинга невелики. Время построения композиции практически полностью определяется временем обучения базовых алгоритмов;

- возможность идентифицировать объекты, являющиеся шумовыми выбросами. Это наиболее «трудные» объекты x_i, для которых в процессе наращивания композиции веса w_i принимают наибольшие значения.

Минусы AdaBoost:

- бывает переобучение при наличии значительного уровня шума в данных. Экспоненциальная функция потерь слишком сильно увеличивает веса «наиболее трудных» объектов, на которых ошибаются многие базовые алгоритмы. Однако именно эти объекты чаще всего оказываются шумовыми выбросами. В результате AdaBoost начинает настраиваться на шум, что ведёт к переобучению. Проблема решается путём удаления выбросов или применения менее «агрессивных» функций потерь. В частности, применяется алгоритм GentleBoost;

- AdaBoost требует достаточно длинных обучающих выборок. Другие методы линейной коррекции, в частности, бэггинг, способны строить алгоритмы сопоставимого качества по меньшим выборкам данных;

- бывает построение неоптимального набора базовых алгоритмов. Для улучшения композиции можно периодически возвращаться к ранее построенным алгоритмам и обучать их заново;

- бустинг может приводить к построению громоздких композиций, состоящих из сотен алгоритмов. Такие композиции исключают возможность содержательной интерпретации, требуют больших объёмов памяти для хранения базовых алгоритмов и существенных временных затрат на вычисление классификаций.

В наши дни подход усиления простых классификаторов является популярным и, вероятно, наиболее эффективным методом классификации за счёт высокой скорости и эффективности работы и относительной простоты реализации.

2.9 Каскадная модель разрабатываемого алгоритма

Алгоритм бустинга для поиска лиц таков:

1) Определение слабых классификаторов по прямоугольным признакам.
2) Для каждого перемещения сканирующего окна вычисляется прямоугольный признак на каждом примере.
3) Выбирается наиболее подходящий порог для каждого признака.
4) Отбираются лучшие признаки и лучший подходящий порог.
5) Перевзвешивается выборка.

Каскадная модель сильных классификаторов – это по сути то же дерево принятия решений, где каждый узел дерева построен таким образом, чтобы детектировать почти все интересующие образы и отклонять регионы, не являющиеся образами. Помимо этого, узлы дерева размещены таким образом, что чем ближе узел находится к корню дерева, тем из меньшего количества примитивов он состоит и тем самым требует меньшего времени на принятие

решения. Данный вид каскадной модели хорошо подходит для обработки изображений, на которых общее количество детектируемых образов мало. В этом случае метод может быстрее принять решение о том, что данный регион не содержит образ, и перейти к следующему. Пример каскадной модели сильных классификаторов представлен на рисунке 2.4:

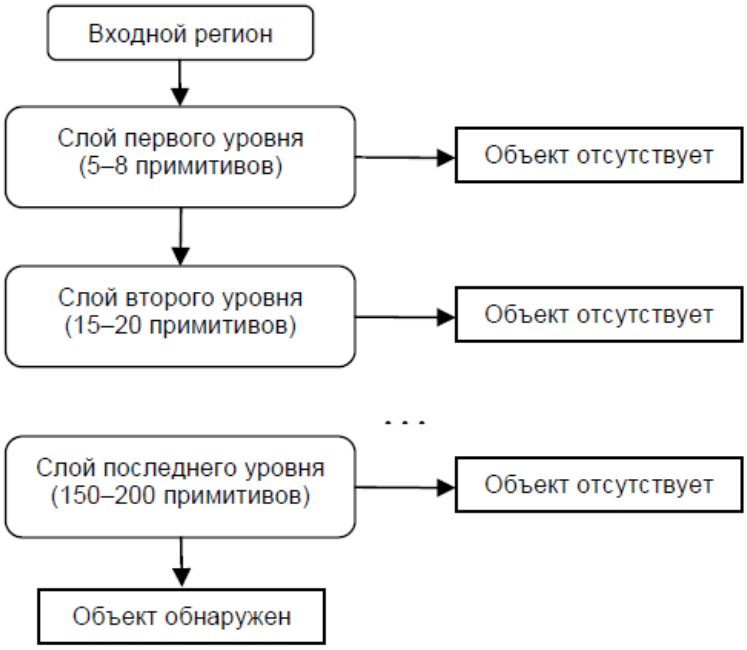

Рисунок 2.4 – Пример каскадной модели сильных классификаторов

Далее, каскад применяется к изображению:

1) Работа с «простыми» классификаторами – при этом отбрасывается часть «отрицательных» окон.

2) Положительное значение первого классификатора запускает второй, более приспособленный и так далее.

3) Отрицательное значение классификатора на любом этапе приводит к немедленному переходу к следующему сканирующему окну, старое окно отбрасывается.

4) Цепочка классификаторов становится более сложной, поэтому ошибок становится намного меньше.

Для тренировки такого каскада потребуются следующие действия:

1) задаются значения уровня ошибок для каждого этапа (предварительно их надо количественно просмотреть при применении к изображению из обучающего набора) – они называются detection и false positive rates – надо чтобы уровень detection был высок, а уровень false positive rates низок;

2) добавляются признаки до тех пор, пока параметры вычисляемого этапа не достигнут поставленного уровня, тут возможны такие вспомогательные этапы, как:

а. Тестирование дополнительного маленького тренировочного набора.

б. Порог AdaBoost умышленно понижается с целью найти больше объектов, но в связи с этим возможно большее число неточных определений объектов.

3) если false positive rates остается высоким, то добавляется следующий этап или слой;

4) ложные обнаружения в текущем этапе используются как отрицательные уже на следующем слое или этапе.

В более формальном виде алгоритм тренировки каскада задан ниже:

1) Пользователь задает значения f (максимально допустимый уровень ложных срабатываний на слой) и d (минимально допустимый уровень обнаружений на слой).

2) Пользователь задает целевой общий уровень ложных срабатываний $F_{target.}$

3) P – набор положительных примеров.

4) N – набор отрицательных примеров.

5) $F_0 = 1,0$; $D_0 = 1,0$; $i = 0$.

6) while ($F_i > F_{target}$)

$i = i+1$; $n_i = 0$; $F_i = F_{i-1}$

while ($F_i = f * F_{i-1}$)

$n_i = n_i + 1$

AdaBoost(P, N, n_i).

7) Оценить полученный каскад на тестовом наборе для определения F_i и D_i;

8) Уменьшать порог для i-того классификатора, пока текущий каскад будет иметь уровень обнаружения по крайней мере $d*D_i$-1 (это же касается F_i)

9) N = Ø;

10) Если $F_i > F_{target}$, то оценить текущий каскад на наборе изображений, не содержащих лиц, и добавить все неправильно классифицированные в N.[32]

2.10 Итоговое представление алгоритма

Распознавание лиц из видеопотока по методу Виолы-Джонса представлено в виде контекстной диаграммы (приложение А) и диаграммой декомпозиции (рисунок 2.5)

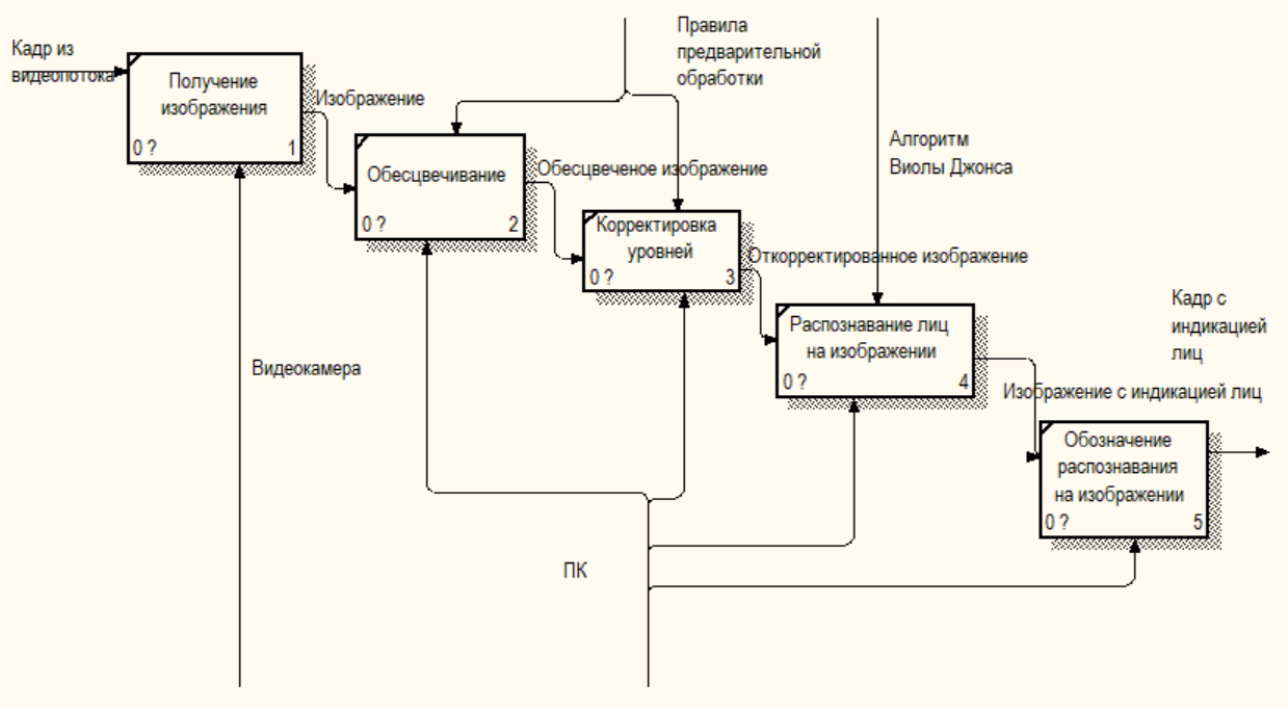

Рисунок 2.5 – Диаграмма декомпозиции распознавания лиц из видеопотока по методу Виолы-Джонса

На диаграмме видно, что после получения изображения оно проходит предварительную обработку – «Обесцвечивание» и «Корректировка уровней».

Следом идет непосредственное распознавание методом Виолы-Джонса и вывод изображения с наложенными фигурами индикации лиц.

В конечном итоге процесс распознавания лиц в видеопотоке будет выглядеть следующим образом (рисунок 2.6):

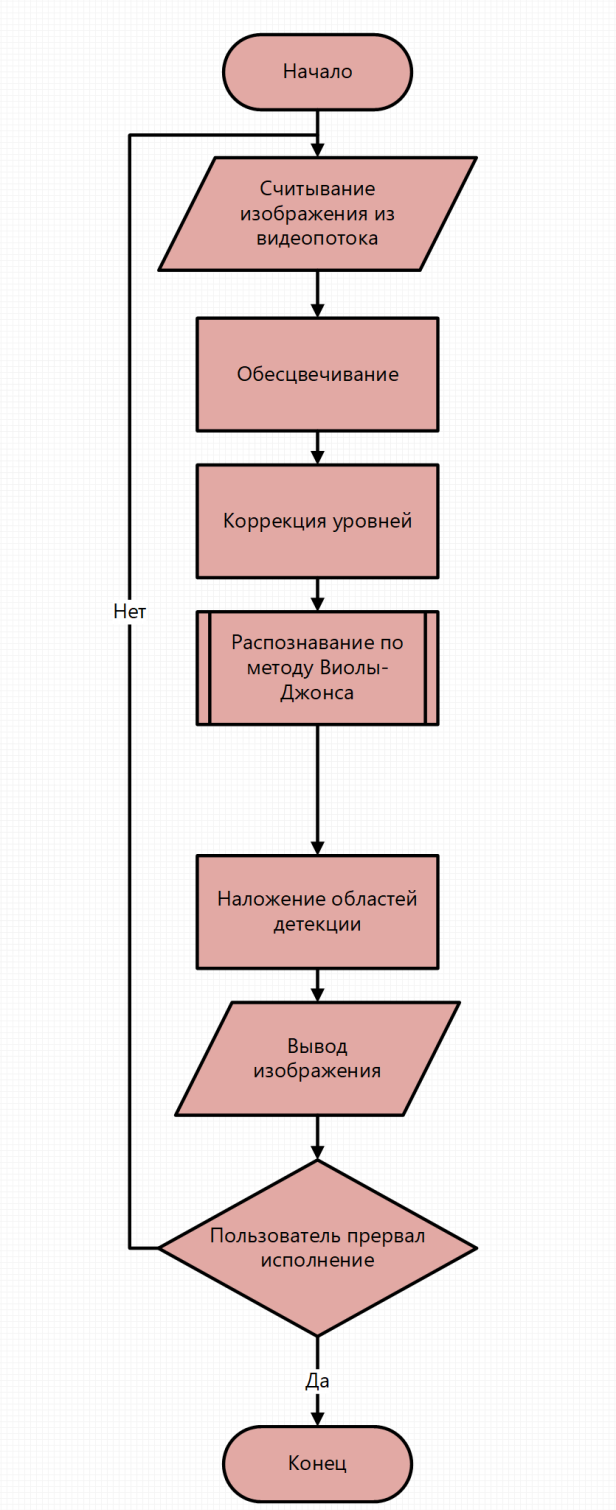

Рисунок 2.6 – Блок-схема алгоритма распознавания лиц в видеопотоке

Каждый раз при получении изображения из видеопотока его нужно привести в надлежащий вид для возможности распознавания. Происходит обесцвечивание изображения, т.к. цвет не нужен при распознавании. Далее корректируются световые уровни изображения, чтобы на распознавание в меньшей степени влияла освещенность и нахождение источника света. Ключевой блок распознавания по методу Виолы-Джонса выделен на блок-схеме подпрограммой. Если алгоритм нашел лица, то на выходе имеются координаты этих лиц на изображении, и мы можем отметить их. Это изображение показываем пользователю и считываем следующий кадр из видеопотока. На рисунке 2.7 представлено описание алгоритма по методу Виолы-Джонса.

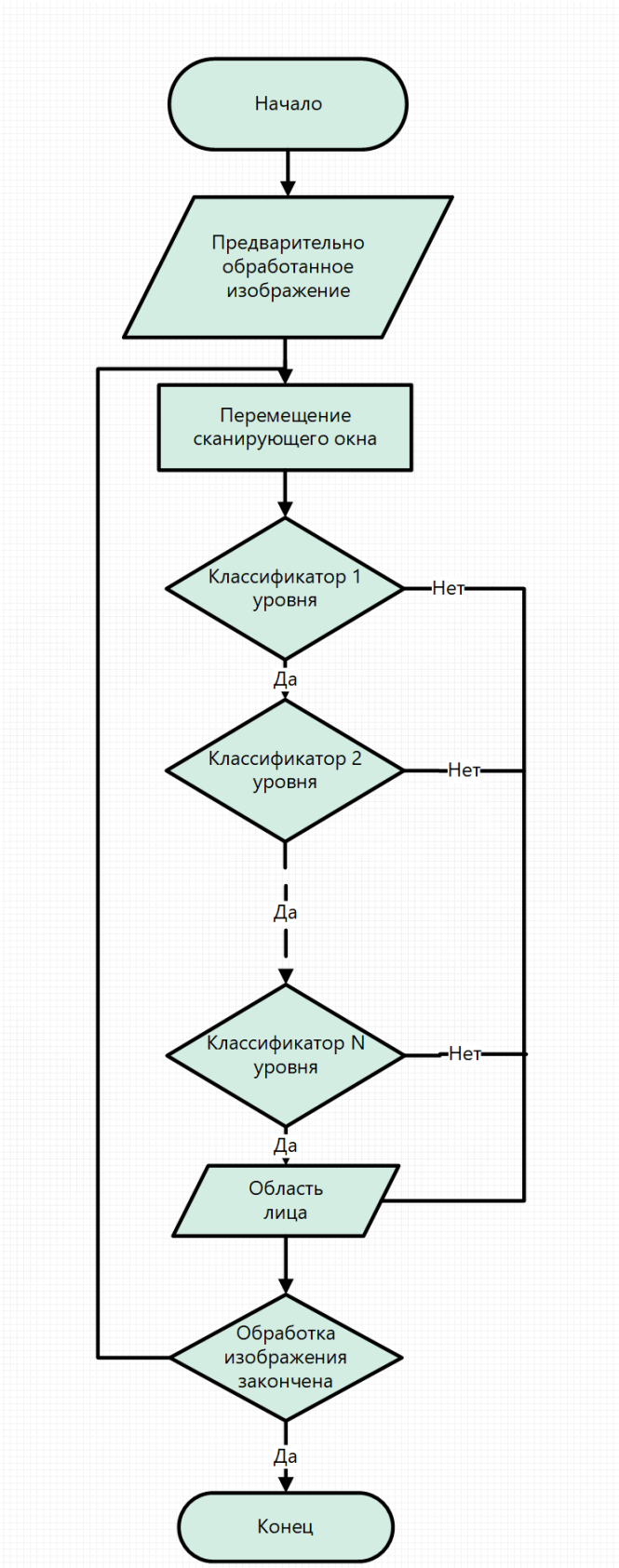

Рисунок 2.7 – Раснознавание по методу Виолы-Джонса.

Сканирующее окно при каждом своем проходе по изображению применяет каскадный классификатор. На каждом последующем уровне проверяются признаки классификатора. Если признаки подходят, то проверяется следующий классификатор. После успешного выполнения последнего классификатора программа считает, что нашла лицо. Если любой из классификаторов возвращает отрицательное значение выполнения, то программа с уверенностью считает, что нужный объект не найден и сканирующее окно смещается дальше.

ГЛАВА 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ РАСПОЗНАВАНИЯ ЛИЦ С ПРИМЕНЕНИЕМ БИБЛИОТЕКИ OPENCV

3.1 Основной модуль

Для подготовки работы распознавания требуется иметь натренированные классификаторы.

Для тренировки классификатора требуются большие наборы изображений – позитивных и негативных. Позитивные представляют собой набор искомых объектов. Негативные являются фоном. Важно, чтобы изображения были одинакового размера, т.к. происходит склейка и указываются те координаты, где находится объект. Проще говоря, те координаты, на которых обучается классификатор. При склейке получается новый набор изображений. Примеры можно взять как из камеры, так и из общедоступных баз данных.

Чтобы начать обучение, нужно иметь папки «Good» - с позитивными изображениями и «Bad» с негативными. Файлы описания для положительных и отрицательных изображений будут иметь разную структуру. У отрицательных он проще – это просто список относительных путей к изображениям. Для положительных запись сложнее. Кроме пути также должно быть указано положение объекта и размер. В целом, каждое положительное изображение может содержать несколько примеров объектов. Лучше всего на один кадр иметь один объект. " Good\17.bmp " — адрес объекта относительно файла описания. «1» — количество положительных объектов на изображении. «212126 254» — координаты прямоугольника на изображении, в котором находится объект. Если объектов несколько, то запись приобретает вид: «Good \17.bmp 3200 300 70 70 400 400 55 65 500 500 45 87» (рисунок 3.1).

Рисунок 3.1 - Пример снимков положительной выборки

Теперь создадим итоговый каскад. Для этих целей будем использовать программы opencv_traincascade.exe и opencv_createsamples.exe. Обучение занимает очень много времени. К примеру, для обучения каскада на 500-1000 изображений процесс займет почти целый день. Создание этих образов осуществляется с помощью createsamplks.cpp, который в свою очередь подгружает для работы cvhaartrainig.h. Эти файлы имеются в стандартной библиотеке OpenCV.

Далее представлены параметры запуска программы.

```
void createTrainingSamples(
    const char* filename, // название выходного файла, с правильными наборами изображений
    const char* imgfilename, //изображение с искомым объектом
    int bgcolor, // цвет изображения
int bgthreshold,// порог цветности, или прозрачности 8-битового изображения
    const char* bgfilename, //изображение с фоном
    int count,
```

int invert = 10,//поворот изображения с объектом в градусах

int maxintensitydev = 30,//максимальное отклонение интенсивности пикселей образцов с исходным элементом

double maxxangle =1.2,
//максимальный угол поворота в радианах по X

double maxyangle=1.2,
//максимальный угол поворота в радианах по Y

double maxzangle =0.6, //максимальный угол поворота в радианах по Z

int showsamples = 0,1//если не ноль, то каждый сделанный образец будет показан

int winwidth = 32,//такая ширина в пикселях будет у конечного образца

int winheight = 32//такая высота в пикселях будет у конечного образца

);

Параметры и стадии обучения классификатора

void createCascadeClassifier(

const char* dirname,//имя директории, в которой создается и хранится классификатор

const char* vecfilename,//имя vec-файла с изображениями объекта если такой есть

const char* bgfilename, //изображение с файлом, описывающим фон

int npos,//количество позитивных образцов

int nneg, //количество негативных образцов

int nstages,//количество этапов или стадий обучения

int numprecalculated,//количество признаков, которые должны быть прегенерированы

int numsplits,//количество слабых классификаторов, используемых на каждой стадии обучения: 1 – отросток, 2 и более - деревья

float minhitrate = 0.996F,//минимальный желаемый коэффициент успеха на каждой стадии обучения классификатора

float maxfalsealarm = 0.6F,//максимальный желаемый результат ложных обнаружений на каждой стадии

float weightfraction = 0.96F,//указывает параметр используемого веса, для хорошо работающего классификатора наиболее оптимальным будет параметр 90

int mode = 3,//используемый алгоритм, устанавливает набор типов признаков Хаара, использующихся во время обучения.

int equalweights = 1,1//если параметр не равен 0, это означает что начальные веса всех образцов будут одинаковыми

int winwidth = 32,//размеры образцов

int winheight = 32,//размеры образцов

intboosttype = 3,//выбор типа алгоритма бустинга, где 0 – DiscreteAdaBoost, 1 – RealAdaBoost, 2 – LogitBoost, 3 – GentleAdaBoost
int stumperror = 0);//тип указываемой ошибки, если применяется алгоритм Discrete AdaBoost. [3]

Результатом данного этапа являются готовые для использования, натренированные классификаторы. Как только классификаторы будут готовы, их можно будет использовать в нашем приложении.

Распознавание хоть и срабатывает почти мгновенно, все же дает ощутимые задержки в плавности если последовательно производить вычисления на каждом кадре изображения, полученного от камеры, а затем отображать результат.

Данное приложение производит распознавание в параллельном потоке и отдает лишь области детекции если таковые имеются. Задержек между получениями кадров с камеры не происходит.

Наглядно схема работы данного механизма показана на рисунке 3.2.

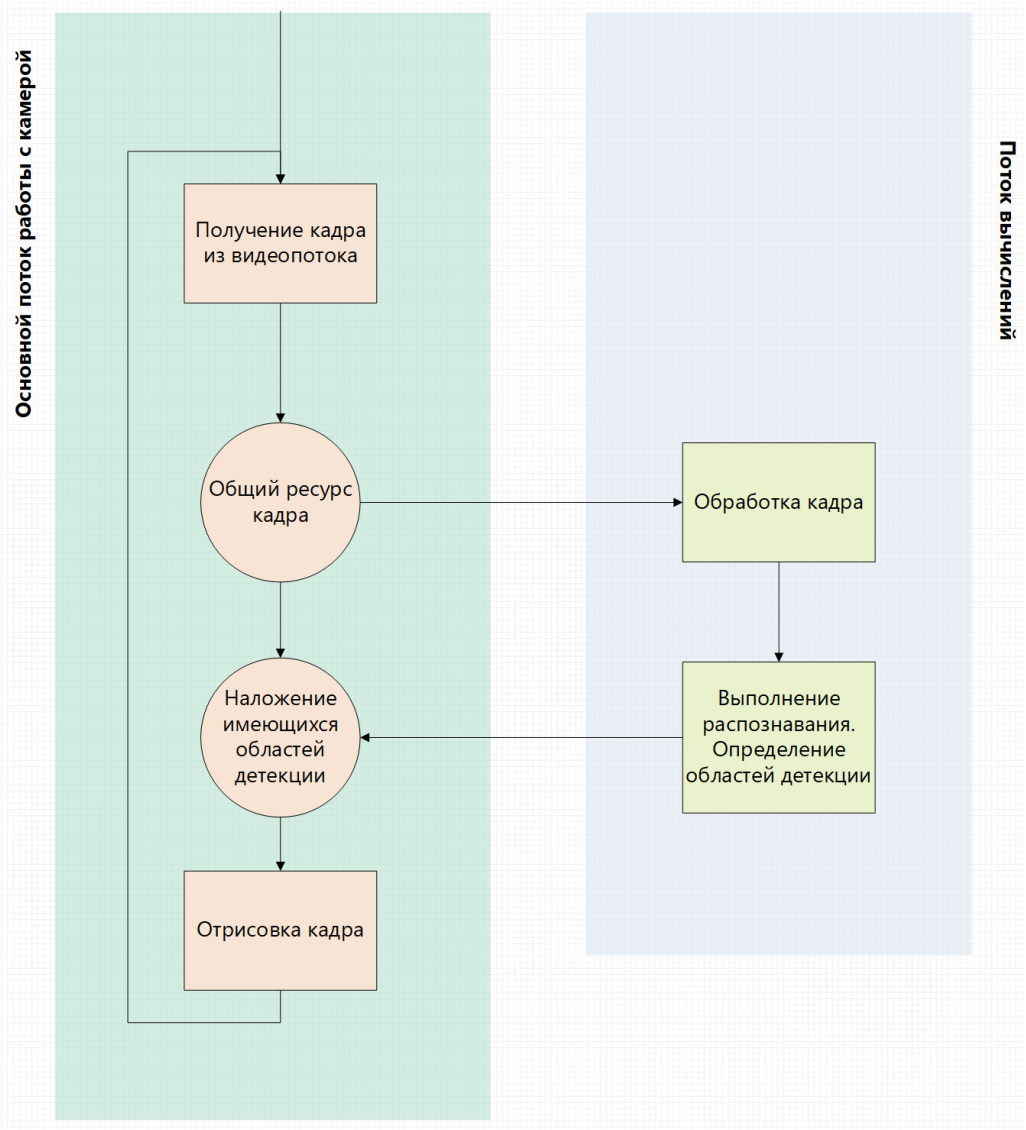

Рисунок 3.2 - Схема распознавания лиц из видеопотока

Сначала производится инициализация распознавателя лиц и получение ресурса камеры. А также определение контейнера, куда будут помещаться считанные кадры.

```
recognizer= new Recognizer(canvas, classifier);
capture= new opencv_videoio.VideoCapture(0);
opencv_core.Mat frameMat = newopencv_core.Mat();
```

Далее в цикле считываем кадр, выводим само изображение, передает это изображение в распознаватель, отрисовываем зоны детекции поверх кадра

```
capture.read(frameMat);
canvas.showImage(converter.convert(frameMat));
```

```
recognizer.setMat(frameMat);
drawDetections(canvas);
```

Теперь рассмотрим работы распознавателя, который выполняется в отдельном потоке.

Определяется контейнер изображения. Для работы данного метода распознавания используется монохромное изображение, поэтому преобразуем полученное изображение в монохромное. Далее происходит выравнивание гистограммы изображения, чтобы алгоритм срабатывал стабильно при разной освещенности.

```
opencv_core.Mat videoMatGray = new opencv_core.Mat();
if(getMat() == null) {
continue;
}
cvtColor(getMat(), videoMatGray, COLOR_BGRA2GRAY);
equalizeHist(videoMatGray, videoMatGray);
```

Далее идет непосредственное распознавание на основе классификатора. Используется функция detectMultiScale, которая принимает параметры:

- исходное изображение;
- контейнер, куда будут записаны прямоугольники детекции;
- коэффициент увеличения масштаба;
- минимальный соседний порог - это то, с какой интенсивностью находится каждая часть лица. Надо настраивать вручную, так как без порога детектор генерирует много одних и тех же распознаваний в одном месте. Обычно изолированные обнаружения являются ложными, так что имеет смысл отказаться от них. Также имеет смысл объединить несколько обнаружений для каждой лицевой области в единую сущность.

Флаговая переменная может принимать несколько значений:

- 0;
- CV_HAAR_DO_CANNY_PRUNING;
- CV_HAAR_FIND_BIGGEST_OBJECT;

- CV_HAAR_DO_ROUGH_SEARCH;
- CV_HAAR_DO_CANNY_PRUNING;
- CV_HAAR_SCALE_IMAGE.

Если выбран флаг «практичной обрезки» (0|CV_HAAR_DO_CANNY_PRUNING), то алгоритм не учитывает области изображения, где нахождение лица не ожидается. Это снижает время вычисления, и, возможно, устраняет некоторые ложные обнаружения. Пропускаемые области идентифицируются детектором, который обнаруживает такие «ненужные» края по всему изображению, перед запуском лицевого детектора. Опять же, выбор установки данного типа флага является компромиссом выбора между скоростью и обнаружением большего количества лиц. Установка флага приведет к увеличению скорости обработки, но может привести к пропуску некоторых лиц. [4]

```
opencv_core.RectVector rectVector = new opencv_core.RectVector();
classifier.detectMultiScale(videoMatGray, rectVector, 1.1, 7,
CV_HAAR_DO_CANNY_PRUNING| CV_HAAR_FIND_BIGGEST_OBJECT,
new opencv_core.Size(48, 48), new opencv_core.Size());
setDetections(rectVector);
```

Методом setDetections зоны детекции передаются для отрисовки основным потоком работы камеры.

Это основные принципы работы приложения.

3.2 Разработка интерфейса

Интерфейс программы представляет собой окно с выводом видеопотоке в центре и панелью управления снизу (рисунок 3.3).

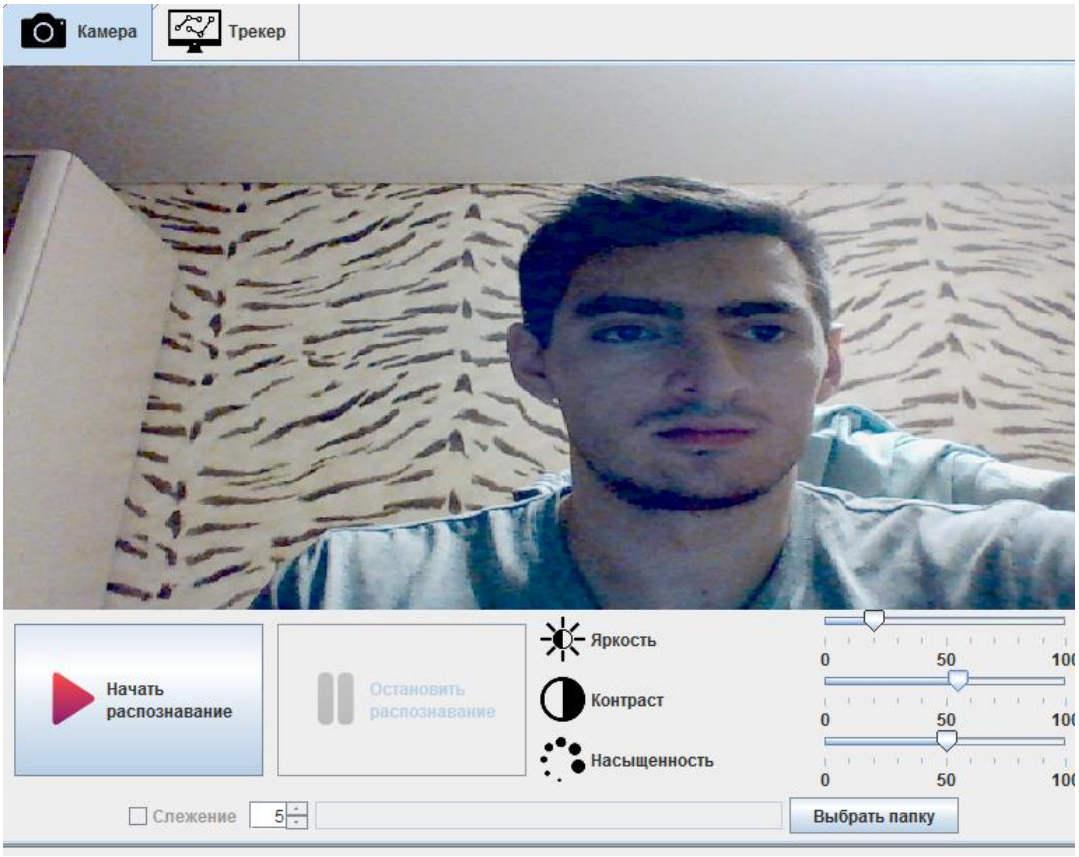

Рисунок 3.3 - Интерфейс программы

При запуске программы работает только камера. Распознавание можно запустить и остановить кнопками «Начать распознавание» и «Остановить распознавание».

Также имеется управление изображением камеры. К изображению с камеры сразу применяются эти настройки:

- яркость;
- контрастность;
- насыщенность.

Еще одной возможность является слежение. Нужно указать папку сохранения изображений и интервал срабатывания в секундах. При включении флага «Слежение» программа через равные промежутки времени производит проверку камеры на наличие лиц (рисунок 3.4).

Если лицо обнаруживается, то делается снимок и сохранение кадра, при этом идет запись в БД с путем сохранения и отметкой времени. Потом

происходит ожидание равное интервалу, по истечение которого обнаружение снова включается.

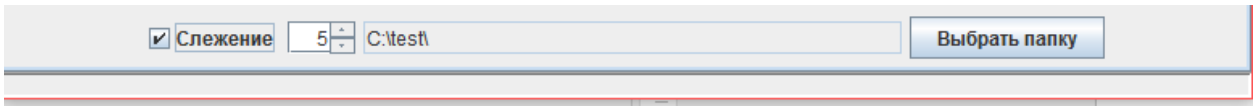

Рисунок 3.4 - Интерфейс слежения

На вкладке «Трекер» можно увидеть результат работы слежения, где отображается таблица с результатами. При выборе строки в левой части показывается сохраненное изображение.

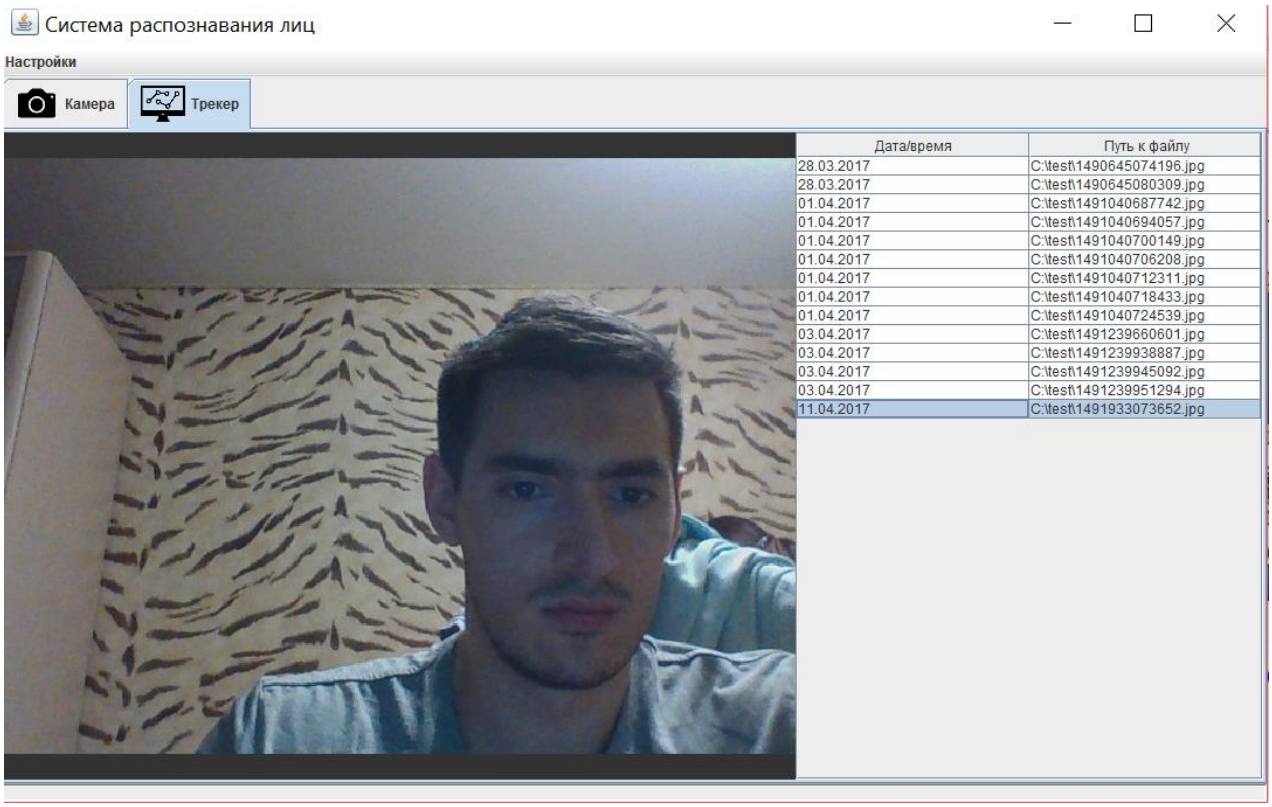

Рисунок 3.5 - Вкладка «Трекер»

В приложении имеется возможность сохранения значений корректировки изображений, папки слежения, интервала.

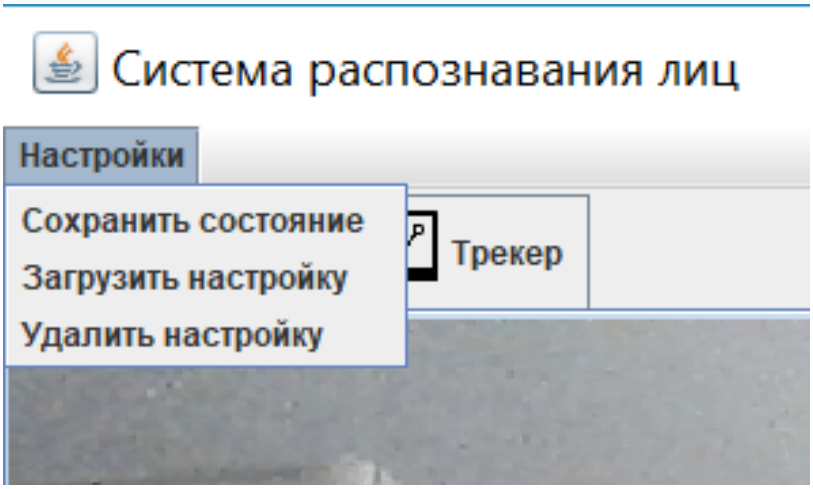

Рисунок 3.6 - Меню настроек

Меню настроек позволяет быстро задать предустановленный параметры слежения и качества картинки видеопотока. Рисунки окон управления настройками показаны в приложении В. Полный код программы представлен в приложении Д.

3.3 Тестирование разработанной программы

Для проверки работоспособности и быстродействия алгоритма были проведены тестовые испытания.

Для самого теста потребовался массив изображений в локальной директории (10 изображений). Все изображения имеют разный размер, есть цветные, а есть и черно-белые фотографии, с разным количеством человек в различных положениях на них.

Для определения статистики правильности распознавания был проведен подсчет ошибок распознавания по формулам:

$$P_1 = N_{расп} : N_{общ}, \quad (3.1)$$

$$P_2 = N_{нерасп} : N_{общ}, \quad (3.2)$$

где P_1 – ошибки первого рода;

P_2 – ошибки второго рода;

$N_{расп}$ – количество распознанных объектов;

$N_{нерасп}$ – количество нераспознанных объектов;

$N_{общ}$ – общее количество объектов.

Для проведения тестов был использован встроенный файл OpenCV - performance.cpp, где ведется подсчет всех данных и выстраивание ROC-кривой. Пример подсчета этапов или стадий классификатора:

```
int* numclassifiers = newint[cascade->count];
numclassifiers[0] = cascade->stage_classifier[0].count;
for( i = 1; i < cascade->count; i++ )
// последовательное применение классификаторов
{ numclassifiers[i] = numclassifiers[i-1] + cascade->stage_classifier[i].count;}
```

В таблице 3.1 представлены результаты тестирования алгоритма с указанием точности работы с натренированным набором для распознавания черт лица.

Таблица 3.1 – Результаты тестирования

Данные\Алгоритм	Виола-Джонс	P_1/P_2
Массив изображений (10 штук)	91%, ~7 сек (7475 ms)	70% / 30%

Использованные для теста изображения были показаны на камеру в процессе работы программы. При нахождении лица программа выделяет участок зеленым прямоугольником. Результаты показаны на рисунках в приложении Г.

На предобработку фотографии затрачивается 250-300 мс. Само определение лица происходит быстрее, так как функция определяет типы сработавших классификаторов и молниеносно выдает результат. Но за счет вынесения вычислений в отдельный поток, это время не учитывается.

По результатам тестирования основная часть лиц была распознана даже несмотря на разность освещения, перекрытия и положения. Те лица, которые не

были распознаны, не обладают достаточными признаками, чтобы классификатор счел их подходящими. Как правило, это происходит, когда на видео присутствует лишь часть лица, изображение размыто, слишком далеко. В остальных случаях отсутствие распознавания является не недостатком алгоритма, а недостаточной натренированностью классификатора.

3.4 Обоснование экономической эффективности системы распознавания лиц с применением библиотеки OpenCV

Программные комплексы и системы видеонаблюдения, в том числе предоставляющие возможности распознавания лиц в видеопотоках в настоящее время являются активно развивающимся направлением. По оценкам компании MarketsandMarkets в ближайшие годы рынок видеоаналитики продолжит расти, и к 2020 году составит более 4000 миллионов долларов. Поэтому можно сказать, что представленная разработка системы распознавания лиц с применением библиотеки OpenCV потенциально является коммерчески перспективным продуктом.

В технико-экономичсеском обосновании данной выпускной квалификационной работы будут рассмотрены следующие блоки:

- составление детализированного плана-графика выполнения работ;
- оценка величины заработной платы и социальных отчислений;
- оценка всех затрат, связанных с разработкой проекта;
- оценка себестоимости проекта.

Все расчеты производились в соответствии с методическими указаниями по дополнительному разделу выпускной квалификационной работы.

Для оценки затрат на заработную плату необходимо составить план-график выполнения работ. Данный план представлен в таблице 3.2.

Таблица 3.2 - План-график выполнения работ

№ п/п	Наименование работ	Исполнитель	Срок выполнения, дн
1.	Разработка ТЗ	Руководитель предприятия	2
2.	Сбор и анализ литературы по теме ВКР	Студент	10
3.	Проведение исследования эффективности работы	Студент	10
4.	Разработка системы распознавания лиц в видеопотоках	Студент	20
5.	Тестирование разработанной системы	Студент	5
6.	Оформление пояснительной записки к ВКР	Студент	10

В результате получены следующие данные о длительности работ:

— общая длительность работ составляет 57 дней;

— продолжительность работы руководителя предприятия составляет 2 дня;

— продолжительность работы студента составляет 55 дней.

Для расчета затрат на основную заработную плату необходимо установить размер заработной платы для исполнителей проекта, а именно руководителя ООО "Триатрон" и студента. Руководителем проекта является директор ООО "Триатрон", в соответствии с приказом "О начислении заработной платы" месячный оклад руководителя составляет 23700 рублей. Для

студента согласно тому же приказу, устанавливается месячный оклад как инженерно-техническому работнику (инженеру-программисту) в размере 6900 рублей. Будем считать, что в месяце имеется 21 рабочий день.

Исходя из приведенных данных, рассчитаем дневную ставку исполнителей работ. Дневная ставка руководителя составит:

$$\text{ЗПДрук} = \frac{23700}{21} = 1128 \frac{\text{рублей}}{\text{день}} \quad (3.3)$$

Дневная ставка студента составляет:

$$\text{ЗПДстуд} = \frac{6900}{21} = 328 \frac{\text{рублей}}{\text{день}} \quad (3.4)$$

В таблице 3.3 представим итоговые затраты на основную заработную плату для всех участников проекта.

Таблица 3.3 - Затраты на основную заработную плату

Исполнитель	Оплата, руб/день	Количество дней	Основная заработная плата, руб
Руководитель	1128	2	2256
Студент	328	55	18040
ИТОГО:			20296

Расходы на дополнительную заработную плату участников проекта рассчитываются по формуле (3.5):

$$\text{Здоп/пл} = \text{Зоснз/пл} * \frac{\text{Ндоп}}{100}, \quad (3.5)$$

где Здоп/пл - расходы на дополнительную заработную плату, руб.;

Зоснз/пл - расходы на основную заработную плату, руб.;

Ндоп - норматив дополнительной заработной платы, составляющий 14%.

В результате дополнительная заработная плата руководителя составит:

$$\frac{\text{Здоп}}{\text{пл}} = 2256 * \frac{14}{100} = 316 \text{ (руб)} \quad (3.6)$$

Таким же образом вычислим дополнительную заработную плату студента:

$$\frac{\text{Здоп}}{\text{пл}} = 18040 * \frac{14}{100} = 2526 \text{ (руб)} \quad (3.7)$$

Отчисления на социальные нужны можно рассчитать по формуле (3.8):

$$\text{Зсоц} = \text{Зоснз/пл} + \text{Здопз/пл} * \frac{\text{Нсоц}}{100} \ , \quad (3.8)$$

где Зсоц - отчисления на социальные нужны с заработной платы, руб.;

Зоснз/пл - расходы на основную заработную плату, руб.;

Здопз/пл - расходы на дополнительную заработную плату, руб.;

Нсоц - норматив отчислений на социальные нужды, составляющий 30%.

Для руководителя ООО «Триатрон» отчисления на социальные нужды составят:

$$\text{Зсоц} = 2256 + 316 * \frac{30}{100} = 772 \text{ (руб)} \quad (3.9)$$

Для студента отчисления на социальные нужды составят:

$$\text{Зсоц} = 18040 + 2526 * \frac{30}{100} = 6170 \text{ (руб)} \quad (3.10)$$

Сведем в таблицу 3.4 данные о полной заработной плате руководителя и студента.

Таблица 3.4 - Затраты на полную оплату труда

Исполнитель	Основная заработная плата, руб	Дополнительная заработная плата, руб	Социальные отчисления, руб	Полная заработная плата, руб
Руководитель	2256	316	772	3344
Студент	18040	2526	6170	26736
ИТОГО:				30080

Для выполнения плана выпускной квалификационной работы (разработка системы распознавания лиц с применением библиотеки OpenCV) необходимо организовать рабочие места для студента и руководителя.

Руководитель выполнял работы с использованием персонального компьютера, студент выполнял работы на ноутбуке. Поскольку данные технические средства были приобретены ранее на предприятии в связи с этим материальные затраты на их покупку не учитывались. Для точного расчета стоимости разработки необходимо учесть амортизационные отчисления по основному средству, которые определяются формулой (3.11):

$$Аi = Цплі * \frac{Наі}{100} \quad , \tag{3.11}$$

где Аі - амортизационные отчисления за год по і-му основному средству, руб.;

Цплі - первоначальная стоимость і-го основного средства, руб.;

Наі - годовая норма амортизации і-го основного средства, которую в данной работе примем равной 10% для всех средств.

Таким образом, годовые амортизационные отчисления по всем материальным средствам составляют 4012 рублей. Величина амортизационных отчислений по і-му основному средству рассчитывается по формуле (3.12):

$$Аівкр = Аі * \frac{Тівкр}{12}, \tag{3.12}$$

где Аівкр - амортизационные отчисления по і-му основному средству, используемого студентом в работе над выпускной квалификационной работе;

Аі - амортизационные отчисления за год по і-му основному средству, руб.;

Тівкр - время, в течении которого студент использует і-ое основной средство, мес.

Время работы над выполнением выпускной квалификационной работы составляет 2 месяца, соответственно амортизационные отчисления по всем основным средствам за время выполнения выпускной квалификационной работы составляет 669 рублей.

Для определения полной себестоимости системы распознавания лиц с применением библиотеки OpenCV сведем все описанные затраты в итоговую таблицу 3.5.

Таблица 3.5 - Затраты на выполнение выпускной квалификационной работы

№ п/п	Наименование статьи	Сумма, руб
1.	Основная заработная плата	20296
2.	Дополнительная заработная плата	2842
3.	Отчисления на социальные нужны	6942
4.	Амортизационные отчисления	669
ИТОГО:		30749

Таким образом, можно сделать следующие выводы по экономической эффективности работы над созданием системы распознавания лиц с применением библиотеки OpenCV:

— составлен план-график выполнения работ студентом и руководителем ООО "Триатрон;

— рассчитана сумма затрат на заработную плату исполнителем;

— произведен расчет суммы отчислений на социальные нужды;

— рассчитана сумма амортизационных отчислений;

— определены затраты на разработку системы распознавания лиц с применением библиотеки OpenCV, которые составили 30749.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были достигнуты все поставленные цели. Согласно поставленным целям были исследованы основные существующие методы распознавания лиц.

На сегодняшний день распознавание объектов в мультимедийном видеопотоке становится особо актуальным. Ведется очень много исследований в этой области. На одной из конференций была презентация одной интересной системы, разработанной немецкими учеными, в которой программное обеспечение распознавало фигуры людей, и в зависимости от того, куда двигался человек программа автоматически поворачивала камеру и следила за ним. Данную систему возможно использовать для автоматической записи лекций, которые читает преподаватель у доски.

Путем выявления достоинств и недостатков был определен самый подходящий метод для распознавания в видеопотоке. Метод Виолы-Джонса на данный момент является максимально подходящим по всем параметрам для задач распознавания объектов в видеопотоке. Был разработан и реализован алгоритм в виде программы и использование библиотеки компьютерного зрения OpenCV, которая содержит алгоритмы для: интерпретации изображений, калибровки камеры по эталону, устранение оптических искажений, определение сходства, анализ перемещения объекта, определение формы объекта и слежение за объектом, 3D-реконструкция, сегментация объекта, распознавание жестов и т.д. Эта библиотека очень популярна за счёт своей открытости и возможности бесплатно использовать как в учебных, так и коммерческих целях.

Разработанное приложение позволяет осуществлять видеонаблюдение, выделяя лица людей в видеопотоке, а также осуществлять автоматическое слежение с фотофиксацией лиц людей, которые были зафиксированы при наблюдении. Этот механизм позволяет делать фотоотчет с датой и временем, когда сработала детекция. При этом отпадает необходимость сохранять всю

видеозапись наблюдения. Попавшие в кадр лица будут сохранены и доступны к просмотру, как в приложении, так и в папке проводника. Кроме того, есть возможность коррекции изображения с целью улучшения качества картинки в условиях плохой освещенности.

Тестирование показало, что программа справляется со своей задачей. Подавляющая часть объектов была успешно распознана. При внедрении программы повысилась безопасность на складах предприятия «Триатрон», за счет детекции персонала, посещающих склад и обеспечении удобства доступа к информации, собранной программой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Egmont-Petersen, M. Image processing with neural networks – a review. Pattern Recognition 35 [Текст]/ M. Egmont–Petersen, D. de Ridder, H. Handels – Elsevier B.V, 2002. – 2301 с.

2 Kaehler, A. Learning OpenCV 3 Computer Vision in C++ with the OpenCV Library [Текст]/ A. Kaehler – O'Reilly Media, 2015. – 650 с.

3 Laganièr, R. OpenCV Computer Vision Application Programming Cookbook Second Edition [Текст]/ R. Laganièr – PACKT Publishing, 2014. – 570 с.

4 Suarez, O. D. OpenCV Essentials [Текст]/ O. D. Suarez – PACKT Publishing, 2014. – 366 с.

5 Форсайт, Д. А., Компьютерное зрение. Современный подход = Computer Vision [Текст]/ Д. А. Форсайт, Д. Понс. — Вильямс, 2004. — 928 с.

6 Шапиро, Л., Компьютерное зрение = Computer Vision [Текст]/ Л. Шапиро, Д. Стокман — Лаборатория знаний, 2006. – 752 с.

7 Фомин, Я. А. Распознавание образов: теория и применения [Текст]/ Я. А. Фомин – ФАЗИС, 2012. — 429 с.

8 Фомин, Я. А., Статистическая теория распознавания образов [Текст]/ Я. А. Фомин, Г. Р. Тарловский – Наука, 1979. — 624 с.

9 Горелик, А. Л., Методы распознавания. — 4 – е изд. [Текст]/ А. Л. Горелик, В. А. Скрипкин — Высшая школа, 2004. — 262 с.

10 Вапник, В. Н., Теория распознавания образов [Текст]/ В. Н. Вапник, А Я. Червоненки — Наука, 1974. — 416 с.

11 Земцов, А. В., Алгоритмы распознавания лиц и их применение в системах биометрического контроля доступа [Текст]/ А. В. Земцов – LAP Lambert Academic Publishing, 2011. – 128 с.

12 Закревский, А.Д., Логика распознавания [Текст]/ А. Д. Закревский. – . 2003, – 144 с. – (статья)

13 Распознавание образов. Состояние и перспективы [Текст]/ К. Верхаген, Р. Дейн, Ф. Грун и др, – Радио и связь 1985. – 104 с.

14 Kaeler, A., Learning OpenCV 3 Computer Vision in C++ with the OpenCV Library [Текст]/ A. Kaehler, G. Bradski – O'Reilly Media, 2015. – 650 с.

15 Николаев, М. И., Измерительный контроль с применением неиндустриальных камер в производственных условиях [Текст]/ М. И. Николаев – Наука, 1989. – 312 с.

16 Горшенин, В. А., Геодезический инвариант в биометрике лица [Текст]/ В. А. Горшенин –. 2014. – 4 с. – (науч. статья)

17 Алфимцев, А.Н., Разработка и исследование методов захвата, отслеживания и распознавания динамических жестов [Текст]/ А. Н. Алфимцева – МГТУ им. Н.Э. Баумана. – Москва: 2008. – 226 с.

18 Ирматов, А.А., Способ и система для распознавания лица с учетом списка людей, не подлежащих проверке [Текст]/ А. А. Ирматов, Д. Ю. Буряк – Корпорация «Самсунг электроникс Ко., Лтд.». – Москва: 2010. – 22 с.

19 Колесник, А. В., Распределенная программная система распознавания лиц [Текст]/ А. В. Колесник, Ю. В. Ладыженский – VI международная научно – техническая конференция студентов, аспирантов и молодых научных работников «Информатика и компьютерные технологии» – Донецк: ДонНТУ, 2010. – 251 с.

20 Умяров, Н. Х., Выделение лица на снимке из видеопотока с целью его распознавания [Текст]/ Н. Х. Умояров, О. И. Федяев – VII международная научно – техническая конференция студентов, аспирантов и молодых научных работников «Информатика и компьютерные технологии» – Донецк: ДонНТУ, 2011. – 177 с.

21 Костецкая, Г. Ю., Кодирование изображений человеческих лиц с помощью самоорганизующейся карты Кохонена [Текст]/ Г. Ю. Костецкая, О. И. Федяев – V международная научно – техническая конференция студентов, аспирантов и молодых научных работников «Информатика и компьютерные технологии» – Донецк: ДонНТУ, 2009. – 268 с.

22 Processing Neocognitron of Face Recognition on High Performance Environment Based on GPU with CUDA Architecture [Текст]/ G. Poli, J. H. Saito, J.

F. Mari, M. R. Zorzan, – 20th International Symposium on Computer Architecture and High Performance Computing, 2008. – 88 с.

23 The Java Language Specification, Second Edition [Текст]/ J. Gosling, B. Joy, G. Steele, G. Bracha – Oracle America, Inc. 2013. – 788 с.

24 The Java Language Specification, Second Edition [Текст]/ J. Gosling, B. Joy, G. Steele, G. Bracha, A. Buckley – Oracle America, Inc. 2013. – 604 с.

25 Шилдт, Г., Java. Полное руководство [Текст]/ Г. Шилдт – Oracle Press, 2014, – 1104 с.

26 Гослинг, Д., Язык программирования Java SE 8. Подробное описание [Текст]/ Д. Гослинг, Б. Джой – Oracle Press, 2014, – 672 с.

27 Блох, Д., Java. Эффективное программирование [Текст]/ Д. Блох – Oracle Press, 2014, – 310 с.

28 Мартин, Р., Чистый код: создание, анализ и рефакторинг. Библиотека программиста [Текст]/ Р. Мартин – Питер, 2016. – 464 с.

29 Макконелл, С., Совершенный код. Мастер – класс [Текст]/ С. Макконелл – Русская редакция, 2017. – 896 с.

30 Лафоре, Р., Структуры данных и алгоритмы в Java [Текст]/ Р. Лафоре – Питер, 2016. – 604 с.

31 Седжвик, Р., Алгоритмы на Java [Текст]/ Р. Седжвик, К. Уейн – Питер, 2016. – 848 с.

32 Метод Виолы – Джонса (Viola – Jones) как основа для распознавания лиц [Электронный ресурс]/ Н. Наумов –, 2011. –. режим доступа: https://habrahabr.ru/post/133826/

33 Open Computer Vision library [Электронный ресурс]/ OpenCV team, 2017. – . – режим доступа: http://opencv.org/

34 The CImg library [Электронный ресурс]/ CImg team, 2017. –. – режим доступа: http://cimg.sourceforge.net/

35 Теория распознавания образов [Электронный ресурс]/ –, 2017 –. – режим доступа: https:// ru.wikipedia.org/wiki/Теория_распознавания_образов

36 Marr – Hildreth [Электронный ресурс]/ –. 2017 –. – режим доступа: http://de.wikipedia.org/wiki/Marr – Hildreth – Operator

37 Face Recognition by Elastic Bunch Graph Matching [Электронный ресурс]/ –. 2017. –. – режим доступа: http://www.face – rec.org/algorithms/ebgm/wisfelkrue99-facerecognition- jainbook.pdf

38 DeepFace Closing the Gap to Human – Level Performance in Face Verification [Электронный ресурс]/ –. 2017. –. – режим доступа: https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf

39 Face Detection and Recognition Using Hidden Markovs Models [Электронный ресурс]/ V. Nefian, H. Hayes –. 2017. –. – режим доступа: http://www.anefian.com/research/nefian98_face.pdf

40 FaceVACSTechnology. B6T8 Algorithm Performance [Электронный ресурс]/ –. 2015. –. – режим доступа: http://www.cognitec-systems.de/fileadmin/cognitec/media/technology/FaceVACS-biometric-performance-b6t8.pdf

ПРИЛОЖЕНИЕ А

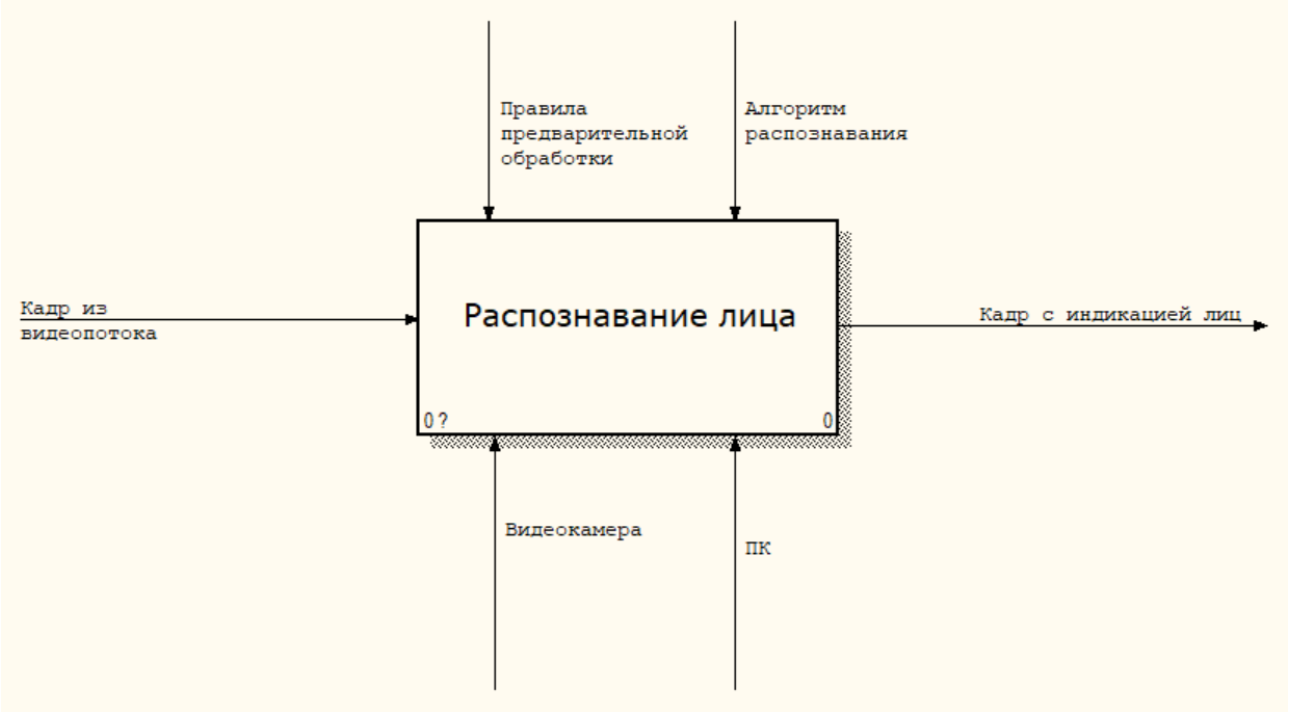

Рисунок А.1 – Контекстная диаграмма распознавания лиц из видеопотока в обобщенном виде.

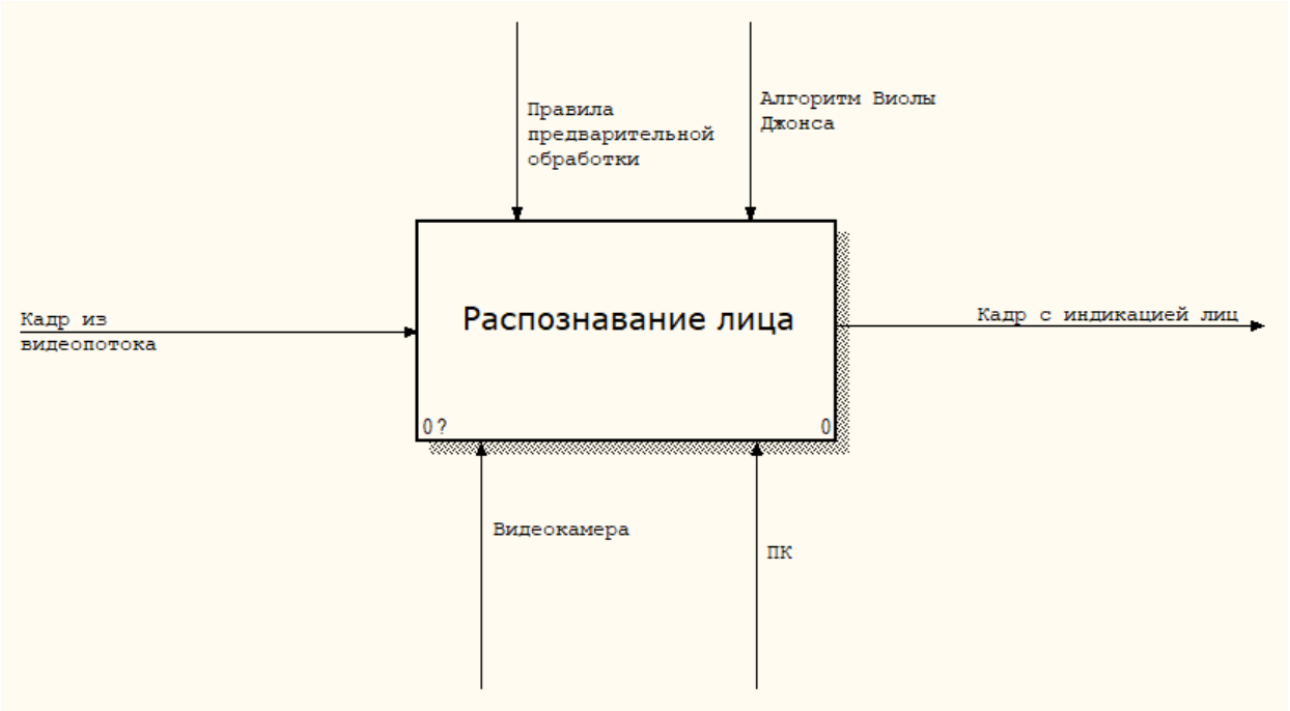

Рисунок А.2 – Контекстная диаграмма распознавания лиц из видеопотока по методы Виолы-Джонса

ПРИЛОЖЕНИЕ Б

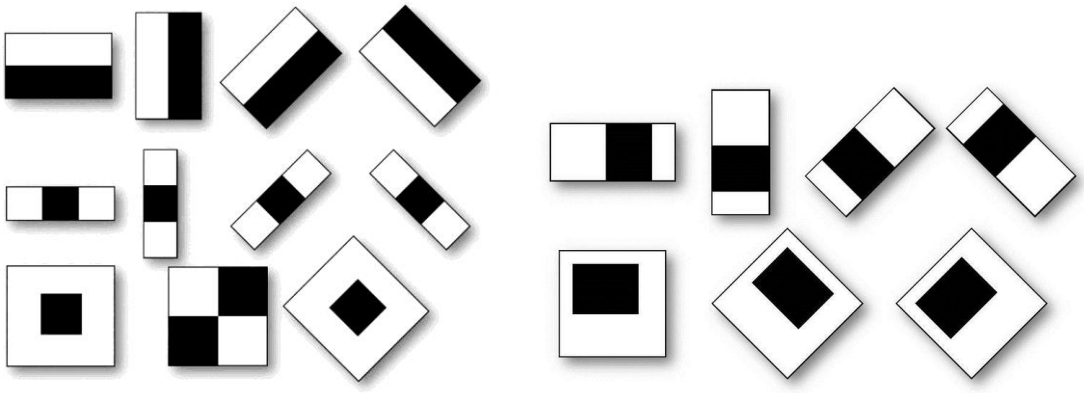

Рисунок Б.1 – прямоугольные и дополнительные признаки Хаара

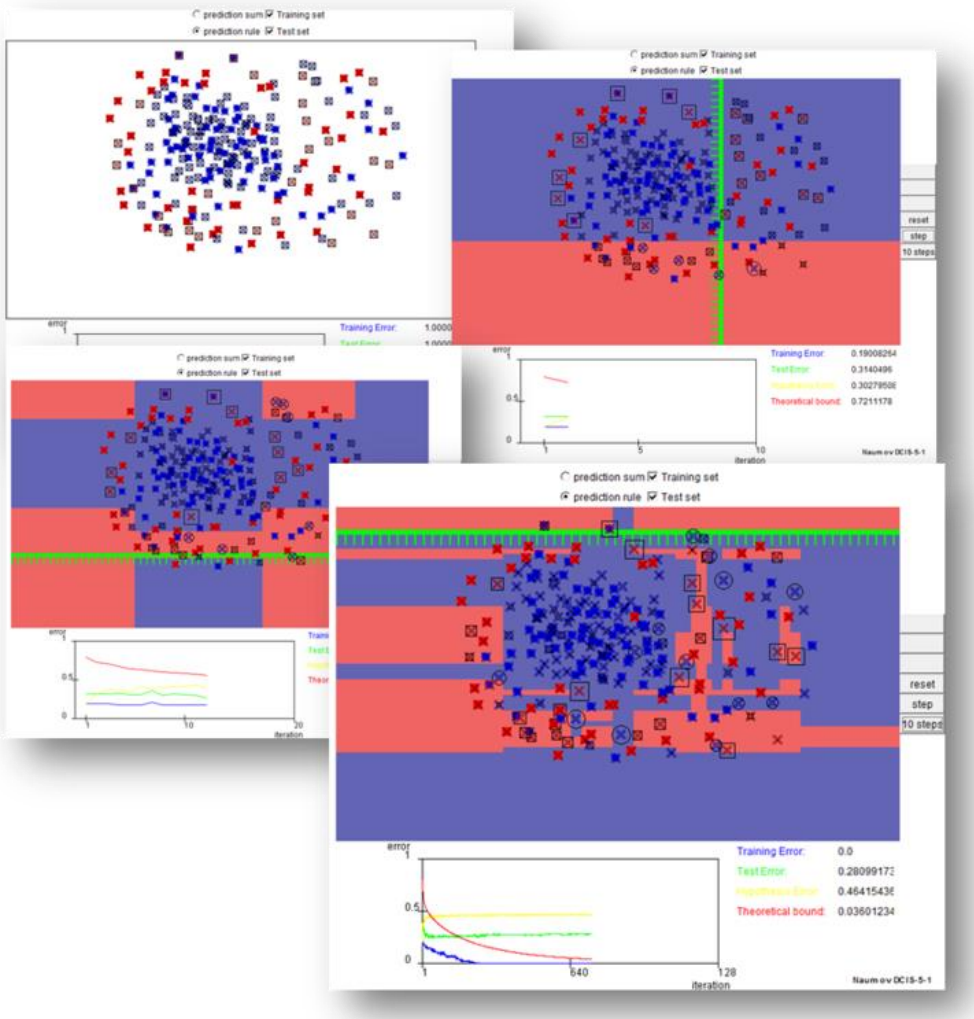

Рисунок Б.2 – Визуализация алгоритма AdaBoost

ПРИЛОЖЕНИЕ В

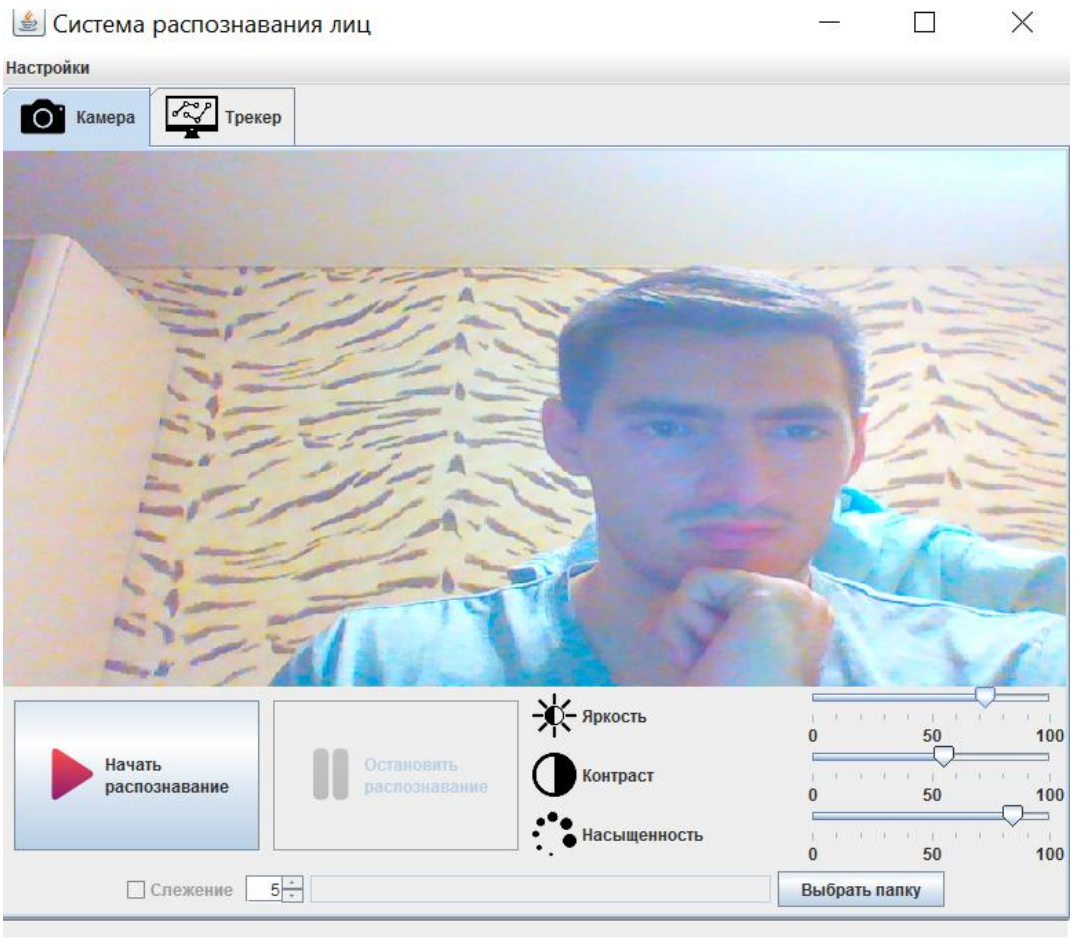

Рисунок В.1 - Результат применения настроек

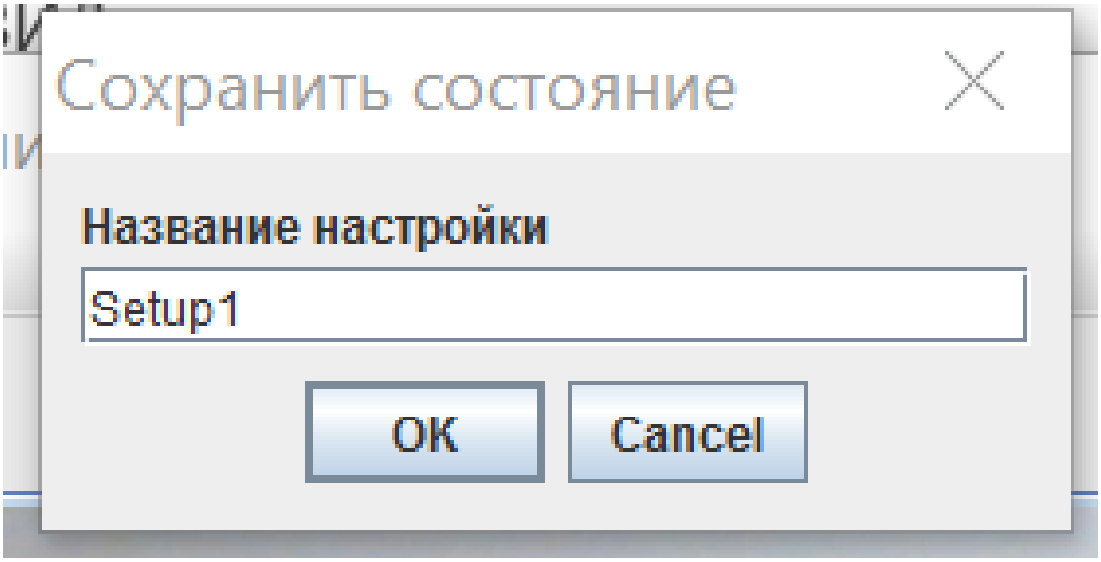

Рисунок В.2 - Окно сохранения настроек

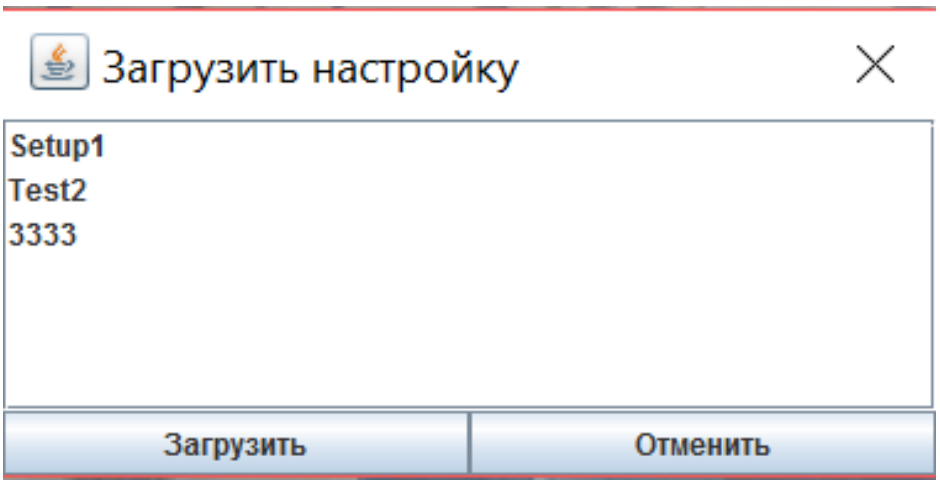

Рисунок В.3 - Окно загрузка настройки

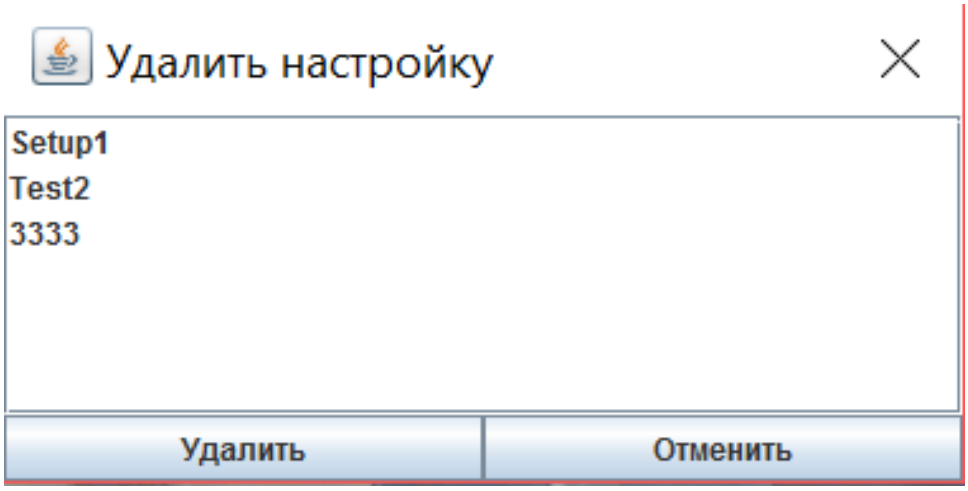

Рисунок В.4 - Окно удаления настроек

ПРИЛОЖЕНИЕ Г

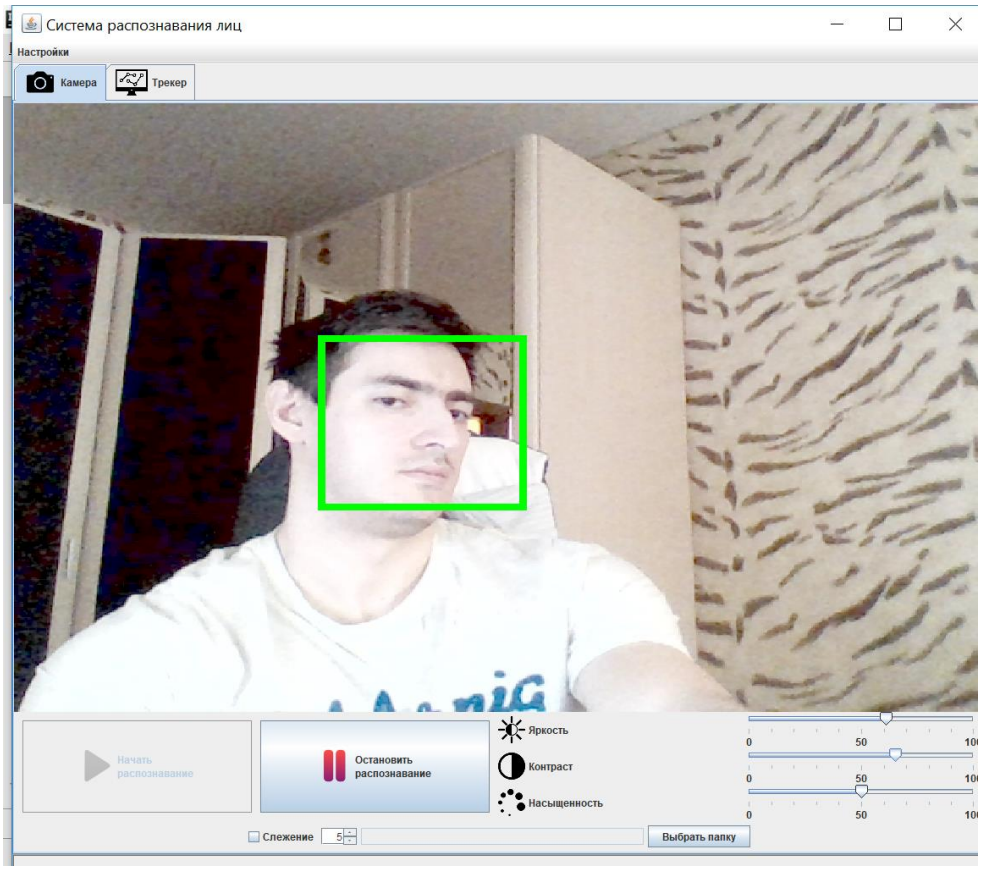

Рисунок Г.1 – Результат тестирования 1

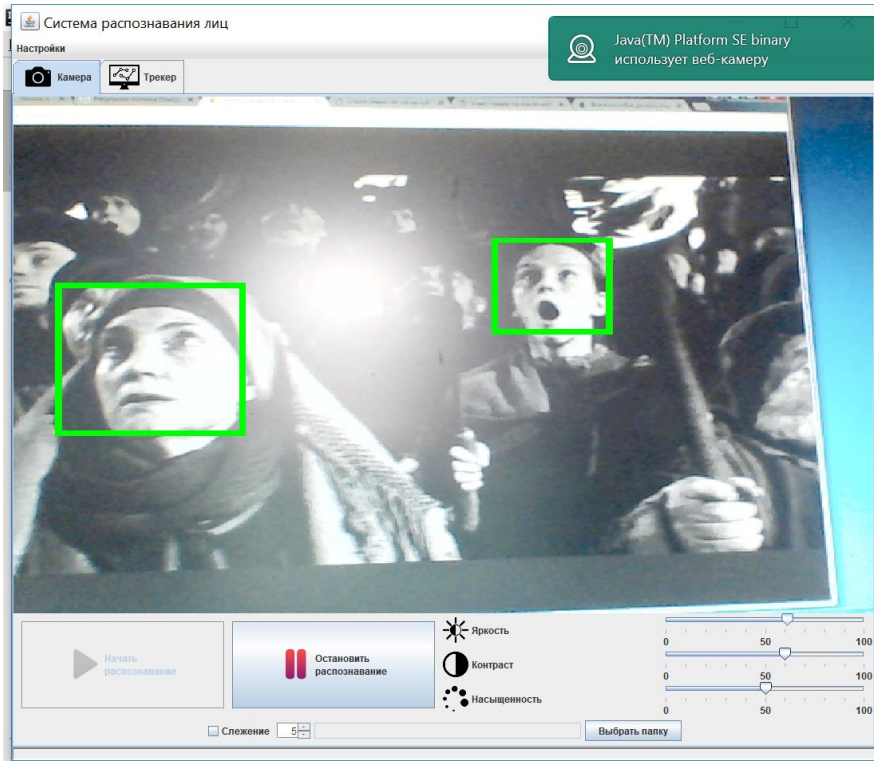

Рисунок Г.2 – Результат тестирования 2

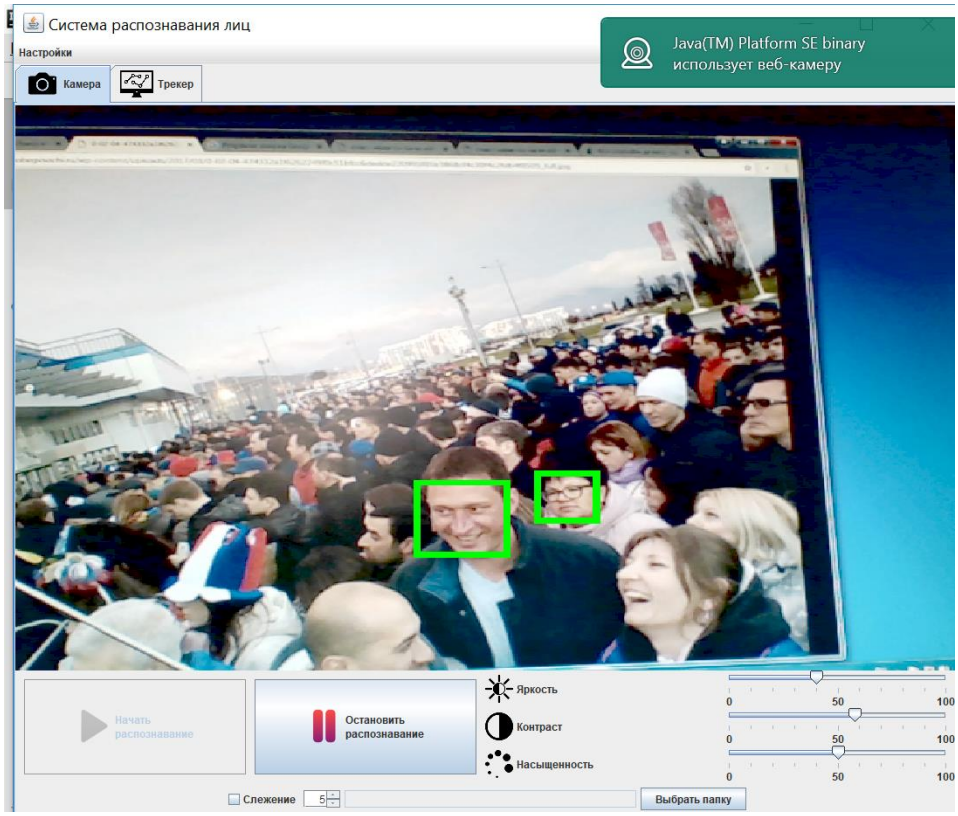

Рисунок Г.3 – Результат тестирования 3

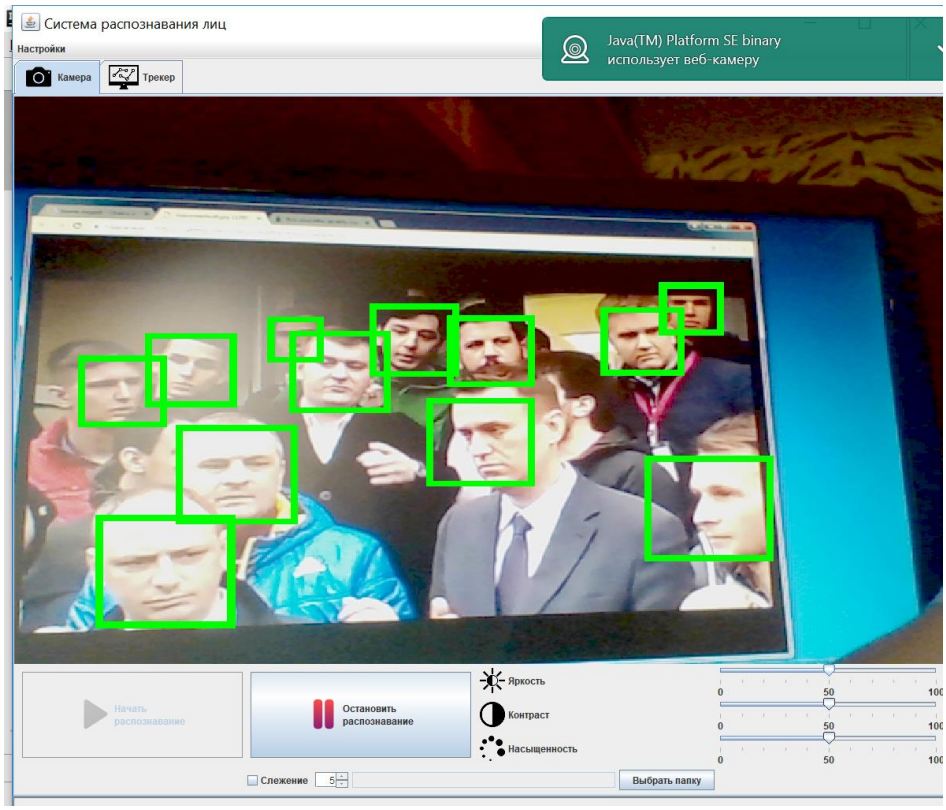

Рисунок Г.4 – Результат тестирования 4

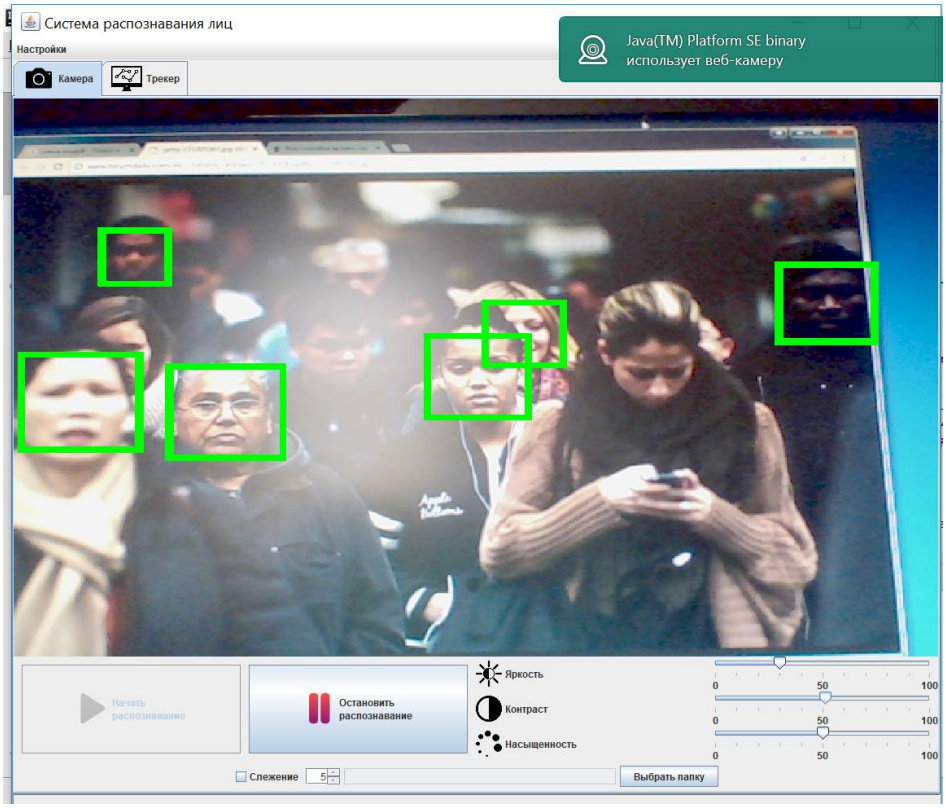

Рисунок Г.5 – Результат тестирования 5

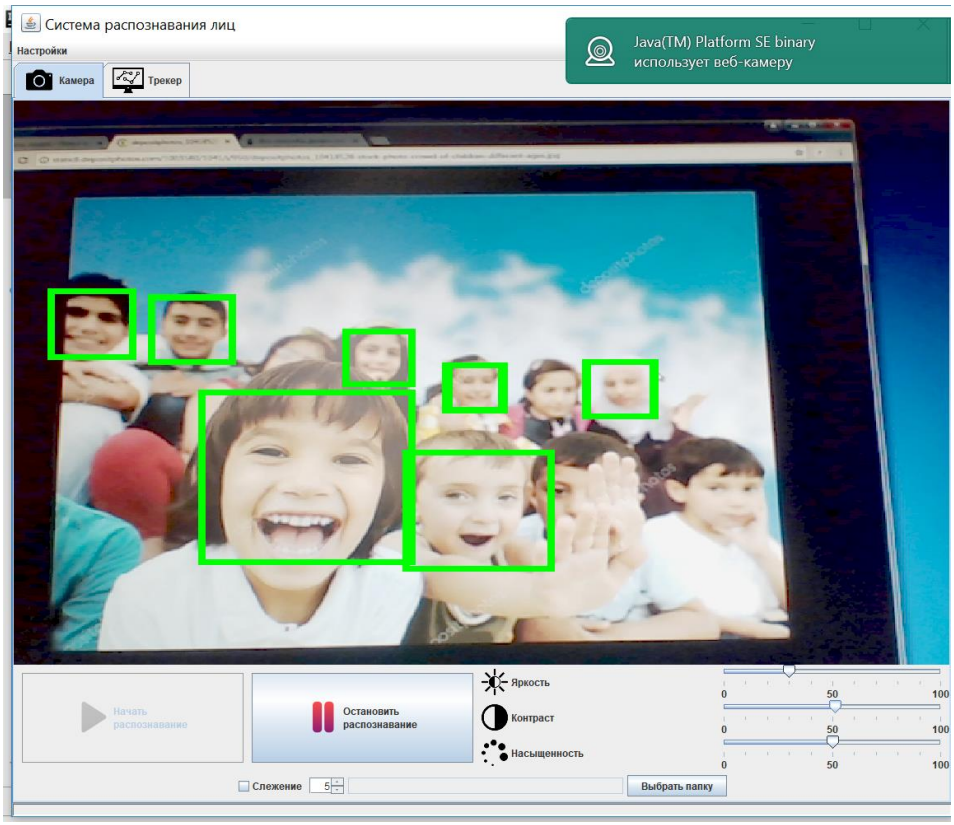

Рисунок Г.6 – Результат тестирования 6

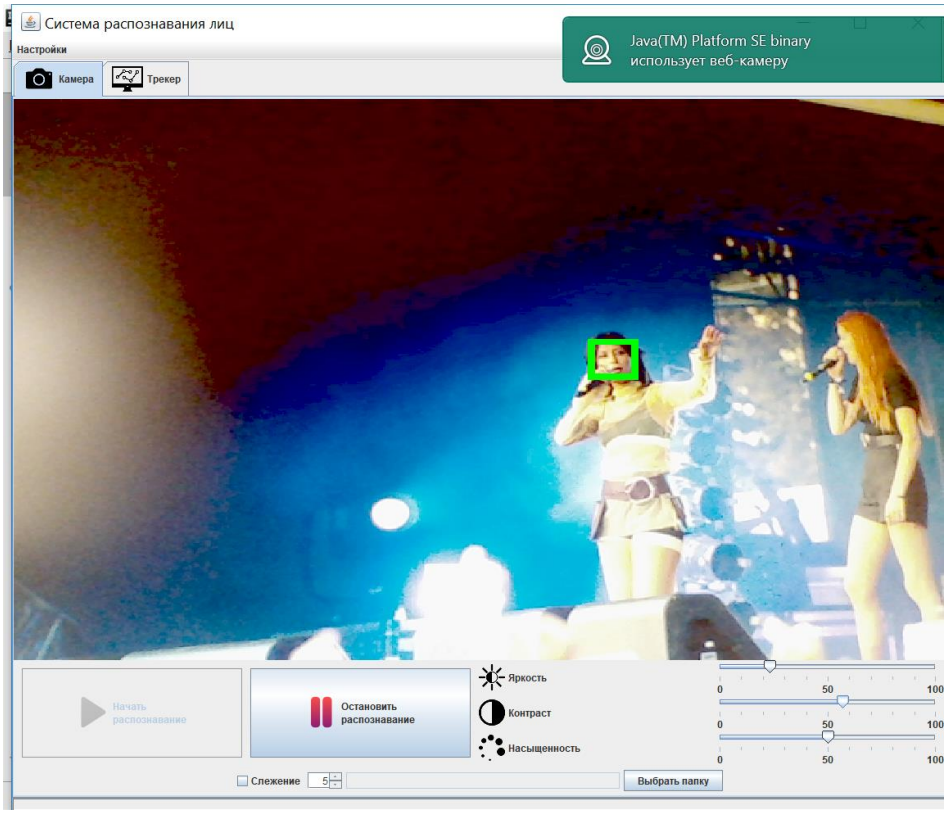

Рисунок Г.7 – Результат тестирования 7

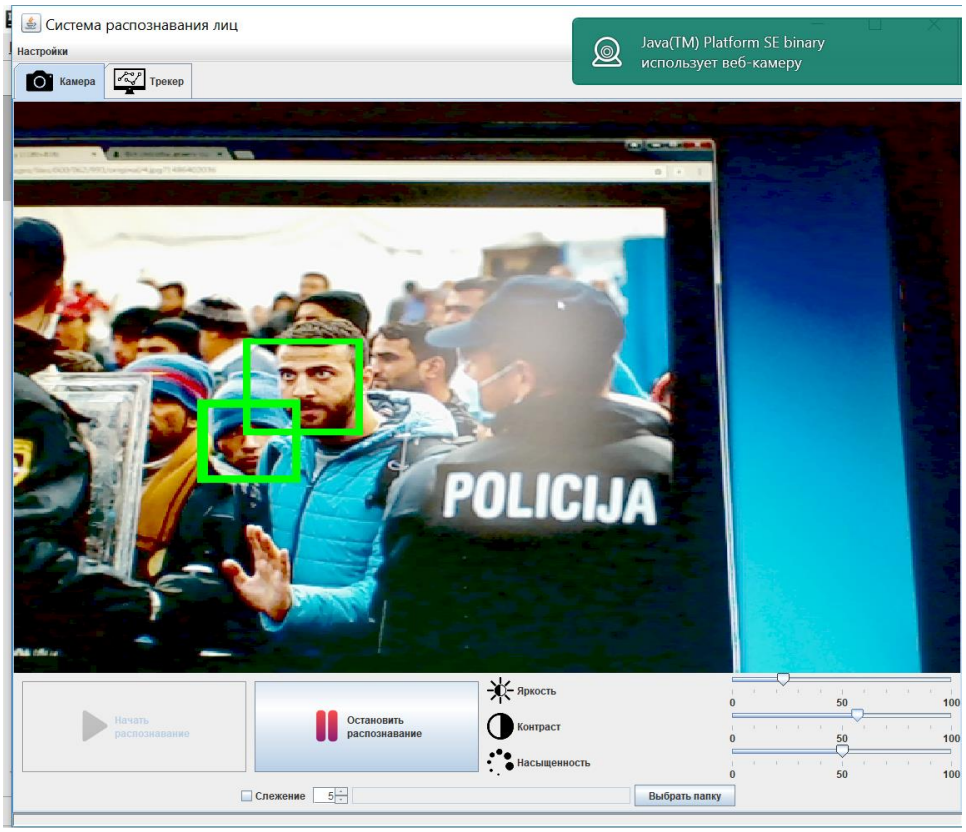

Рисунок Г.8 – Результат тестирования 8

ПРИЛОЖЕНИЕ Д

Исходный код разработанной программы по распознаванию лиц

```
import component.ControlPanel;
import component.MenuBar;
import component.StatusBar;
import component.tracker.TrackerPanel;
import config.ConfigReader;
import config.ControlConfig;
import config.WindowConfig;
import cv.CameraCanvas;

import javax.swing.*;
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import static component.Utils.getIcon;

/**
 * Основноеокнопрограммы
 */
public class MainFrame extends JFrame {

private CameraCanvas cameraCanvas;

private StatusBar statusBar = new StatusBar();

private ControlPanel controlPanel;

public MainFrame() throws HeadlessException {
        WindowConfig windowConfig =
ConfigReader.getInstance().getConfig(WindowConfig.class);
        setSize(windowConfig.getWindowWidth(), windowConfig.getWindowHeight());
        setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        setTitle("Системараспознаваниялиц");

cameraCanvas = new CameraCanvas(statusBar::setStatus, this::revalidate);
controlPanel = new ControlPanel(cameraCanvas);

        Container container = getContentPane();
        container.setLayout(new BorderLayout());
        container.add(new MenuBar(this, cameraCanvas, controlPanel), BorderLayout.NORTH);
        container.add(initTabBar(), BorderLayout.CENTER);
        container.add(initSouth(), BorderLayout.SOUTH);
```

```
cameraCanvas.startCamera();
        addWindowListener(new WindowAdapter() {

            @Override
public void windowClosing(WindowEvent e) {
cameraCanvas.close();
                System.exit(0);
            }
        });
    }

private JTabbedPane initTabBar() {
        JTabbedPane screenPane = new JTabbedPane();
        ControlConfig controlConfig =
ConfigReader.getInstance().getConfig(ControlConfig.class);

        JPanel panel = new JPanel(new BorderLayout());
        panel.add(cameraCanvas.getCanvas(), BorderLayout.CENTER);
        panel.add(controlPanel, BorderLayout.SOUTH);
        screenPane.addTab(controlConfig.getCameraTab(),
getIcon(controlConfig.getCameraTabIcon()), panel);
        screenPane.addTab(controlConfig.getIvTab(),
getIcon(controlConfig.getIvTabIcon()), new TrackerPanel());
        screenPane.addChangeListener(e -> {
switch (screenPane.getSelectedIndex()) {
case 0: {
if (!cameraCanvas.isRunning()) {
cameraCanvas.startCamera();
controlPanel.updateButtonsState();
                    }
break;
                }
case 1: {
if (!controlPanel.getMonitoringState()) {
cameraCanvas.close();
                    }
                }
            }
        });
return screenPane;
    }

private JPanel initSouth() {
        JPanel south = new JPanel();
        south.setLayout(new BorderLayout());
        south.add(statusBar, BorderLayout.SOUTH);
```

```
        south.setSize(new Dimension(getWidth(), 20));
return south;
    }
}



package cv;

import config.ConfigReader;
import config.ErrorConfig;
import org.bytedeco.javacpp.Loader;
import org.bytedeco.javacpp.opencv_core;
import org.bytedeco.javacpp.opencv_objdetect;
import org.bytedeco.javacpp.opencv_videoio;
import org.bytedeco.javacv.CanvasFrame;
import org.bytedeco.javacv.OpenCVFrameConverter;

import java.awt.*;
import java.awt.image.BufferedImage;

import static org.bytedeco.javacpp.opencv_core.cvClearMemStorage;
import static org.bytedeco.javacpp.opencv_videoio.*;

/**
 * Холсткамеры
 */
public class CameraCanvas {

/**
     * Слушательустановкистатуса
     */
public interface StatusListener {

/**
         * Обработчикустановкистатуса
         *
         * @param status статус
         */
void onSetStatus(String status);
    }

/**
     * Слушательпервогокадра
     */
public interface FirstFrameListener {
```

```
/**
         * Обработчикпервогокадра
         */
void onFirstFrame();
    }

/**
     * Окнохолста
     */
private CanvasFrame canvas = new CanvasFrame(null);

/**
     * Конвертеризображений
     */
private OpenCVFrameConverter.ToIplImage converter = new
OpenCVFrameConverter.ToIplImage();

/**
     * Классификатор
     */
private opencv_objdetect.CascadeClassifier classifier;

/**
     * Захваткамеры
     */
private opencv_videoio.VideoCapture capture;

/**
     * Распознаватель
     */
private Recognizer recognizer;

/**
     * Индикаторисполненияпотока
     */
private volatile boolean isRunning;

/**
     * Слушательстатуса
     */
private StatusListener statusListener;

/**
     * Слушательпервогокадра
     */
private FirstFrameListener firstFrameListener;
```

```
private Thread thread;

/**
     * @return исполнительпотока
     */
private Runnable getRunnable() {
return () -> {
isRunning = true;
            Loader.load(opencv_objdetect.class);
if (!initQualifier()) {
return;
            }
recognizer = new Recognizer(canvas, classifier);
capture = new opencv_videoio.VideoCapture(0);
            opencv_core.Mat frameMat = new opencv_core.Mat();

boolean firstFrame = true;
while (isRunning) {
capture.read(frameMat);
canvas.showImage(converter.convert(frameMat));
recognizer.setMat(frameMat);
                drawDetections(canvas);

if (firstFrame) {
firstFrameListener.onFirstFrame();
                    firstFrame = false;
                }
            }
capture.close();
        };
    }

/**
     * путькkаскадномуклассификатору
     */
private static final String FACE_CASCADE_PATH =
"training/data/haarcascade_frontalface_default.xml";

/**
     * Созданиехолста
     *
     * @param statusListener     слушательстатуса
     * @param firstFrameListener слушательпервогокадра
     */
publicCameraCanvas(StatusListenerstatusListener, FirstFrameListenerfirstFrameListener) {
canvas.setVisible(false);
this.statusListener= statusListener;
```

```
this.firstFrameListener= firstFrameListener;

initThread();
    }

/**
     * инициализация потока
     */
privatevoidinitThread() {
thread= newThread(getRunnable());
    }

/**
     * Инициализация классификатора
     *
     * @returnстатус инициализации
     */
privatebooleaninitQualifier() {
classifier= newopencv_objdetect.CascadeClassifier(FACE_CASCADE_PATH);

if(classifier.isNull()) {
ErrorConfigerrorConfig = ConfigReader.getInstance().getConfig(ErrorConfig.class);
statusListener.onSetStatus(errorConfig.getFaceCascadeLoadFailed(FACE_CASCADE_PATH));
returnfalse;
        }
returntrue;
    }

/**
     * Запуск детекции
     */
publicvoidstartDetection() {
recognizer.startDetection();
    }

/**
     * Остановка детекции
     */
publicvoidstopDetection() {
if(recognizer!= null) {
recognizer.stop();
        }
    }

/**
     * @returnзначения насыщенности
     */
```

```
publicdoublegetSaturation() {
returncapture.get(CAP_PROP_SATURATION);
    }

/**
     * @returnзначение яркости
     */
publicdoublegetBrightness() {
returncapture.get(CAP_PROP_BRIGHTNESS);
    }

/**
     * @returnзначение контрастности
     */
public double getContrast() {
return capture.get(CAP_PROP_CONTRAST);
    }

/**
     * Установка значения яркости
     *
     * @param v значения яркости
     */
public void setBrightness(double v) {
capture.set(CAP_PROP_BRIGHTNESS, v);
    }

/**
     * Установка значение насыщенности
     *
     * @param v значение насыщенности
     */
public void setSaturation(double v) {
capture.set(CAP_PROP_SATURATION, v);
    }

/**
     * Установка контрастности
     *
     * @param v установка контрастности
     */
public void setContrast(double v) {
capture.set(CAP_PROP_CONTRAST, v);
    }

/**
     * Запуск камеры
```

```
     */
public void startCamera() {
        initThread();
thread.start();
    }

/**
     * @return зачение индикатора выполнения потока
     */
public boolean isRunning() {
return isRunning;
    }

/**
     * @return индикатор запуска детекции
     */
public boolean isDetectionRunning() {
return recognizer != null &&recognizer.isRunning();
    }

/**
     * Остановка камеры
     */
public void close() {
        stopDetection();
isRunning = false;
try {
while (thread != null &&thread.isAlive()) {
thread.join();
            }
        } catch (InterruptedException e1) {

        }
    }

/**
     * @return распознаватель
     */
public Recognizer getRecognizer() {
return recognizer;
    }

/**
     * @return холст для изображений
     */
public Canvas getCanvas() {
return canvas.getCanvas();
```

```
    }

/**
     * Отрисовка областей детекции
     *
     * @param canvas холст
     */
private void drawDetections(CanvasFrame canvas) {

long total = recognizer.getDetections().size();
        Dimension canvasSize = canvas.getCanvasSize();

for (int i = 0; i < total; i++) {
            opencv_core.Rect rect = recognizer.getDetections().get(i);

            BufferedImage image = new BufferedImage(recognizer.getMat().arrayWidth(),
recognizer.getMat().arrayHeight(),
                    BufferedImage.TYPE_INT_ARGB_PRE);
            Graphics2D g = (Graphics2D) image.getGraphics();
            g.setStroke(new BasicStroke(5));
            g.setColor(new Color(0, 255, 0));
            g.drawRect(rect.x(), rect.y(), rect.width(), rect.height());
            ((Graphics2D)
canvas.getCanvas().getGraphics()).setComposite(AlphaComposite.Clear);
            canvas.getCanvas().getGraphics().drawImage(image,
                    0, 0, (int) canvasSize.getWidth(), (int) canvasSize.getHeight(),
null);
            g.dispose();
        }
    }
}
```

```
package cv;

import org.bytedeco.javacpp.opencv_core;

import java.util.Date;

import static org.bytedeco.javacpp.opencv_imgcodecs.imwrite;

/**
 * Трекер распознавания лиц
 */
public class FaceTracker {
```

```
/**
     * Таймер периодических срабаытваний
     */
private static PeriodicTimer timer;

/**
     * Обратный вызов детекции
     */
public interface SaveCallback {

/**
         * Обработчик вызова детекции
         *
         * @param path путь к файлу
         */
void callback(String path);
    }

/**
     * Запуск трекинга распознавания
     *
     * @param sec          интервал в секундах
     * @param path         путь к папке с файлами
     * @param recognizer   распозванатель
     * @param saveCallback обработчик обратного вызова
     */
public static void startTracking(int sec, String path, Recognizer recognizer,
SaveCallback saveCallback) {
if (timer == null) {
timer = new PeriodicTimer();
        }
timer.run(sec, () -> {
            opencv_core.Mat recognizedMat = recognizer.getRecognizedMat();
if (recognizedMat == null) {
return false;
            }
            String fullPath = path + new Date().getTime() + ".jpg";
imwrite(fullPath, recognizedMat);
            saveCallback.callback(fullPath);
return true;
        });
    }

/**
     * Остановка трекинга
     */
```

```
public static void stopTracking() {
if (timer != null) {
timer.stop();
        }
    }
}
```

```
package cv;

/**
 * Таймерпериодическоговызова
 */
class PeriodicTimer {

public interface ConditionalRunnable {

boolean run();
    }

private Thread thread;

private int sleepSeconds;

private ConditionalRunnable action;

private volatile boolean isRunning;

private Runnable runnable = () -> {
long start = System.currentTimeMillis();
while (isRunning) {
if ((System.currentTimeMillis() - start) / 1000 >sleepSeconds &&action.run()) {
                start = System.currentTimeMillis();
            }
        }
    };

/**
     * Запусктаймера
     *
     * @param sleepSeconds период
     * @param action       действие
     */
public void run(int sleepSeconds, ConditionalRunnable action) {
```

```
        stop();
this.sleepSeconds = sleepSeconds;
this.action = action;
isRunning = true;
thread = new Thread(runnable);
thread.start();
    }

/**
     * Остановкатаймера
     */
public void stop() {
if (thread != null) {
isRunning = false;
while (thread.isAlive()) {
try {
                    thread.join();
                } catch (InterruptedException e) {
// nothing
}
            }
        }
    }
}
```

```
package cv;


import org.bytedeco.javacpp.opencv_core;
import org.bytedeco.javacpp.opencv_objdetect;
import org.bytedeco.javacv.CanvasFrame;

import java.awt.*;
import java.awt.image.BufferedImage;

import static org.bytedeco.javacpp.opencv_imgproc.*;
import static org.bytedeco.javacpp.opencv_objdetect.CV_HAAR_DO_CANNY_PRUNING;
import static org.bytedeco.javacpp.opencv_objdetect.CV_HAAR_FIND_BIGGEST_OBJECT;

/**
 * Распознавательлиц
 */
public class Recognizer {
```

```
private Thread workingThread;

private volatile boolean isRunning;

/**
     * Областидетекции
     */
private volatile opencv_core.RectVector detections = new opencv_core.RectVector();

/**
     * Заглушкапустойдетекции
     */
private opencv_core.RectVector emptyDetections = new opencv_core.RectVector();

private opencv_core.Mat mat;

private opencv_core.Mat recognizedMat;

private CanvasFrame canvasFrame;

private opencv_objdetect.CascadeClassifier classifier;

/**
     * Циклдетекции
     */
private Runnable runnable = () -> {
while (isRunning) {
            opencv_core.Mat videoMatGray = new opencv_core.Mat();
if (getMat() == null) {
continue;
            }
            cvtColor(getMat(), videoMatGray, COLOR_BGRA2GRAY);
            equalizeHist(videoMatGray, videoMatGray);
            opencv_core.RectVector rectVector = new opencv_core.RectVector();
            classifier.detectMultiScale(videoMatGray, rectVector, 1.1, 7,
                    CV_HAAR_DO_CANNY_PRUNING | CV_HAAR_FIND_BIGGEST_OBJECT, new
opencv_core.Size(48, 48), new opencv_core.Size());
            setDetections(rectVector);

long total = getDetections().size();
            Dimension canvasSize = canvasFrame.getCanvasSize();
            recognizedMat = total > 0 ? getMat() : null;

for (int i = 0; i < total; i++) {
                opencv_core.Rect rect = getDetections().get(i);

                BufferedImage image = new BufferedImage(getMat().arrayWidth(),
```

```
getMat().arrayHeight(), BufferedImage.TYPE_INT_ARGB_PRE);
                Graphics2D g = (Graphics2D) image.getGraphics();
                g.setStroke(new BasicStroke(5));
                g.setColor(new Color(0, 255, 0));
                g.drawRect(rect.x(), rect.y(), rect.width(), rect.height());
                ((Graphics2D)
canvasFrame.getCanvas().getGraphics()).setComposite(AlphaComposite.Clear);
                canvasFrame.getCanvas().getGraphics().drawImage(image,
                        0, 0, (int) canvasSize.getWidth(), (int) canvasSize.getHeight(),
null);
                g.dispose();
            }
        }
    };


public Recognizer(CanvasFrame canvasFrame, opencv_objdetect.CascadeClassifier classifier)
{
this.canvasFrame = canvasFrame;
this.classifier = classifier;
    }

public opencv_core.Mat getRecognizedMat() {
return recognizedMat;
    }

public void startDetection() {
        isRunning = true;
        workingThread = new Thread(runnable);
        workingThread.start();
    }

public void stop() {
        isRunning = false;
while (workingThread != null && workingThread.isAlive()) {
try {
                workingThread.join();
            } catch (InterruptedException e) {
// already interrupted
}
        }
    }

public boolean isRunning() {
return isRunning;
    }
```

```
public synchronized void setMat(opencv_core.Mat mat) {
this.mat = mat;
    }

public synchronized opencv_core.Mat getMat() {
return mat;
    }

public synchronized void setDetections(opencv_core.RectVector rectVector) {
this.detections = rectVector;
    }

public synchronized opencv_core.RectVector getDetections() {
return isRunning ? detections : emptyDetections;
    }
}
```

```
package component.tracker;

import data.TrackingData;
import state.TrackerRecordsManager;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.util.Vector;

/**
 * Панельтаблицысрезультатамимониторинга
 */
class TrackerGridPanel extends JPanel {

/**
     * Слушательвыборастрокирезультатамониторинга
     */
public interface SelectListener {

/**
         * Обработиквыборастрокисрезультатами
         *
         * @param path путькизображению
         */
void onSelect(String path);
    }

/**
     * Модельданныхтаблицы
     */
private DefaultTableModel tableModel;

/**
     * Таблица
     */
private JTable table;

/**
     * Конструкторпанели. Начальныеустановки
     */
TrackerGridPanel() {
```

```
super();
        setLayout(new BorderLayout());
        initTableModel();
table = new JTable(tableModel);
        JScrollPane scrollPane = new JScrollPane(table);
        add(scrollPane, BorderLayout.CENTER);

table.getColumnModel().getColumn(1).setCellRenderer(new DefaultTableCellRenderer() {

    SimpleDateFormat f = new SimpleDateFormat("dd.MM.yy HH:mm:ss");

public Component getTableCellRendererComponent(JTable table,
                                                    Object value, boolean isSelected,
boolean hasFocus,
int row, int column) {
if (value instanceof Date) {
            value = f.format(value);
        }
return super.getTableCellRendererComponent(table, value, isSelected,
                hasFocus, row, column);
    }
});

        TrackerRecordsManager.getInstance().addListener(tableModel::addRow);
    }

/**
     * Установкаобработчикавыборастрокимониторинга
     *
     * @param selectListener обработчик
     */
void setSelectListener(SelectListener selectListener) {
        ListSelectionModel cellSelectionModel = table.getSelectionModel();
        cellSelectionModel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        cellSelectionModel.addListSelectionListener(e -> {

int[] selectedRow = table.getSelectedRows();

            String imagePath = (String) table.getValueAt(selectedRow[0], 2);
            selectListener.onSelect(imagePath);
        });
    }

/**
     * Инициализациямоделиданныхтаблицы
     */
private void initTableModel() {
        TrackingData.deleteInvalidTracking();
        Vector<Object>[] array = TrackingData.fetchTrackingData();

        Vector<Object> columnNames = array[0];
        Vector<Object> data = array[1];
tableModel = new DefaultTableModel(data, columnNames) {

            @Override
public Class getColumnClass(int column) {
for (int row = 0; row < getRowCount(); row++) {
                    Object o = getValueAt(row, column);
if (o != null) {
return o.getClass();
                    }
                }
return Object.class;
            }
        };
    }
}
```

```
package component.tracker;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

/**
 * Панельмониторингараспознавания. Содержиттаблицуипросмотрщикизображений
 */
public class TrackerPanel extends JPanel {

/**
     * Выбранноеизображение
     */
private BufferedImage selectedImage;


/**
     * Конструктор.
     */
public TrackerPanel() {
super();
        setLayout(new BorderLayout());

// Установкаперерисовкисодержимогоприобновлении
JPanel viewerPanel = new JPanel(new BorderLayout()) {

            @Override
protected void paintComponent(Graphics g) {
                drawImage(g, getSize());
            }
        };
        add(viewerPanel, BorderLayout.CENTER);
        TrackerGridPanel trackerGridPanel = new TrackerGridPanel();
        trackerGridPanel.setPreferredSize(new Dimension(400, getHeight()));

// обработчиквыборастроки. считываетизображениеиотображает
trackerGridPanel.setSelectListener(path -> {
            File f = new File(path);
if (f.exists()) {
try {
selectedImage = ImageIO.read(f);
                    drawImage(viewerPanel.getGraphics(), viewerPanel.getSize());
                } catch (IOException e) {
throw new RuntimeException(e);
                }
            }
        });
        add(trackerGridPanel, BorderLayout.EAST);
    }

/**
     * Прорисовкаизображения
     *
     * @param g         графическийконтекст
     * @param dimension размерыдоступногопространства
     */
private void drawImage(Graphics g, Dimension dimension) {
        g.fillRect(0, 0, (int) dimension.getWidth(), (int) dimension.getHeight());
if (selectedImage != null) {
            Dimension d = getScaledDimension(
new Dimension(selectedImage.getWidth(), selectedImage.getHeight()), dimension);
            Point start = getStartingPoint(d, dimension);
```

```
            g.drawImage(selectedImage, start.x, start.y, d.width, d.height, null);
        }
    }

/**
     * Определениеначальногоположения
     *
     * @param imgSize  размеризображения
     * @param boundary доступноепространство
     * @return точкуначальногоположения
     */
private Point getStartingPoint(Dimension imgSize, Dimension boundary) {
return new Point((boundary.width - imgSize.width) / 2, (boundary.height - imgSize.height)
/ 2);
    }

/**
     * Масштабированиеизображенияпоразмерупанели
     *
     * @param imgSize  размеризображения
     * @param boundary доступноепространство
     * @return размерыизображения
     */
private Dimension getScaledDimension(Dimension imgSize, Dimension boundary) {
int originalWidth = imgSize.width;
int originalHeight = imgSize.height;
int boundWidth = boundary.width;
int boundHeight = boundary.height;
int newWidth;
int newHeight;

        newWidth = boundWidth;
        newHeight = (newWidth * originalHeight) / originalWidth;


if (newHeight > boundHeight) {
            newHeight = boundHeight;
            newWidth = (newHeight * originalWidth) / originalHeight;
        }
return new Dimension(newWidth, newHeight);
    }
}
```

```
package component;


import config.ConfigReader;
import config.ControlConfig;
import cv.CameraCanvas;
import cv.FaceTracker;
import data.TrackingData;
import state.TrackerRecordsManager;

import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.event.ChangeListener;
import java.awt.*;
import java.io.File;
import java.util.Date;

import static component.Utils.getIcon;

/**
 * Контрольнаяпанельуправлениякамерой
 */
public class ControlPanel extends JPanel {
```

```
/**
     * Конфигурациядляконтролов
     */
private ControlConfig controlConfig =
ConfigReader.getInstance().getConfig(ControlConfig.class);

/**
     * Путьсохраненияизображений
     */
private JTextField pathText;

/**
     * Контролинтерваладетекции
     */
private JSpinner trackSpinner;

/**
     * Состояниемониторинга
     */
private boolean monitoringState;

/**
     * Холсткамеры
     */
private CameraCanvas cc;

/**
     * Контролуправлениияркостью
     */
private JSlider brightness;

/**
     * Контролуправленияконтрастностью
     */
private JSlider contrast;

/**
     * Контролуправлениянасыщенностью
     */
private JSlider saturation;

/**
     * Кнопказапускадетекции
     */
private JButton playButton;

/**
     * Кнопкаостановкидетекции
     */
private JButton pauseButton;

/**
     * Контролвключениямониторинга
     */
private JCheckBox monitoringCheckbox = new
JCheckBox(controlConfig.getValue(ControlConfig.MONITORING));

/**
     * Контструктор
     *
     * @param cc холсткамеры
     */
public ControlPanel(CameraCanvas cc) {
super();
this.cc = cc;
        setLayout(new BorderLayout());

        JPanel topPanel = new JPanel();
        topPanel.setLayout(new GridLayout(0, 2));
```

```
        topPanel.add(initButtonPanel());
        topPanel.add(initSlidersPanel());

        JPanel trackPanel = new JPanel();
trackSpinner = new JSpinner(new SpinnerNumberModel(5, 5, 50, 1));
pathText = new JTextField();
pathText.setEditable(false);
pathText.setColumns(30);
        JButton selectFile = new
JButton(controlConfig.getValue(ControlConfig.SELECT_FOLDER));
        selectFile.addActionListener(e -> {
            JFileChooser fc = new JFileChooser();
            fc.setAcceptAllFileFilterUsed(false);
            fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
if (!"".equals(getTrackerPath())) {
                fc.setCurrentDirectory(new File(getTrackerPath()));
            }
if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
pathText.setText(fc.getSelectedFile().getAbsolutePath());
            }
        });
monitoringCheckbox.addChangeListener(e -> {
            updateMonitoringState(monitoringCheckbox.isSelected());
        });
        updateMonitoringState(monitoringCheckbox.isSelected());
        trackPanel.add(monitoringCheckbox);
        trackPanel.add(trackSpinner);
        trackPanel.add(pathText);
        trackPanel.add(selectFile);

        add(topPanel, BorderLayout.NORTH);
        add(trackPanel, BorderLayout.SOUTH);
    }

/**
     * Обновлениесостояниямониторинга
     *
     * @param state состояние
     */
private void updateMonitoringState(boolean state) {
monitoringState = state;
if (monitoringState && !"".equals(getTrackerPath())) {
            FaceTracker.startTracking(getMonitoringSeconds(), getTrackerPath(),
cc.getRecognizer(), (path) ->

TrackerRecordsManager.getInstance().add(TrackingData.addTrackRecord(new Date(), path)));
        } else {
            FaceTracker.stopTracking();
        }
    }

/**
     * @return состояниемониторинга
     */
public boolean getMonitoringState() {
return monitoringState;
    }

/**
     * Устанавливаетсостояниемониторинга
     *
     * @param state состояниемониторинга
     */
public void setMonitoringState(boolean state) {
        updateMonitoringState(state);
    }

/**
     * @return интервалдетекциимониторинга
     */
```

```
public int getMonitoringSeconds() {
return (int) trackSpinner.getValue();
    }

/**
     * Устанавливаетинтервалмониторинга
     *
     * @param v секунды
     */
public void setMonitoringSeconds(int v) {
trackSpinner.setValue(v);
    }

/**
     * Устанавливаетпутьсохранениякадров
     *
     * @param p путь
     */
public void setTrackerPath(String p) {
pathText.setText(p);
    }

/**
     * @return путьсохранениякадров
     */
public String getTrackerPath() {
if ("".equals(pathText.getText())) {
return "";
        }
return pathText.getText().endsWith(File.separator) ? pathText.getText() :
pathText.getText() + File.separator;
    }

/**
     * Инициализацияпанеликнопок
     *
     * @return панель
     */
private JPanel initButtonPanel() {
        JPanel buttonPanel = new JPanel(new GridLayout(0, 2, 10, 10));
        ControlConfig controlConfig =
ConfigReader.getInstance().getConfig(ControlConfig.class);
playButton = new JButton(controlConfig.getValue(ControlConfig.START_DETECTION),
getIcon("play.png"));
pauseButton = new JButton(controlConfig.getValue(ControlConfig.STOP_DETECTION),
getIcon("pause.png"));

playButton.addActionListener(e -> {
            setDetectionState(true);
            updateButtonsState();
        });

pauseButton.addActionListener(e -> {
            setDetectionState(false);
            updateButtonsState();
        });
        updateButtonsState();
        buttonPanel.setBorder(new EmptyBorder(10, 10, 10, 10));
        buttonPanel.add(playButton);
        buttonPanel.add(pauseButton);

return buttonPanel;
    }

/**
     * @return включеналидетекция
     */
private boolean isDetectionEnabled() {
return cc.isDetectionRunning();
    }
```

```
/**
     * Устанавливаетсостояниедетекции
     *
     * @param state состояние
     */
private void setDetectionState(boolean state) {
if (state) {
cc.startDetection();
        } else {
cc.stopDetection();
        }
    }

/**
     * Обновляетсостояниеконопокдетекции
     */
public void updateButtonsState() {
playButton.setEnabled(!isDetectionEnabled());
pauseButton.setEnabled(isDetectionEnabled());
monitoringCheckbox.setEnabled(isDetectionEnabled());
if (!isDetectionEnabled()) {
monitoringCheckbox.setSelected(false);
        }
    }

/**
     * Установкапанелислайдеровкамеры
     *
     * @return панель
     */
private JPanel initSlidersPanel() {
        JPanel slidersPanel = new JPanel(new GridLayout(0, 2));

        ControlConfig controlConfig =
ConfigReader.getInstance().getConfig(ControlConfig.class);

brightness = createSliderControl(0, 100, e ->
cc.setBrightness(((JSlider) e.getSource()).getValue()));
        slidersPanel.add(new JLabel(controlConfig.getBrightness(),
getIcon("brightness_128.png", 32), SwingConstants.LEFT));
        slidersPanel.add(brightness);

contrast = createSliderControl(0, 100, e ->
cc.setContrast(((JSlider) e.getSource()).getValue()));
        slidersPanel.add(new JLabel(controlConfig.getContrast(),
getIcon("contrast_128.png", 32), SwingConstants.LEFT));
        slidersPanel.add(contrast);

saturation = createSliderControl(0, 100, e ->
cc.setSaturation(((JSlider) e.getSource()).getValue()));
        slidersPanel.add(new JLabel(controlConfig.getSaturation(),
getIcon("saturation_128.png", 32), SwingConstants.LEFT));
        slidersPanel.add(saturation);
return slidersPanel;
    }

/**
     * Обновляетзначенияслайдеров
     */
public void updateViewSliders() {
brightness.setValue((int) cc.getBrightness());
contrast.setValue((int) cc.getContrast());
saturation.setValue((int) cc.getSaturation());
    }

/**
     * Создаетэлементуправления - слайдер
     *
     * @param min      минимальноезначение
```

```
     * @param max      максимальноезначение
     * @param listener обработчикизменений
     * @return созданныйслайдер
     */
private JSlider createSliderControl(int min, int max, ChangeListener listener) {
        JSlider slider = new JSlider(JSlider.HORIZONTAL, min, max, max / 2);
        slider.setPaintLabels(true);
        slider.setPaintTicks(true);
        slider.setMajorTickSpacing(50);
        slider.setMinorTickSpacing(10);
        slider.addChangeListener(listener);
return slider;
    }
}
```

```
package component;


import config.ConfigReader;
import config.ControlConfig;
import config.MenuConfig;
import cv.CameraCanvas;
import data.SetupData;

import javax.swing.*;
import java.awt.*;
import java.awt.event.WindowEvent;

/**
 * Панельменю
 */
public class MenuBar extends JMenuBar {

private MenuConfig menuConfig = ConfigReader.getInstance().getConfig(MenuConfig.class);
private ControlConfig controlConfig =
ConfigReader.getInstance().getConfig(ControlConfig.class);

/**
     * Создаетпанельменю
     *
     * @param parent       родительскийкомпонент
     * @param cc           холсткамеры
     * @param controlPanel контрольнаяпанель
     */
public MenuBar(JFrame parent, CameraCanvas cc, ControlPanel controlPanel) {
super();
        JMenu setupsMenu = new JMenu(menuConfig.getSetups());

        JMenuItem addSetup = new JMenuItem(menuConfig.getValue(MenuConfig.ADD_SETUP));
        JMenuItem loadSetup = new JMenuItem(menuConfig.getValue(MenuConfig.LOAD_SETUP));
        JMenuItem removeSetup = new
JMenuItem(menuConfig.getValue(MenuConfig.REMOVE_SETUP));

        addSetup.addActionListener(e -> {
            String setupName = (String) JOptionPane.showInputDialog(
this,
"Названиенастройки",
                    menuConfig.getValue(MenuConfig.ADD_SETUP),
                    JOptionPane.PLAIN_MESSAGE,
null,
null,
"");
            SetupData.addSetup(SetupData.contructSetupObject(setupName,
cc.getSaturation(),
                    cc.getBrightness(), cc.getContrast(),
controlPanel.getMonitoringState(),
                    controlPanel.getMonitoringSeconds(), controlPanel.getTrackerPath()));
```

```
        });

        loadSetup.addActionListener(e -> {
            JList<SetupData.SetupObject> list = initList();
            JButton loadButton = new
JButton(controlConfig.getValue(ControlConfig.BUTTON_LOAD));
            loadButton.addActionListener(e1 -> {
                SetupData.SetupObject so = list.getSelectedValue();

                cc.setSaturation(so.getSaturation());
                cc.setBrightness(so.getBrightness());
                cc.setContrast(so.getContrast());
                controlPanel.setMonitoringSeconds(so.getTrackPeriod());
                controlPanel.setTrackerPath(so.getTrackPath());
                controlPanel.setMonitoringState(so.isTrackState());

                controlPanel.updateViewSliders();

                Window w = SwingUtilities.windowForComponent(loadButton);
                w.dispatchEvent(new WindowEvent(w, WindowEvent.WINDOW_CLOSING));

            });
            initListBoxDialog(parent, menuConfig.getValue(MenuConfig.LOAD_SETUP), list,
loadButton);
        });

        removeSetup.addActionListener(e -> {
            JList<SetupData.SetupObject> list = initList();
            JButton removeButton = new
JButton(controlConfig.getValue(ControlConfig.BUTTON_REMOVE));
            removeButton.addActionListener(e1 -> {
                SetupData.SetupObject so = list.getSelectedValue();
                SetupData.removeSetup(so.getId());
                DefaultListModel<SetupData.SetupObject> model =
(DefaultListModel<SetupData.SetupObject>) list.getModel();
                model.removeElement(so);
            });
            initListBoxDialog(parent, menuConfig.getValue(MenuConfig.REMOVE_SETUP), list,
removeButton);
        });

        setupsMenu.add(addSetup);
        setupsMenu.add(loadSetup);
        setupsMenu.add(removeSetup);
        add(setupsMenu);
    }

/**
     * Инициализируетсписокнастроек
     *
     * @return список
     */
private JList<SetupData.SetupObject> initList() {
        JList<SetupData.SetupObject> list = new JList<>(loadData());
        list.setSelectionMode(ListSelectionModel.SINGLE_INTERVAL_SELECTION);
        list.setLayoutOrientation(JList.VERTICAL);
        list.setVisibleRowCount(-1);
return list;
    }

/**
     * Инициализациядиалоговогоокнасосписокомнастроек
     *
     * @param parent       родительскийкрмпонент
     * @param title        заголовок
     * @param list         список
     * @param actionButton кнопкаподтвержения
     */
private void initListBoxDialog(JFrame parent, String title, JList list, JButton
actionButton) {
```

```
        JScrollPane listScroller = new JScrollPane(list);
        JDialog dialog = new JDialog(parent, title, true);

        JPanel buttonPanel = new JPanel(new GridLayout(0, 2));

        JButton closeButton = new
JButton(controlConfig.getValue(ControlConfig.BUTTON_CANCEL));
        closeButton.addActionListener(e1 -> {
            dialog.dispatchEvent(new WindowEvent(dialog, WindowEvent.WINDOW_CLOSING));
        });
        buttonPanel.add(actionButton);
        buttonPanel.add(closeButton);

        dialog.setLayout(new BorderLayout());
        dialog.add(listScroller, BorderLayout.CENTER);
        dialog.add(buttonPanel, BorderLayout.SOUTH);
        dialog.setPreferredSize(new Dimension(400, 200));
        dialog.pack();

        dialog.setLocationRelativeTo(parent);
        dialog.setVisible(true);
    }

/**
     * Загрузканастроек
     *
     * @return модельсписка
     */
private ListModel<SetupData.SetupObject> loadData() {
final DefaultListModel<SetupData.SetupObject> listModel = new DefaultListModel<>();
        SetupData.fetchAll().forEach(listModel::addElement);
return listModel;
    }
}
```

```
package component;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;

/**
 * Вспомагательныеметоды
 */
public class Utils {

/**
     * Загрузкииконки
     *
     * @param iconName название
     * @return иконка
     */
public static Icon getIcon(String iconName) {
try {
return new ImageIcon(getScaledImage(ImageIO.read(
                    Utils.class.getClassLoader().getResource("resource/icon/" +
iconName)), 40, 40));
        } catch (Exception e) {
return null;
        }
    }

/**
     * Загрузкаиконкисмасштабированием
     *
     * @param iconName названиеиконки
```

```
     * @param wh размер
     * @return иконку
     */
public static Icon getIcon(String iconName, int wh) {
try {
return new ImageIcon(getScaledImage(ImageIO.read(
                    Utils.class.getClassLoader().getResource("resource/icon/" +
iconName)), wh, wh));
        } catch (Exception e) {
return null;
        }
    }

/**
     * Получаетмасштабированноеизображение
     *
     * @param srcImg изображение
     * @param w ширина
     * @param h высота
     * @return масштабированноеизображение
     */
public static Image getScaledImage(Image srcImg, int w, int h) {
        BufferedImage resizedImg = new BufferedImage(w, h, BufferedImage.TYPE_INT_ARGB);
        Graphics2D g2 = resizedImg.createGraphics();

        g2.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
RenderingHints.VALUE_INTERPOLATION_BILINEAR);
        g2.drawImage(srcImg, 0, 0, w, h, null);
        g2.dispose();

return resizedImg;
    }
}
```

```
package data;


import config.ConfigReader;
import config.DatabaseConfig;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

/**
 * ПомощникработысБД
 */
public class DBHelper {

/**
     * НазваниеБД
     */
private static final String DBNAME = "deviant";

/**
     * СкриптиспользованияБД
     */
private static final String USE_DB = "USE " + DBNAME;

/**
     * СкриптсозданияБД
     */
private static final String CREATE_DB = "CREATE DATABASE IF NOT EXISTS " + DBNAME;

/**
```

```
     * Исполнительзапросов
     */
public interface QueryExecutor {

/**
         * Исполнениезапроса
         *
         * @param statement запрос
         * @throws SQLException ошибка
         */
void execute(Statement statement) throws SQLException;

    }

/**
     * ИнициализацияБД
     */
static {
initDatabase();
    }

/**
     * ИнициализацияБД
     */
private static void initDatabase() {
executeQuery(statement -> {
            statement.execute(CREATE_DB);
            statement.execute(USE_DB);
        });
    }

/**
     * Выполнениезапроса
     *
     * @param qe исполнительзапросов
     */
protected static void executeQuery(QueryExecutor qe) {
        DatabaseConfig databaseConfig =
ConfigReader.getInstance().getConfig(DatabaseConfig.class);
try {
            Class.forName(databaseConfig.getDriver());
            Connection connection =
                    DriverManager.getConnection(databaseConfig.getUrl(),
databaseConfig.getUser(), databaseConfig.getPassword());
            Statement stmt = connection.createStatement();
            stmt.execute(USE_DB);
try {
                qe.execute(stmt);
            } finally {
                stmt.close();
                connection.close();
            }
        } catch (SQLException e) {
throw new RuntimeException("Database connection problem. Check setups", e);
        } catch (ClassNotFoundException e) {
throw new RuntimeException("Unable to find class: " + databaseConfig.getDriver());
        }
    }
}
```

```
package data;


import org.apache.commons.lang.StringEscapeUtils;

import java.sql.ResultSet;
import java.text.MessageFormat;
```

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

public class SetupData {

private static final String TABLE_NAME = "setups";

private static final String CREATE_TABLE = "create table if not exists `" + TABLE_NAME +
"` (" +
"`id` int(9) not null AUTO_INCREMENT PRIMARY KEY, " +
"`setup_name` varchar(128)," +
"`saturation` double," +
"`brightness` double," +
"`contrast` double," +
"`track_state` boolean," +
"`track_period` int," +
"`track_path` varchar(512))";

private static final String INSERT = "insert into `" + TABLE_NAME +
"` (setup_name, saturation, brightness, contrast, " +
"track_state, track_period, track_path) values ({0})";

private static final String DELETE = "delete from `" + TABLE_NAME + "` where id = {0}";

private static final MessageFormat SELECT = new MessageFormat("select * from `" +
TABLE_NAME + "` where id = {0}");

private static final String SELECT_ALL = "select * from `" + TABLE_NAME + "`";

static {
        initSetupData();
    }

public static class SetupObject {

private String setupName;

private double saturation;

private double brightness;

private double contrast;

private boolean trackState;

private int trackPeriod;

private String trackPath;

private Long id;

public String getSetupName() {
return setupName;
        }

public void setSetupName(String setupName) {
this.setupName = setupName;
        }

public double getSaturation() {
return saturation;
        }

public void setSaturation(double saturation) {
this.saturation = saturation;
        }

public double getBrightness() {
return brightness;
```

```
        }

public void setBrightness(double brightness) {
this.brightness = brightness;
        }

public double getContrast() {
return contrast;
        }

public void setContrast(double contrast) {
this.contrast = contrast;
        }

public boolean isTrackState() {
return trackState;
        }

public void setTrackState(boolean trackState) {
this.trackState = trackState;
        }

public int getTrackPeriod() {
return trackPeriod;
        }

public void setTrackPeriod(int trackPeriod) {
this.trackPeriod = trackPeriod;
        }

public String getTrackPath() {
return trackPath;
        }

public void setTrackPath(String trackPath) {
this.trackPath = trackPath;
        }

public Long getId() {
return id;
        }

public void setId(Long id) {
this.id = id;
        }

        @Override
public String toString() {
return getSetupName();
        }
    }

public static SetupObject contructSetupObject(String name, double saturation, double
brightness, double contrast,
boolean trackState, int trackPeriod, String trackPath) {
        SetupObject o = new SetupObject();
        o.setSetupName(name);
        o.setSaturation(saturation);
        o.setBrightness(brightness);
        o.setContrast(contrast);
        o.setTrackState(trackState);
        o.setTrackPeriod(trackPeriod);
        o.setTrackPath(trackPath);
return o;
    }

private static String destructSetupObject(SetupObject s) {
return "'" + StringEscapeUtils.escapeJava(s.getSetupName()) + "', " +
                s.getSaturation() + ", " +
                s.getBrightness() + ", " +
```

```
                s.getContrast() + ", " +
                s.isTrackState() + ", " +
                s.getTrackPeriod() + ", '" +
                StringEscapeUtils.escapeJava(s.getTrackPath()) + "'";
    }

private static void initSetupData() {
        DBHelper.executeQuery(s -> s.execute(CREATE_TABLE));
    }

public static SetupObject fetchSetup(Long id) {
        List<SetupObject> l = new ArrayList<>();
        DBHelper.executeQuery(s ->  {
            ResultSet rs = s.executeQuery(SELECT.format(id));
if (rs.next()) {
                l.add(contructSetupObject(rs.getString(1), rs.getInt(2), rs.getInt(3),
                        rs.getInt(4), rs.getBoolean(5), rs.getInt(6),
                        rs.getString(7)));
            }
        });
return l.isEmpty() ? null : l.iterator().next();
    }

public static Collection<SetupObject> fetchAll() {
        Collection<SetupObject> list = new ArrayList<>();
        DBHelper.executeQuery(s ->  {
            ResultSet rs = s.executeQuery(SELECT_ALL);
while (rs.next()) {
                SetupObject so = contructSetupObject(rs.getString(2), rs.getInt(3),
rs.getInt(4),
                        rs.getInt(5), rs.getBoolean(6), rs.getInt(7),
                        rs.getString(8));
                so.setId(rs.getLong(1));
                list.add(so);
            }
        });
return list;
    }

public static void addSetup(SetupObject setupObject) {
        DBHelper.executeQuery(s -> s.execute(MessageFormat.format(INSERT,
destructSetupObject(setupObject))));
    }

public static void removeSetup(Long id) {
        DBHelper.executeQuery(s -> s.execute(MessageFormat.format(DELETE, id)));
    }

}
```

```
package data;


import config.ConfigReader;
import config.ControlConfig;
import org.apache.commons.lang.StringEscapeUtils;

import java.io.File;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Vector;

public class TrackingData {

private static final String TABLE_NAME = "detection";

private static final String QUERY = "select id, rec_time, path from `" + TABLE_NAME +
"`";

private static final String INSERT_QUERY = "insert into `" + TABLE_NAME + "` (rec_time,
path) " +
"values (STR_TO_DATE(''{0}'', ''%d.%m.%Y %H:%i:%s''), ''{1}'')";

private static final String CREATE_TABLE = "create table if not exists `detection` (" +
"`id` int(9) not null AUTO_INCREMENT PRIMARY KEY, " +
"`rec_time` datetime," +
"`path` varchar(512))";

private static final String DELETE_QUERY ="delete from `" + TABLE_NAME + "` where id in
({0})";

static {
initTrackingData();
    }

private static void initTrackingData() {
        DBHelper.executeQuery(s -> s.execute(CREATE_TABLE));
    }


public static Vector<Object>[] fetchTrackingData() {
initTrackingData();

        Vector<Object> columnNames = new Vector<>();
        Vector<Object> data = new Vector<>();
        DBHelper.executeQuery(statement -> {
            statement.execute(CREATE_TABLE);
            ResultSet rs = statement.executeQuery(QUERY);
            ResultSetMetaData md = rs.getMetaData();

            ControlConfig controlConfig =
ConfigReader.getInstance().getConfig(ControlConfig.class);
int columns = md.getColumnCount();
for (int i = 1; i <= columns; i++) {
                columnNames.addElement(controlConfig.getValue(md.getColumnName(i)));
            }

while (rs.next()) {
                Vector<Object> row = new Vector<>(columns);
for (int i = 1; i <= columns; i++) {
                    row.addElement(rs.getObject(i));
                }
                data.addElement(row);
            }
        });
return new Vector[] {columnNames, data};
    }

public static Vector<Object> addTrackRecord(java.util.Date date, String path) {
        Vector<Object> row = new Vector<>();
final String p = StringEscapeUtils.escapeJava(path);
        DBHelper.executeQuery(statement -> {
           statement.executeUpdate(
                    MessageFormat.format(INSERT_QUERY, new SimpleDateFormat("dd.MM.yyyy
HH:mm:ss").format(date), p),
                    Statement.RETURN_GENERATED_KEYS);
            ResultSet rs = statement.getGeneratedKeys();
            rs.next();
            row.addElement(rs.getLong(1));
            row.addElement(date);
```

```
            row.addElement(path);
        });
return row;
    }

public static void deleteInvalidTracking() {
        List<Long> ids = new ArrayList<>();
        DBHelper.executeQuery(statement -> {
            ResultSet rs = statement.executeQuery(QUERY);
while (rs.next()) {
                String path = (String) rs.getObject("path");
if (!new File(path).exists()) {
                    ids.add(rs.getLong("id"));
                }
            }
            ids.stream().map(e -> e + "").reduce((acc, s) -> acc + ", " + s)
                    .ifPresent(s -> {
try {
                            statement.execute(MessageFormat.format(DELETE_QUERY, s));
                        } catch (SQLException e) {
throw new RuntimeException(e);
                        }
                    });
        });
    }
}
```

```
package state;

import data.TrackingData;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Vector;

public class TrackerRecordsManager {

public interface RecordsUpdateListener {

void onRecordsUpdated(Vector<Object> v);
    }

static {
        TrackingData.deleteInvalidTracking();
    }

private static TrackerRecordsManagertrm;

private TrackerRecordsManager() {
    }

public static TrackerRecordsManager getInstance() {
if (trm == null) {
trm = new TrackerRecordsManager();
        }
return trm;
    }

private List<RecordsUpdateListener>listeners = new ArrayList<>();

public void addListener(RecordsUpdateListener listener) {
listeners.add(listener);
    }

public void removeListener(RecordsUpdateListener listener) {
listeners.remove(listener);
```

```
    }

private void fireListeners(Vector<Object> v) {
listeners.forEach(listener -> listener.onRecordsUpdated(v));
    }

private List<Vector<Object>>array = new
ArrayList<>(Arrays.asList(TrackingData.fetchTrackingData()));

public void add(Vector<Object> v) {
array.add(v);
        fireListeners(v);
    }
}
```