МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Информационных систем и систем безопасности

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(специалитет)

На тему: Разработка модели обеспечения безопасности на основе алгоритма распознавания лиц

Исполнитель: Закиров Илья Ильясович

(фамилия, имя, отчество)

Руководитель: Переспелов Анатолий Витальевич

(фамилия, имя, отчество)

Санкт-Петербург

2023 г

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной работе объектом исследования является область применения и анализ работы алгоритма распознавания лиц с точки зрения информационной безопасности.

В нынешний момент времени большинство предприятий и организаций уже начали использовать в своей охранной системе – СКУД (систему контроля и управления доступом). С помощью новых биометрических системам идентификации, можно значительно безопасность предприятия, сохранности данных, возможности НСД (несанкционированного доступа) и его сотрудников.

Раньше, для распознавания людей, на проходных предприятия устанавливали турникеты со считывателями карт, но сегодня подобную технологию можно назвать устаревшей, в связи с быстротечным развитием технологий, компании все чаще переходят на иные методы распознавания, более точные и удобные.

В настоящее время наблюдается усиливающийся интерес к проблеме распознавания лиц. Распознание лиц, можно поделить на две основные части:

- Идентификация;
- Верификация.

Основной задачей идентификации можно назвать сравнение захваченного лица со всеми изображениями лиц, которые хранятся в базе данных.

Целью верификации является сравнение лица человека с фотографией кандидата, с лицом на эталонной фотографии, которая хранится в базе данных.

Подобные алгоритмы актуальны как в области интеллектуальных сред, так и в системах безопасности.

Среди продукции с открытым исходным кодом стоит обратить внимание на OpenCV.

На основе которой будет рассматриваться и проектироваться практическая часть диплома. Данный сервис можно свободно использовать в академических и коммерческих целях- распространяется по модели BSD.

Алгоритм распознавания лиц по изображению человека имеет ряд преимуществ по сравнению с другими способами идентификации:

- Нет необходимости использовать дорогое оборудование;
- Не нужен непосредственный физический контакт с устройством, зачастую достаточно лишь ненадолго задержаться перед камерой или просто пройти мимо.

К недостаткам же подобных алгоритмов стоит отнести следующее:

- Подобная система сама по себе не может обеспечить 100% надежность идентификации. Там, где требуется высокая надежность, применяется несколько способов биометрических методов.
- Распознавание лица неэффективно тогда, когда значительные изменения, например, вследствие пластической операции, делают невозможным даже человеческую визуализацию.
- Система может совершать ошибки и по другим критериям, таким как: признаки старения, мимики, освещения и угла зрения. Однако этот фактор зависит от наполненности базы данных различными изображениями.

В настоящее время используются несколько десятков компьютерных методов для распознавания лиц:

- Нейронные сети;
- Метод главных компонентов (собственных лиц);
- Локальные бинарные шаблоны;
- Эластичные графы.

Данная совокупность факторов определяет актуальность настоящей ВКР, целью которой является выбрать, обосновать и реализовать условия обеспечения информационной безопасности алгоритма распознавания лиц в условиях стремительно развивающихся информационных технологий и

вынужденных мер по частичному переходу на дистанционный формат значительного количества сервисов.

Для достижения сформулированной цели в ВКР поставлена и решена задача разработки алгоритма распознавания лиц с использованием языка программирования Python с кодом OpenCV и также рассмотрены процессы обеспечения информационной безопасности

Распознавание лиц попадает под категорию искусственного интеллекта и использует компьютерную оптику, акустику, физические датчики, биологические и статистические принципы и передовые математические методы для создания моделей, которые превращают физиологические характеристики человека в что-то идентифицируемое.

Также стоит отметить, что область распознавания лиц в современном мире имеет практически безграничную область эксплуатации.

Распознавание лиц, можно широко использовать во многих сферах, таких как аэропорты, культурные достопримечательности, отели, железнодорожные вокзалы и другие места.

С быстрым развитием технологии искусственного интеллекта распознавание лиц стало широко применяться в социальной работе и жизни.

Какие-то технологии, например проверка лиц и открытие дверей по биометрическим параметрам, обеспечивают удобство, их собственные проблемы с безопасностью приводили к выявлению безопасности пользователей. Это означает тот факт, что стоит понять необходимость улучшения системы распознавания лиц.

Идентификация лиц является сложной задачей в области просмотра изображений и компьютерного зрения.

Информационная безопасность пользователей становится крайне значимой и сложной задачей для специалистов в современном мире.

В своей работе я рассматриваю распознавание лиц людей с использование библиотеки OpenCV. Она решает задачи компьютерного зрения. Компьютерное зрение работает по принципу моста между

компьютерным программным обеспечением и визуальной картиной вокруг нас. Оно предоставляет возможность для ПО понимать и изучать все видимые процессы в окружающей среде.

Сначала мы будем собирать данные, затем выполнять определенные действия по их обработке, а потом многократно обучаем модель, как ей распознать лицо по размеру, форме или цвету.

В настоящее время существуют различные пакеты для выполнения задач машинного обучения, глубокого обучения и компьютерного зрения, проработан лучше других.

Исследуя текущую тему распознавания лиц, и моя цель в этой ВКР — изучить модель распознавания лиц и найти нюансы её применения с точки зрения информационной безопасности.

1. НАУЧНО-ТЕХНИЧЕСКИЕ ОСНОВЫ ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ «АЛГОРИТМА РАСПОЗНАВАНИЯ ЛИЦ» НА БАЗЕ «OPENCV»

1.1. Теоретические сведения о «алгоритме распознавания лиц» и принципы его функционирования.

Для начала стоит дать определение такому понятию, как «распознавание лица». Это некое биометрическое программное приложения, которое способно однозначно и максимально точно идентифицировать или же верифицировать человека путем сравнения и анализа шаблонов на основе контуров лица человека.

Каждый человек имеет неповторимое строение лица. Программное обеспечение способно анализировать его и обращаться к базе данных с информацией для последующей идентификации вашей личности.

Принцип работы алгоритма можно условно разделить на 3 основных этапа и большинство методов распознавания лица, имеют примерно идентичную структуру. Метод рассматриваемый в ВКР, также относится к тем, которые опирается на них.

- Обнаружение лица;
- Анализ лица;
- Конвертация изображения в данные;
- Поиск совпадений.

Первым шагом в ходе работы алгоритма будет обнаружение лиц. Первым делом камера обнаружит лицо человека, будь он один или же в окружение других людей.

Лицо лучше всего обнаруживается в тот момент, когда человек смотрит прямо в камеру, однако современные достижения в области технологий позволяют также обнаруживать лицо и в тех ситуациях, когда человек не смотрит прямо в камеру.

Вторым шагом будет анализ лица.

Снимается фотография лица и начинается его анализ. Значительная часть алгоритмов используют двухмерное изображение, поскольку сопоставление двухмерных изображений может выполняться с сопоставлением с общедоступными фотографиями или фотографиями, которые находятся в базе данных.

Каждое лицо составлено из различимых ориентиров или же узловых точек. Каждое человеческое лицо имеет 80 узловых точек. Программы анализируют расстояния между этими точками.

Следующим этапом будет конвертация изображения в данные.

После анализа вашего лица программа интерпретирует его в виде математической формулы. Ваше лицо становится числовым отпечатком.

Далее выполняется поиск совпадений с базой данных изображений.

Числовой отпечаток сравнивается по основным идентификаторам.

1.1.1 Обзор современных методов распознавания

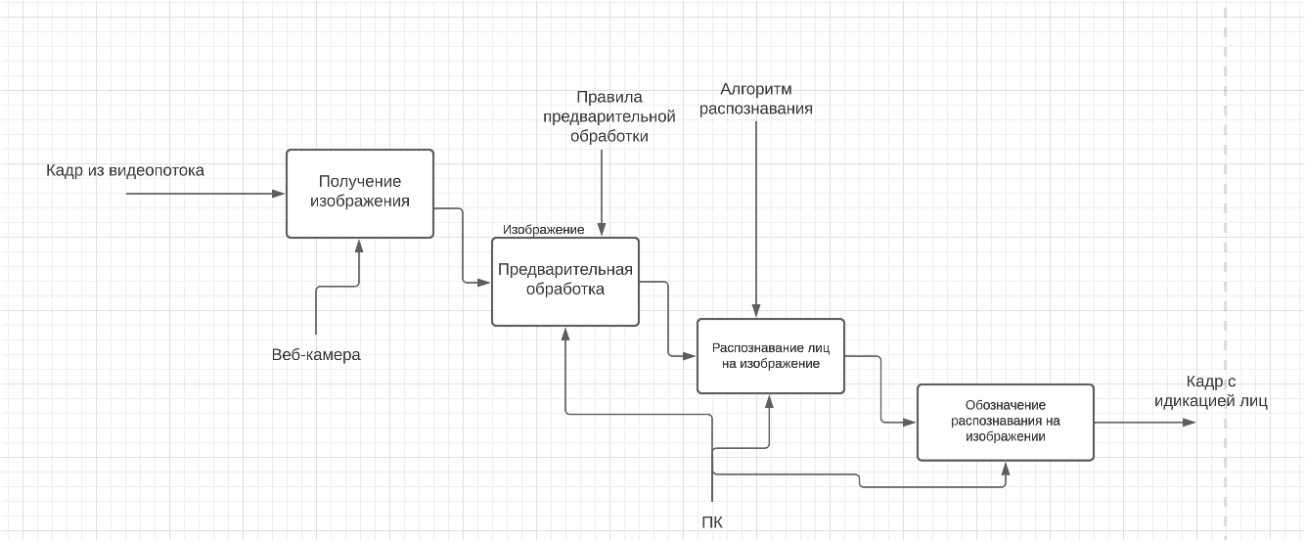

Рисунок 1- Диаграмма распознавания лиц с веб-камеры

Практически все алгоритмы обнаружения лиц, можно условно разделить на 4 типа:

- Эмпирический метод;
- Метод характерных инвариантных признаков;
- Распознавание с помощью шаблонов, заданных разработчиком;

- Метод обнаружения по внешним признакам , обучающиеся системы.

Эмпирический подход, который базируется на «знаниях сверху-вниз» подразумевает реализацию алгоритма, который будет соответствовать определенному набору данных. Для которых должен соответствовать участок изображения.

Составление таких правил является способ формализовать эмпирические знания о том, как будет выглядеть лицо человека на изображениях. По итогу по этим правилам определяется: имеется ли лицо на участке изображения или нет.

Пример самых базовых правил:

- На изображения виднеется значительная разница в яркости между центральной честью лица;
- Яркость и цвет по центру части лица являются однородными;
- Сильно отличаются по яркости относительно остальной части лица, два симметрично расположенных глаза, нос и рот

Для того, чтобы сгладить помехи, применятся метод сильного уменьшения изображения, который производит действие по уменьшению изображения. Это также значительно понижает количество вычислительных операций(рисунок 2). Метод позволяет более быстро и просто определить зону равномерного распределения яркости(зона, где предположительно находится лицо), чтобы в дальнейшем выполнить проверку на наличие значительно отличающихся по яркости областей внутри: именно такие области с разной вероятностью отнести к «лицу».

Рисунок 2 - МетодYang&Huang

Метод построения гистограмм использует вертикальные и горизонтальные гистограммы. В областях-кандидатах осуществляется поиск черт лица. При получении гистограммы определенной формы можно определить вероятность наличия лиц.

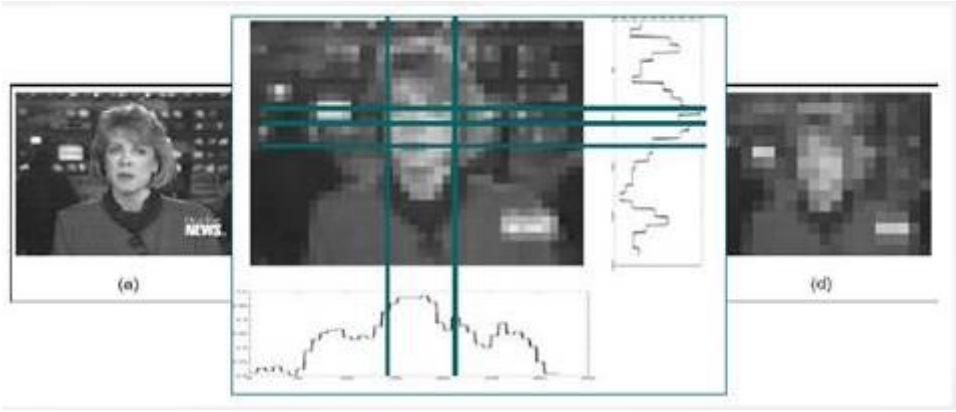

Рисунок 3 - Метод Kotropoulos & Pitas

Вышеописанный подход использовался в самом начале пути развития компьютерного зрения, так как для реализации этого метода требовались низкие требования к вычислительной мощности процессора для обработки изображения.

Методы, которые мы описали имеют не самые плохие показатели по определению лиц на изображениях с однородным фоном. Их просто реализовать с помощью машинного кода, что позволяло разрабатывать множество схожий алгоритмов.

Недостатком является их полная непригодность при наличии сложного заднего фона и чувствительность к наклону и повороту головы.

Методы характерных инвариантных признаков, которые основываются на знаниях снизу-вверх образуют вторую группу методов определения объектов. Тут виднеется подход к проблеме: не происходит формализации протекающих в человеческом мозге процессов в явном виде.

Приверженцы этого подхода пытаются найти инвариантные особенности, выявить неявные закономерности и особенности объектов, независимо от угла наклона и положения.

Основные этапы алгоритмов этой группы методов:

- Обнаружить: границы лица, форма, яркость, текстура, цвет;
- Детектирование на изображении явных признаков лица: глаза, нос, рот;
- Объединение всех найденных инвариантных признаков и их верификация.

В особенно сложных сценах предполагается поиск правильных геометрически расположения форм лица. Для того, чтобы это реализовать применяется гауссовский производный фильтр с большим количеством различных масштабов и ориентаций. Далее случайным перебором выполняется поиск соответствия выявленных черт лица и их взаимное расположение.

Сущность метода группировки признаков с применением второй производной гауссовского фильтра для поиска необходимый областей изображения(рисунок 4). После этого группируются края вокруг каждый такой области при помощь порогового фильтра.

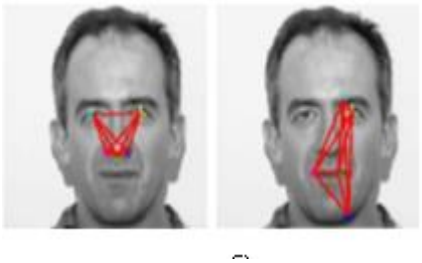

Рисунок 4 - Верные и случайные срабатывания

Для того, чтобы скомбинировать найденные признаки используется оценка при помощи байесовской сети. Выполняется выборка черт лица.

В этой группе методов имеется ряд недостатков. Значительное влияние оказывает сложный задний фон изображения, при котором могут возникать проблемы с обнаружением. При небольшом скрытии лица другими объектами, засветке или возникновении шумов процент достоверного распознавания также сильно падает. Основа рассмотренных подходов – эмпирика, одновременно сильная и слабая сторона этого подхода.

В этом случае обнаружение объектов на изображении будет относится к задачам высокой сложности из-за следующих факторов: значительная изменчивость объекта расположения, зависимость от освещения и условий съемки.

Однако использование эмпирических правил позволяет свести задачу распознавания объекта до фиксированного количества простых проверок. Но стоит отметить, что данные методы первой категории, пока что далеки по своей эффективности от уже давно отлично функционирующего инструмента ─ человеческого зрения, так исследователи, которые решили изучать этот вопрос, сталкиваются с рядом серьезных трудностей.

Первой причиной является факт того, что внутренние процессы протекающие в человеческом мозге далеко не полностью изучены, и набор эмпирических знаний, который на нынешний момент доступны на сознательном уровне, далеко не полностью исчерпывает весь спектр подсознательных инструментов.

Второй причиной, можно выделить невозможность перенести неформальный человеческий опыт в набор определенных правил, поскольку в ряде случаев это привести к огромному количеству ложных срабатываний, либо наоборот ─ обнаружение и вовсе не произойдет.

Распознавание с использованием шаблонов заданных разработчиком. Шаблоны создают некий стандартный образ изображения лица, к примеру путем описания свойств разных областей лица и их теоретического расположения. Обнаружения лица при помощи шаблона заключается в проверки каждой из областей изображения на соответствие данному шаблону. Стоит выделить следующие особенности этого подхода:

- Два вида шаблонов(недеформированные, деформированные);
- Шаблоны заранее запрограммированы и являются обучаемыми;
- Используются корреляция для нахождения лица на изображение;

Метод детектирования лица с использованием трехмерных форм предусматривает использование шаблонов в виде неких пар яркостей в двух

областях. Для детекта лица требуется просканировать все изображение и сравнить с заданным шаблоном. Причем совершать это необходимо с разным масштабом(рисунок 5).

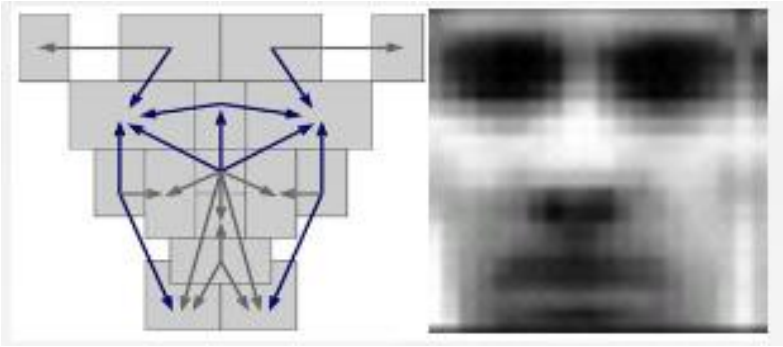

Рисунок 5 - Метод детектирования лица при помощи трехмерных форм

Модели распределения опорных точек являются статистическими моделями, которые представляют объекты, форма которых может измениться.

Их польза для метода ─ возможность выделить форму переменных объектов в рамках учебного набора с небольшим количеством параметров формы.

Это компактная и достоверная параметризация может использоваться для разработки эффективных систем классификации.

Из основных достоинств распознавания с помощью шаблонов можно отметить относительную простоту реализации и неплохие результаты срабатывания для изображений с несложным задним фоном.

Основной недостаток этого метода ─ необходимость калибровки шаблона вблизи с изображением лица.

Рентабельность метода не считается высокой из-за сложности вычисления шаблонов для различных поворотов лица и ракурсов.

Зачастую поиск лиц на изображениях при помощи методов, основанных на построении математической модели изображения лица, заключается в полном переборе всех прямоугольных фрагментов на наличие лица. Так как схема полного перебора включает в себя такие очевидные недостатки, как избыточность и значительная вычислительная сложность.

Применяются способы сокращения количества рассматриваемых фрагментов.

Основные принципы методов:

- Схоластика: каждый сканируется окном и представляется векторами ценности.
- Блочная структура: изображение разбивается на пересекающихся и непересекающиеся участки(рисунок 6) разных показателей масштаба и проводится оценка при помощи алгоритмов оценки весов векторов.

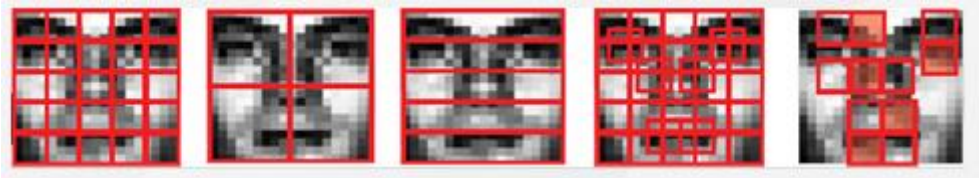

Рисунок 6 - Примеры разбиения изображения на участки

Чтобы обучить алгоритм требуется библиотека вручную или программно подготовленных изображений лиц и «не лиц», то есть изображений на которых нет лица или же присутствует его часть.

Важно отметить, что важной задачей является выделение сильных классификаторов. Конкретно они будут иметь наивысший приоритет при проверки обнаруженных признаков в изображении. Количество же слабых классификаторов стоит уменьшить за счет схожести друг с другом и удалении классификаторов, которые возникают за счет шумовых выбросов.

Важно выделить основные методики выполнения этих задача:

- Искусственные нейронные сети
- Метод главных компонентов
- Сравнение шаблонов
- Скрытые Марковские модели
- Метод гибкого сравнения на графах
- Совмещение ФА и метода главных компонентов
- Разреженная сеть окон

- Активные модели
- Адаптированное улучшение и основанный на нем Метод Виолы-Джонса

В методе Виолы-Джонса эксплуатирует алгоритм сканирования окна. Реализуется он следующим образом: На исходном изображении устанавливается окно сканирования в начальное положение, выбраны используемые признаки. Окно поступательно двигается по изображению с шагом в 1 ячейку.

Проводится поступательное сканирования для разных масштабов. Масштабируется само сканирующее окно. Все обнаруженные признаки попадают к классификатору, который «производит оценку».

Вычислять все признаки на слабом ПК просто невозможно, поэтому классификатор должен реагировать на строго определенный набор признаков. Совершенно логично, что необходимо обучить классификатор по данному набору. Это все будет производится автоматически.

После описания существующих систем, стоит выделить их основные недостатки.

Недостатки методов свойственных инвариантных признаков: При незначительном перекрытии лица другими предметами, возникновение шумов или засветке процент корректного распознавания значительно понижается. Задний фон оказывает значительное влияние.

Недостатки методов распознавания с помощью шаблонов: значительным недостатком является то, что важно откалибровать шаблон вблизи с изображением лица. Рентабельность использования также ставится под сомнение, поскольку необходимо вычислять шаблоны для разных ракурсов и поворотов лица.

Недостатки нейронных сетей: когда мы будем добавлять нового пользователя в БД требуется полное переобучение сети для всего имеющегося набора. Проблемы математического характера: попадание в локальный оптимум, выбор правильного шага для оптимизации,

переобучение и так далее. Трудно формализуемый этап выбора архитектуры сети. Исходя из изложенного, можно заключить, что нейронная сеть ─ «черный ящик» с трудно интерпретируемыми результатами работы.

Недостатки методов главных компонентов: метод обязывает для своего применения идеальных условий таких как например: строго единые параметры освещенности, нейтральное выражение лица, отсутствие помех на лице по типу очков или бороды.

Недостатки метода сравнения шаблонов: недостатком можно назвать то, что требуется много ресурсов как для сравнения участков изображений, так и для хранения. Эксплуатируется простой алгоритм сравнений. Это накладывает ограничение на оригинальное изображение ─ должны быть сняты в строго поставленных условиях. Не допускаются заметные изменения ракурса, освещения, эмоциональных выражений.

Недостатки скрытых Марковских моделей:

- Необходимость подбора параметров модели для каждой отдельной базы данных;
- Не имеют различающую способность, 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

- Значительная вычислительная сложность процедуры распознавания;
- Низкая технологичность при запоминании новых эталонов;
- Линейная зависимость времени работы от размера базы данных лиц

1.1.2 Обнаружение лица OpenCV

Сперва для того, чтобы программа смогла определить лицо и не спутать его с какими-либо другими объектами. Нам необходимо выделить его основные компоненты, такие как нос, лоб, глаза, губы.

Подобный подход удобнее всего показать на схематичном шаблоне, которые будет теоретически обоснован в дальнейшем.

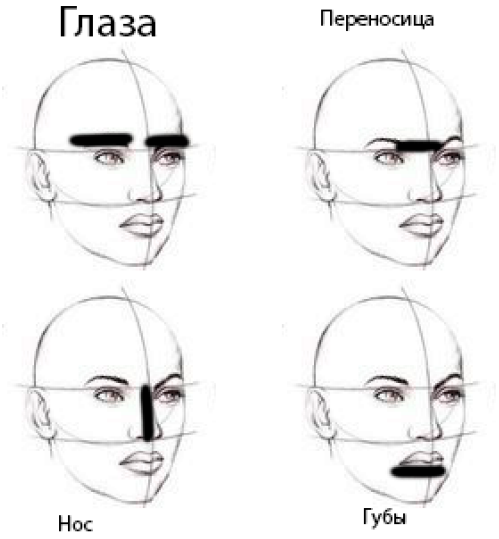

Рисунок 7 – Основные шаблоны

Для обнаружения лица метод OpenCV использует признаки Хаара.

По сути это признаки цифрового изображения, используемые в распознавании образов. Эти самые признаки использовали в самом первом детекторе лиц, который работал в режиме реального времени.

Признак Хаара представляет из себя множество смежный прямоугольных областей.

Они позиционируются на изображении, а далее суммируются с интенсивностью пикселей в областях, после чего вычисляется разность между суммами.

Эта разница и представляет из себя значение определенного признака, определенного размера, определенным образом с позиционированным на изображении.

Поскольку в ВКР мы рассматриваем распознавание лиц, то и признаки Хаара будут опираться на базу данным с человеческими лицами.

Общим для всех изображений является то, что область в районе глаз будет темнее, нежели чем в районе щек. Из этого следует, что общим признаком Хаара для лиц будут являться два смежным прямоугольника региона, которые лежат на глазах и щеках, соответственно.

Главной особенность признаков Хаара является высокая скорость обработки.

1.1.3 Прямоугольные признаки Хаара

Простейший прямоугольный признак Хаара можно рассматривать как разность сумм пикселей двух смежных областей внутри прямоугольника, который может занимать разные положения и масштабы на каком-либо изображении.

Такой вид признаков называется 2-прямоугольным. Каждый признак может показать наличие или отсутствие какой-либо определенной характеристики изображения, такой как границы или изменения текстур.

Например, 2-х прямоугольный признак может дать информацию о том, где находится граница межу темным и светлым регионами.

Данный признак напрямую связан с распознаванием лиц, так как на него будет опираться база данных при определении лица по причине, описанной выше.

1.1.4 Наклонные признаки Хаара

Наклонные признаки оказались удачным нововведением, так как некоторые наклонные признаки были способны лучше описывать объект.

Например, 2-прямоугольных наклонных признака Хаара могут показать нам наличие края, который был наклонен на 45 градусов.

Для ускорения вычислений, детектор использует изображения низкого разрешения, что приводит к ошибке округления. Исходя из этого, наклонные признаки Хаара обычно не используются

1.1.5 Метод Виолы-Джонса

Обычно у каждого метода есть основа, то, без чего этот метод не мог бы существовать в принципе, а уже над этой основой строится вся остальная часть.

В методе Виолы-Джонса эту основу составляют примитивы Хаара, представляющие собой разбивку заданной прямоугольной области на наборы разнотипных прямоугольных подобластей, изображенных на рисунке 2:

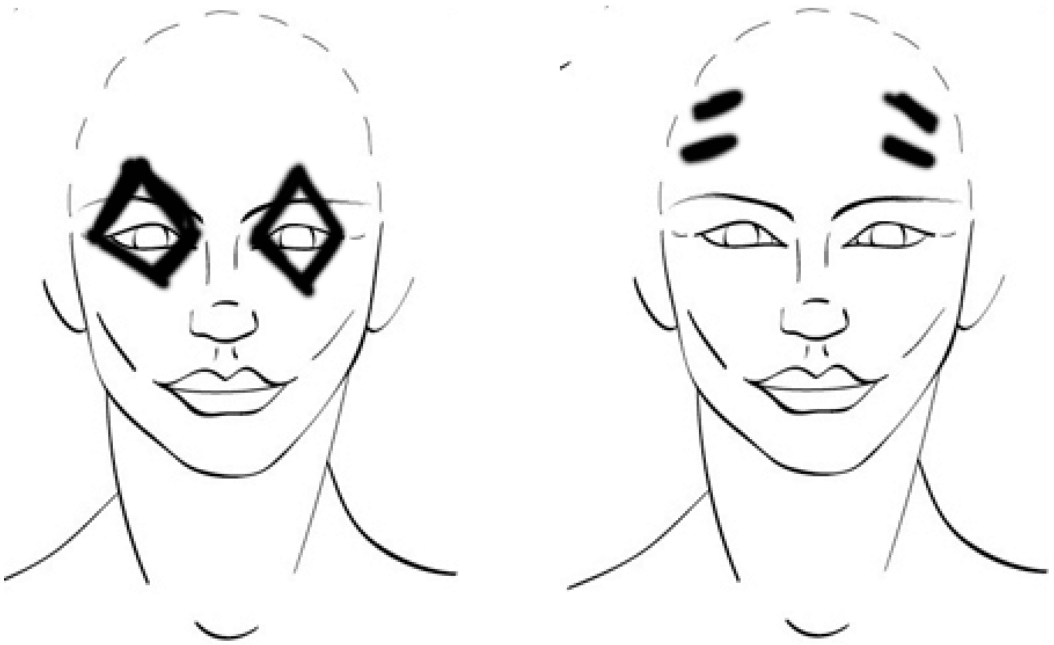

Рисунок 8 – Шаблоны с поворотом

В оригинальной версии алгоритма Виолы-Джонса использовались только примитивы без поворотов, а для вычисления значения признака сумма яркостей пикселей одной подобласти вычиталась из яркостей другой подобласти. В развитии метода были предложены примитивы с наклоном на 45 градусов и несимметричные конфигурации.

Вместо вычисления разности, было предложено приписывать каждой подобласти определенный вес и значения признака вычислять как взвешенную сумму пикселей разнотипных областей. Используя формулы под номером 1:

$$feature = \sum_{i \in I=1,\ldots,N} w_i \cdot RectSum(r_i) \qquad (1)$$

Сложность вычисления признака так же как и получения значения пикселя остается O(1): значение каждой подобласти можно вычислить скомбинировав 4 значения интегрального представления (Summed Area Table — SAT), которое в свою очередь можно построить заранее один раз для всего изображения за O(n), здесь n — число пикселей в изображении, используя формулы 2:

$$SAT(x,y) = SAT(x,y-1) + SAT(x-1,y) + I(x,y) - SAT(x-1,y-1)$$
$$SAT(-1,y) = SAT(x,-1) = SAT(-1,-1) = 0 \quad (2)$$

Это позволило создать быстрый алгоритм поиска объектов, который пользуется успехом уже больше десятилетия. Но вернемся к нашим признакам.

Для определения принадлежности к классу в каждом каскаде, находиться сумма значений слабых классификаторов этого каскада. Каждый слабый классификатор выдает два значения в зависимости от того больше или меньше заданного порога значение признака, принадлежащего этому классификатору.

В конце сумма значений слабых классификаторов срав-нивается с порогом каскада и выносится решения найден объект или нет данным каскадом

1.1.6 Метод наклонных векторов

Метод опорных векторов — это контролируемый метод обучения, который можно широко использовать в статистической классификации и регрессионном анализе.

Машины опорных векторов представляют собой обобщенные линейные классификаторы.

Характеристика этого семейства классификаторов заключается в том, что они могут минимизировать эмпирическую ошибку и одновременно максимизировать геометрическую площадь края, поэтому метод опорных векторов также называют классификатором максимальной площади края.

1.1.7 Распознавание и детектирование лица. Алгоритм AdaBoost

Алгоритм AdaBoost можно использовать для повышения производительности любого алгоритма машинного обучения.

Машинное обучение стало мощным инструментом, позволяющим делать прогнозы на основе больших объемов данных.

В машинном обучении бустинг нужен для преобразования слабых классификаторов в сильные. Слабый обучающий алгоритм или классификатор – это обучающий алгоритм, который работает лучше, нежели чем случайное угадывание, и работать хорошо он будет в случае переобучения, поскольку при большом наборе слабых классификаторов, любой слабый классификатор будет работать лучше, чем случайная выборка.

Смысл использования данного алгоритма при детектирование лица с помощью OpenCV заключается в том, что у нас есть некоторый набор эталонных объектов.

То есть, есть значение и класс к которому они принадлежат. К примеру, -1 – нет лица, +1 – есть лицо, кроме того имеется множество простых классификаторов, то мы можем составить один более совершенный и мощный классификатор.

В процессе составления и обучение финального классификатора акцент делается на эталоны, которые распознают хуже, в этом и заключается адаптивность алгоритма, в процессе обучения он подстраивается под наиболее сложные объекты.

В заключение данного раздела основных составляющих элементов распознавания лица при помощи OpenCV, можно структуру модели обнаружения человеческого лица.

Рисунок 9 – Структура модели обнаружения человеческого лица

Извлечение прямоугольных признаков Хаара и строгий классификатор на основе AdaBoost являются важной частью распознавания лиц. Функция Хаара состоит из нескольких одинаковых прямоугольников, которые отличаются черно-белой разницей цветов, а значения функций Хаара определяются значениями пикселей прямоугольника.

Метод Adaboost, непосредственно определяющий частоту ошибок простого классификатора, позволяет избежать трудоемких процессов, таких как итеративное обучение и статистическое распределение вероятностей.

1.1.8 Сверточные нейронные сети

Сверточные нейронные(CHH) сети являются наиболее часто используемым типом метода глубокого обучения для распознавания лиц.

Основное преимущество метода глубокого обучения заключается в том, что вы можете использовать большой объем данных для обучения, чтобы получить надежное представление об изменениях, происходящих в обучающих данных.

Этот метод не требует разработки конкретных характеристик, устойчивых к различным типам классовых различий (таких как освещение, поза, выражение лица, возраст и т. д.), но может быть извлечен из обучающих данных.

Основным недостатком методов глубокого обучения является то, что для обучения им необходимо использовать очень большие наборы данных, и

эти наборы данных должны содержать достаточно изменений, чтобы иметь возможность обобщать шаблоны, которые никогда раньше не наблюдались.

Некоторые крупномасштабные наборы данных о лицах, содержащие изображения естественных лиц, были обнародованы и могут использоваться для обучения моделей СНН.

В дополнение к обучению функциям распознавания нейронные сети также могут уменьшать размерность и могут обучаться с помощью классификаторов или использовать метрические методы обучения.

СНН считается сквозной системой обучения, и ее не нужно сочетать с какими-либо другими конкретными методами.

Компьютерная система распознавания образов в основном состоит из трех взаимосвязанных, но четко дифференцированных процессов; а именно, генерация данных, анализ паттернов и классификация паттернов. Генерация данных заключается в преобразовании исходной информации входного шаблона в вектор и придании ей формы, удобной для обработки компьютером.

Анализ шаблонов — это обработка данных, включая выбор признаков, извлечение признаков, сжатие данных по размеру и определение возможных категорий.

Классификация изображений — это использование информации, полученной в результате анализа изображений, для обучения компьютера формулированию критериев распознавания для классификации изображений распознавания.

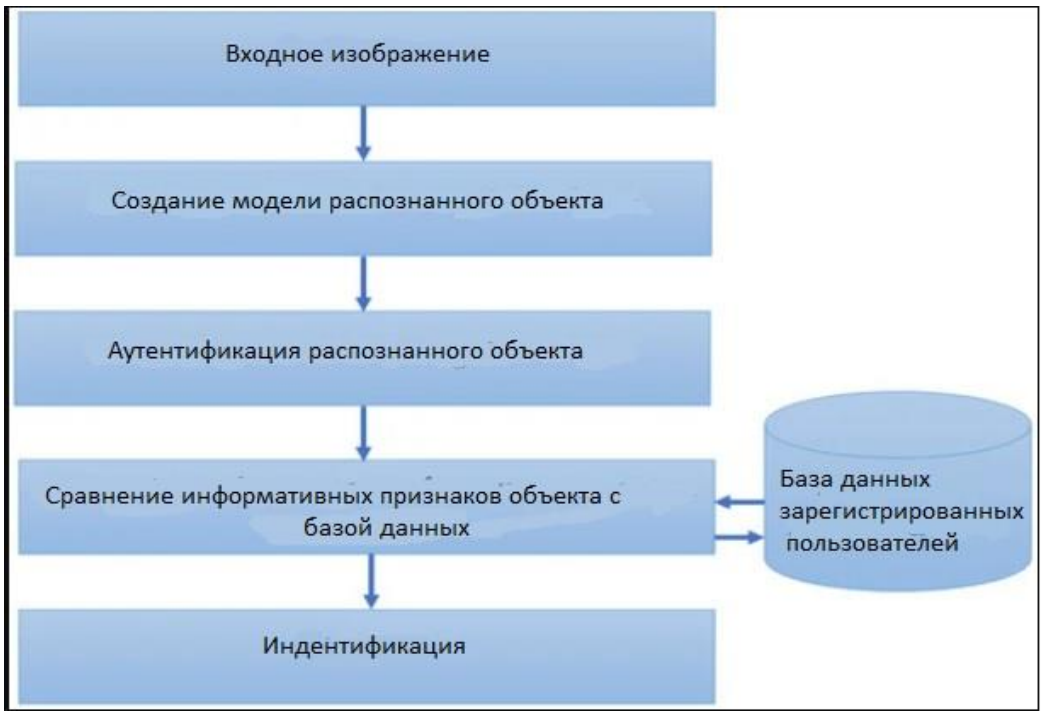

Рисунок 10 – Структура системы распознавания образов лиц

- Блок построения модели распознавания объектов (поиск координат поверхности человека на рисунке, определение зонального расположения информации, предварительная обработка и нормализация;
- Блок аутентификации распознавания объектов (алгоритмы аутентификации объекта, подлежащего идентификации путем управления доступом к системе фотораспознавания пользователя, зарегистрированного в базе данных);
- блок расчета информационных идентификационных меток (сверточные нейронные сети, индикаторы корреляции, расстояние Минковского и др.).

1.2 Анализ угроз для функционирования «Алгоритма распознавания лиц»

При всей скорости и практичности алгоритмов распознавания лица, в ходе их эксплуатации может возникнуть ряд угроз для безопасности информации.

Оценка угроз безопасности информации проводится в целях определения угроз для безопасности информации, возникновение которых

возможно в системах и сетях при эксплуатации алгоритмов распознавания лица.

1.2.2 Ошибка идентификации

Источник угрозы — это потенциальные антропогенные, техногенные или стихийные носители угрозы безопасности.

Данная угроза относится к техногенным угрозам информационной безопасности. Так как напрямую зависит от качества написанного алгоритма и вероятности точного определения лица.

Угроза(действие) — это возможная опасность совершения какого-либо деяния, направленного против объекта защиты, наносящего ущерб собственнику, владельцу или пользователю, проявляющегося в опасности искажения и потери информации.

При ошибке идентификации нарушается один из основных принципов защиты информации, а именно её доступность поскольку пользовать не может своевременно получить информацию.

Доступность – свойство информации, гарантирующее, что лица имеющие доступ к информации в нужный момент смогут получить доступ.

К примеру, сотрудник информационной безопасности не смог своевременно быть идентифицирован своим рабочим компьютером и в итоге не смог получить информацию об атаке на сервер, что повлекло за собой последствия. И таких примеров, можно привести ещё множество.

Фактор — это присущие объекту информатизации причины, приводящие к нарушению безопасности информации на конкретном объекте.

Обусловленные недостатками процесса функционирования объекта информатизации, свойствами архитектуры автоматизированной системы, протоколами обмена и интерфейсами, применяемыми программным обеспечением и аппаратной платформой, условиями эксплуатации.

Как правило подобные ошибки возникают в результате плохого качества изображений или недостатка информации в базе данных. Плохое освещение или низкое качество изображения могут затруднить выполнение

точного анализа узловых точек человека. Например, на данные может повлиять затемнения ряда черт лица. Это создает ошибку в отпечатке лица, в результате чего будет невозможно сопоставить его с правильными данными в базе данных.

Последствия — это возможные последствия реализации угрозы при взаимодействии источника угрозы через имеющиеся факторы.

Последствия в результате возникновения данной угрозы могут совершено разнообразными. От экономических из-за несвоевременного доступа к необходимой пользователю информации, что может повлечь за собой материальный ущерб.

1.2.3 Конфиденциальность

В данном случае может быть нарушен ещё один из «постулатов» информационной безопасности.

Конфиденциальность — свойство информации, гарантирующее, что доступ к информации имеет доступ только определенные лица.

Многие обеспокоены проблемами конфиденциальности, которые связаны с алгоритмами распознавания лица. Данная технология не застрахована от того, что может произойти крупное нарушение данных, а потому вся личная информация, которую собирают программы для распознавания лиц, вовсе не застрахована.

Источниками угроз в этом случае может выступать злоумышленники, которые смогли получить несанкционированных доступ к приватной информации пользователей.

Фактором могут выступать недостаточные меры информационной защиты алгоритма или какие-либо дыры в информационной системе, которая эксплуатирует данные алгоритмы.

А так как на системе распознавания лиц завязано огромное количество сервисов, то допущение подобных сбоев необходимо сводить к минимуму

Последствия в ходе нарушения конфиденциальности информации могут быть значительными, т. к. они несут за собой материальный и моральный ущерб для пользователей.

1.2.4 Злоупотребление данными

Данная уязвимость опять-таки затрагивает конфиденциальность. Большинство людей не считает приемлемым использование данной технологии в некоторых сферах человеческой жизни.

В таких направлениях, как например, наблюдения в многоквартирных домах, отслеживания присутствия сотрудников на рабочем месте, контроль реакции людей на рекламные объявления. Такое недоверие обусловлено тем, что люди опасаются того, что частные компании могут злоупотреблять их персональными данными.

Некоторые частные лица имеют доступ к таким данным, но и многие базы данных для распознавания лиц являются публичными. А это означает, что любой человек, даже со злым умыслом, может найти вас в базе данных и выследить.

Данная угроза безопасности является техногенной и антропогенной одновременно. Поскольку люди сами предоставляют свои конфиденциальные данные, когда подписывают пользовательское соглашении.

1.3 Стандарты и оценка информационной безопасности

Биометрическая система аутентификации, как и иные системы, может быть подвергнута нападению злоумышленников. Начиная с 2011 года, международная стандартизация в области информационных технологий начала предусматривать и проводить мероприятия по защите биометрических данных — IS0/IEC 24745:2011.

Этот стандарт предусматривает анализ угроз и контрмер, проводимых для них, требования безопасности, модели применения и также сценарии хранения биометрических эталонов. В нем предусматривает ещё руководство

по защите частной жизни физического лица при обработке биометрической информации.

В российском законодательстве защите биометрических данных регламентирует следующий закон: «О персональных данных» № 152-ФЗ от 27 июля 2006 г., с последними изменениями в 2022 году. Этот нормативно-правовой акт регламентирует, какие данные относятся к персональной информации о гражданах, требования, сценарии и стандарты хранения.

2. АНАЛИЗ МАТЕМАТИЧЕСКОГО ОБОСНОВАНИЯ АЛГОРИТМА И ПРОГРАММНЫХ МЕТОДОВ РЕАЛИЗАЦИИ ПРОЕКТА. РАЗРАБОТКА МОДЕЛИ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТ

2.1. Математическое обоснование алгоритма распознавания лиц

Ниже представлена математическая модель распознавания лиц. Определим набор изображений лиц в базе данных: $X_1, X_2, X_3 \dots X_L$.

Наборы разбиты на L классов, где каждый класс соответствует зарегистрированному человеку. Для каждого изображения мы будем определять вектор значений K:

$$\mu = (\mu_1, \mu_2, \mu_3 \dots \mu_k)^{\tau}, (1)$$

где τ является оператором транспонирования, то есть поворот данных.

Для каждого изображения мы определяем функцию расстояния. Функция расстояния $d(\mu_l, \mu_s)$ для вектора признаков на наибольшем расстоянии μ в форме ввода принадлежит классу X_L:

$$d(\mu_l, \mu_s) > d(\mu_j, \mu_s), l \neq j, l, j = 0,1, \dots, L-1 \ (2)$$

Для функции расстояния класс X_L должен превышать предварительно вычисленное пороговое значение $d(\mu_1, \mu_2) > \tau_c$.

Входными данными для алгоритма распознавания лиц является изображение, а выходными данными — последовательность координат кадра лица (0 кадров лица или 1 кадр лица или несколько кадров лиц).

Как правило, выходной кадр координат лица представляет собой обращенный вверх квадрат, но существуют также некоторые технологии обнаружения лиц, которые выводят обращенный вверх прямоугольник или прямоугольник с направлением вращения.

Обычный алгоритм обнаружения лиц представляет собой процесс «сканирования» и «различения», то есть алгоритм сканирует диапазон изображений, а затем, в свою очередь, определяет, является ли область-кандидат лицом.

Следовательно, скорость вычислений алгоритма обнаружения лиц связана с размером изображения и содержимым изображения. Входными данными для алгоритма регистрации лица является «изображение лица» плюс «кадр координат лица», а на выходе — последовательность координат ключевых точек черт лица. Количество ключевых точек черт лица представляет собой предварительно определенное фиксированное значение, которое может быть определено в соответствии с различной семантикой (обычно 5 точек, 68 точек, 90 точек и т. д.).

Используя математическую модель для определения интегрального изображения, можно найти координаты лица (алгоритм Виолы и Джонса):

$$I_i(x,y) = \sum_{i=1}^{x*y} i(x'y'), \ (3)$$

где $I_i(x,y)$ значения i-го элемента интегрального изображения с координатами равно $(x,y), (x'y')$ яркость пикселя рассматриваемого изображения с координатами $(x'y')$.

Интегральное изображение рассчитывается независимо от размера или расположения изображения и используется для быстрого расчета яркости заданных частей изображения.

s_i — сумма яркостей пикселей, лежащих в белых областях, вычитается из суммы яркостей пикселей, лежащих в черных областях:

$$S_i = \sum_{i=0}^{x*y} r_i, k - \sum_{i=0}^{y*x} r_i f, \quad (4)$$

где $\sum_{i=0}^{x*y} r_i, k, \sum_{i=0}^{y*x} r_i f$ представляют из себя сумму яркостей писклей.

Метод Adaboost, непосредственно определяющий частоту ошибок простого классификатора, позволяет избежать трудоемких процессов, таких как итеративное обучение и статистическое распределение вероятностей.

Чем ближе сумма яркостей пикселей к 0, тем ниже яркость; чем ближе значение к 255, тем выше яркость.

Имеется n обучающих выборок $(x_1, y_2), (x_2, y_2), \dots, (x_n, y_n)$, где $x_n \in X$, $y_n \in Y$ будет соответствовать отрицательным и положительным пробам. В обучающем набор с есть m образцов отрицательных случаев и I образцов положительных случаев. При этом имеется набор определенных значений $w_i, 1 \leq i \leq n$, которые соответствуют каждому из образцов.

Выражением для этого окончательного классификатора с пороговым значением τ_c будет являться следующее:

$$W = \begin{cases} 1, & f_i \geq \tau_c \\ -1, & f_i < \tau_c \end{cases} \quad (5)$$

Лучший сильный классификатор вычисляется из точно определенного числа слабых классификаторов.

Определяться он будет следующим выражением:

$$W = \begin{cases} 1, \sum_{c=1}^{c} a_c w_c \geq \frac{1}{2} \sum_{c=1}^{c} a_c \\ 0, \sum_{c=1}^{c} a_c \, w_c < \frac{1}{2} \sum_{c=1}^{c} a_c \end{cases} \quad (6)$$

$$a_c = \log \frac{1}{\beta_c} \quad (7)$$

где w_c является слабы классификатором; a_c, β_c — являются коэффициентами слабого классификатора; $C = (1, \dots, C)$ — количество слабы классификаторов. Итерационный алгоритм, реализующий сильный классификатор (6) и (7), позволяет получить классификаторы на основе слабых (5). Окончательный классификатор необходим для обучения для минимизации ошибок. Изображение до и после конвертации и освещения изображения, можно описать следующим образом:

$$q_i(x, y) = \frac{g_{0,j}(x, y)}{g_{1,j}(x, y) + 1}, \quad (8)$$

где j номер текущего значения следующей последовательности:$Y, g_{(b,j)}(x, y)$ — значение яркости пикселя массива $I_{(b,j)}$, который соответствует изображению лица пользователя; b — это идентификатор массива пикселей, который указывает на режим подсветки, в котором был подготовлен массив;

b = (0,1), x и y — координаты рассматриваемого пикселя; W и H — количество пикселей, соответствующих массиву ширины и высоты $I_{(b,j)}$;

$x = 0, 1, \dots, W-1, y = 0, 1, \dots, H-1.$.

Следующим шагом, нам необходимо определить дисперсию всех величин (8). Для описания дисперсии, как значений числовой характеристики выборки (9) вводится суммарная характеристика выборочной дисперсии и математическое ожидание (10) среднего значения $q_i(x_1, y)$. Для оценки степени дисперсии относительно среднего значения (математического ожидания) рассчитывается дисперсия:

$$D\left(I_{0,j}, I_{1,j}\right) = \frac{1}{WH} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} (q_j(x,y) - q_j)^2 , (9)$$

где (q_i) — среднее значение или математическое ожидание дискретной случайной величины $q_j(x,y)$, которое рассчитывается по формуле:

$$q_j = \frac{1}{WH} \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} \left(q_j(x,y)\right) \ (10)$$

Описанные выше данные, можно найти во множестве публикаций, которые посвящены идентификации пользователей по изображениям лиц и моделированию информационных систем.

2.2. Программные методы реализации проекта

Поскольку данный проект был реализован на языке программирования Python, то стоит внести теоретические основы для понимания того, что это за язык программирования и то как он развивался.

За последние несколько лет Python стал одним из самых популярных и востребованных языков программирования в мире. Сферы применения данного языка программирования, совершенно разнообразны, можно сказать, что он используется во всем. От машинного обучения до создания веб-сайтов и тестирования программного обеспечения.

Его используют как разработчики, так и рядовые пользователи, которые заинтересованы в компьютерных технологиях.

Рисунок 11 – Логотип Python

Если в очередной раз касаться гибкости данного языка программирования, то стоит отметить, что на нем реализовано большинство алгоритмов рекомендаций, которые стали неотъемлемой частью жизни современного человека. Также из общеизвестных фактов, то стоит отметить, что он также эксплуатируется в создании программного обеспечения, управляющего беспилотными автомобилями.

Python — это в первую очередь язык общего назначения, что будет означать, что он предназначен для использования в совершенно разных приложениях.

Из наиболее известных сфер, которые находятся у всех на слуху, стоит отметить следующие: «Data science», «Web-разработка», «Автоматизация».

Какие основные задачи решает Python:

- Анализ данных и машинное обучение;
- Web-разработка;
- Тестирование программного обеспечения и создание прототипов.

Этот язык программирования является одним из основных в сфере «Data Science», что позволяет использовать язык для проведения и аналитики сложных статистических расчетов, создания визуализации данных,

построения алгоритмов машинного обучения, обработки и анализа данных и выполнения совершенно разных задач, которые связанны с данными.

В арсенале Python есть множество механизмов для реализации визуализаций данных, например, таких как линейные и гистограммы, круговые диаграммы, трехмерные графики.

В Python представлен ряд библиотек, которые дают возможность для программистов быстрее и более качественно писать программы для анализа данных и машинного обучения. Из самых известны примеров, стоит отметить: TensorFlow и Keros.

Что касается web-разработки, то можно выделить, что Python используется для разработки «back end» частей веб-сайтов или же приложения, если выражаться проще, то это те части, которые находятся вне поля зрения пользователя.

В web-разработке Python реализует возможность отправки данных на сервера и с серверов, обработку данных и взаимодействие с базами данных, маршрутизацию URL-адресов и обеспечение безопасности.

Если мы рассматриваем python в сфере автоматизации и скриптов, то данный язык программирования может позволить оптимизировать ход работы и сделать выполнение комплексной задачи более эффективной.

Написания кода подобного рода, называется сценарием. В кодинге автоматизация может использоваться для проверки ошибок в сразу нескольких файлах, преобразования этих файлов, а также выполнение простых математических операций и удаления дубликатов в данных.

В разработке программного обеспечения Python может помочь в следующих задачах: контроль сборки, отслеживание ошибок, тестирование.

Можно реализовать автоматизированное тестирование новых продуктов или же функций.

Возможности и преимущества Python:

- Этот язык программирования совместим с различными платформами, включая Windows, Mac, Linux, Raspberry Pi и др.

- Использование простого синтаксиса, который можно сравнить с английским языком, что позволяет использовать меньшее количество строк кода, чем некоторые другие популярные языки программирования.
- Позволяет работать в системе интерпретатора, что позволяет выполнять код немедленно, быстро отслеживая прототипирование.
- Имеет возможность обрабатываться процедурным, объектно-ориентированным, функциональным способом.

Синтаксис Python:

- Синтаксис Python похож на английский, но с значительным математическим влиянием.
- Использует точку с запятой и / или круглые скобки, как способ завершения команды, использует новые строки для той же функции.
- Определяет область действия (циклы, функции, классы), полагаясь на отступы, используя проблемы, а не фигурные скобки.

Гибкие стороны Python: Поскольку это язык с динамической типизацией, устраняются сложные правила для создания функций и предлагаются различные методы для решения возникших проблем. Также позволяет компилировать и запускать программы до проблемной области, так как он использует проверку типов во время выполнения, а не во время непосредственной компиляции.

Искусственный интеллект в Python: Python имеет под собой гибкий функционал для машинного обучения и тренировки нейронной сети.

Значительное количество библиотек предоставит вам возможность без особых проблем обучить нейронную сеть для совершенно разных задач.

После выбора языка программирования нам необходимо выбрать среду разработки, что также является важным критерием для комфортной работы и разработки алгоритма.

В ходе написания ВКР и разбора доступных средств, мой выбор пал на интегрированную среду разработки «PyCharm».

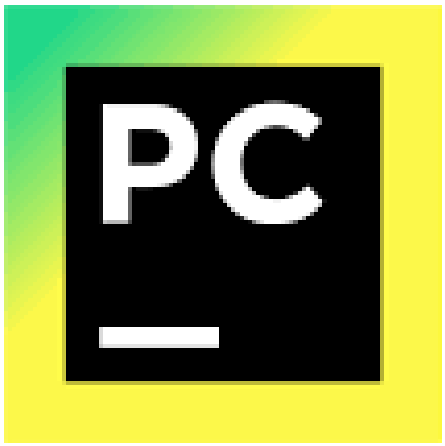

Рисунок 12 – логотип «PyCharm»

PyCharm — это Python IDE (интегрированная среда разработки) с полным наборов функций и инструментов для разработки Python. Помимо того, что среда IDE предоставляет возможности для профессиональной веб-разработки с использованием среды Django.

Данная среда разработки позволяет ускорить написания кода, ввиду наличия интеллектуального и настраиваемого редактора с автозавершением кода, фрагментов кода и поддержкой разделения кода для удобства выполнения множества задач одновременно.

Для того, чтобы лучше понимать принципы работы и актуальность данного ПО для разработки, стоит более подробно рассмотреть его ключевые особенности

Особенности PyCharm:

- **Интеллектуальная помощь при написании кода.** ПО обеспечивает интеллектуальное завершение кода, проверку кода и своевременное нахождение ошибок, и предложение их быстрого исправления, а также автоматический рефакторинг кода и широкие возможности навигации.
- **Интеллектуальная навигация по коду.** Среда разработки предлагает удобные инструменты для быстро поиска и перехода к любому классу, файлу, символу или к любому действию.
- **Быстрый и безопасный рефакторинг**. В среде разработки представлены способы для безопасного переименования и удаления,

извлечения метода, введения переменной, встроенной переменной или метода и другие способы рефакторинга.

- **Встроенные инструменты разработчика.** В репертуаре доступных инструментов, данная среда разработки включает в себя встроенный отладчик и средство запуска тестов, профилировщик, встроенный терминал, интеграцию с основными VCS(система контроля версий) и встроенными инструментами для работы с базами данных.

Стоит также отметить, что данная среда разработки создавалась конкретно под язык программирования Python с учетом всех его особенностей.

Основным преимуществом будет факт того, что большинство необходимых инструментов описанных выше, идут непосредственно в предустановленном режиме без необходимости добавления и загрузки, необходимых для комфортной работы средств.

После выбора среды разработки. Мы непосредственно приступаем к написанию кода алгоритма, однако из-за специфики ВКР. Для оптимизации и упрощения работы по обучения классификаторов.

В ходе написания ВКР было выявлена одна из основных подзадач. А именно обучения своего собственного HAAR-каскада.

В интернете существует множество вариаций предварительно обученных классификаторов и стоит отметить, что их объемность и точность для задачи по обнаружению лиц в присутствии разных условий освещения и разных поз лица поражает, однако это не отражает полного спектра работы для самостоятельного обучения алгоритм.

Есть значительный минус в использование таких классификаторов для любой задачи обнаружения. Она заключается в том, что мы никогда не будет знать, как будет проходить обучение таких классификаторов.

Также при обучение своего собственного классификаторов, мы можем подготовить набор данных для конкретной задачи и задать критерии, разные параметры классификаторов при его подготовке.

В своей ВКР я буду использовать узконаправленное ПО, которое позволяет создать и настроить свой собственный классификатор HAAR-каскада, а именно **«GUI Cascade Trainer».**

Рисунок 13 – логотип «GUI Cascade Trainer»

Это инструмент, который был разработан специально для решения задач в области обнаружения лица или же обнаружения объектов. В ходе эксплуатации данного ПО, мы создаем набор данных, который включает в себя положительные и отрицательные образцы для использования во время обучения.

Данная программа имеет в себе все необходимые средства для обучения классификатора и также для анализа ошибки. В ходе её эксплуатации, мы можем наблюдать за каждым этапом обучения, что в конечном итоге, может помочь нам повысить точность классификатора в детекторной работе.

Эта программа использует графический интерфейс для установки параметров и упрощает использование инструментов OpenCV для обучения и тестирования классификаторов.

2.3 Разработка модели обеспечения безопасности

Как уже говорилось ранее, системы с алгоритмами распознавания ─ это совершенно новый шаг в развитии методов аутентификации.

В нынешнее время такие алгоритмы используются повсеместно. В операционных системах уже давно активно применяется технология распознавания лиц, так как это дает возможность получать доступ к системе в три раза быстрее, чем при входе с паролем, большинство современных смартфонов поддерживают функцию разблокировки при помощи фронтальной камеры.

При всех своих явных плюсах системы распознавания с некоторой вероятностью могут выдавать ложный результат.

Одной из самых значительных характеристик любой биометрической системы представляются ошибки первого и второго рода.

FAR (False Acceptance Rate) ─ вероятность, что некий неавторизированный пользователь, которого нет в базе данных системе будет распознан как авторизированный пользователь (ошибка 1-го рода).

FRR (False Rejection Rate) ─ вероятность, что авторизированный пользователь будет распознан, как неавторизованный(ошибка 2-го рода).

EER (Equal Error Rate) ─ точка равновесия, в которой будут пересекаться FAR и FRR.

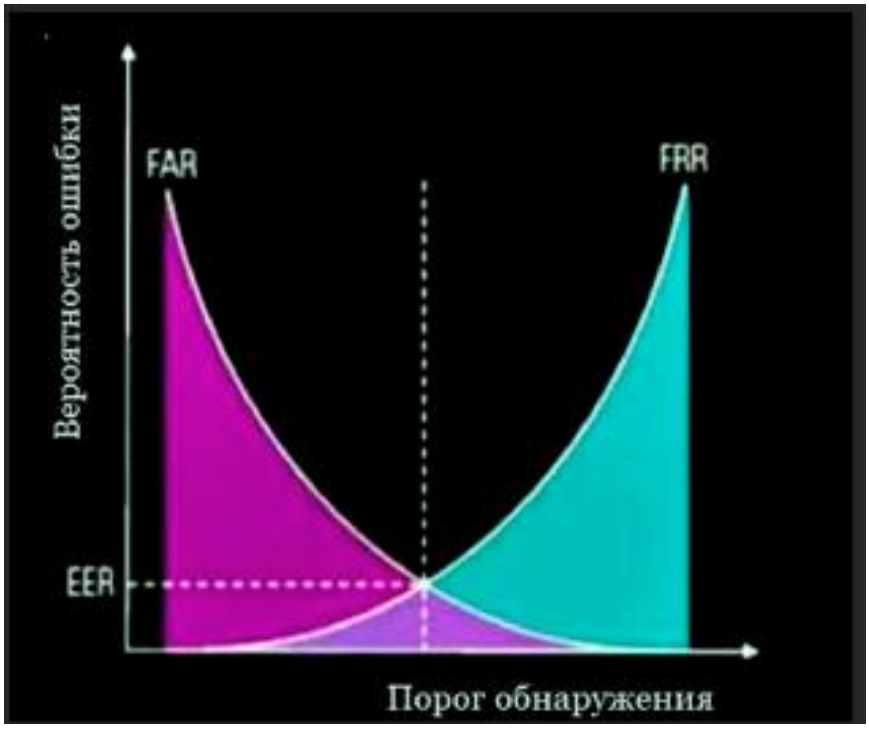

Рисунок 14 - График зависимости ошибки от порога обнаружения

Ошибки 2-го рода не несут ущерба при использовании алгоритмов для распознавания, однако ошибки 1-го рода являются действительно опасными, так как в случае их возникновения дают возможность злоумышленнику получить доступ к конфиденциальной информации.

В ходе выше описанного, будем считать, что способы достижения ошибок 1-го рода являются уязвимостями алгоритмов распознавания лиц.

В нынешний момент времени существуют следующие уязвимости алгоритмов распознавания:

1) Фото авторизованного пользователя. Злоумышленник может сфотографировать цель или найти фотографию в интернете в социальных сетях(U1).

2) Видеозапись авторизованного пользователя. Видеозапись может быть использована и для взлома более сложных алгоритмов распознавания, где будет проверяться подлинность, путем анализа человека с нескольких ракурсов(U2).

3) 3D макет. В значительной мере более сложный и финансово затратный вариант, включает в себя создание трехмерной модели объекта. Так как мы рассматриваем распознавание лица ─ объектом будет 3D модель головы. Совершенно понятен факт того, что получить слепок лица авторизованного пользователя будет довольно сложной задачей жизни, поэтому скорее всего злоумышленники используют нейронные сети, которые прогнозируют 3D форму лица по набору фотографий или по видеозаписям. После формирования 3D модели лица, злоумышленник может распечатать слепок на 3D принтере или сделать маску(U3).

4) Специальный макияж или грим. Используя профессиональный грим, злоумышленник может загримировать себя под сотрудника с которым у него есть общие черты(U4).

5) Случайное ложное срабатывание. В случае малого числа отличительных черт лица пользователя для распознавания, их легко можно спутать с похожими на него другими чертами(U5).

6) Взлом алгоритмов распознавания. Самая сложная в своей реализации уязвимость, которая предполагает использование внутренних уязвимостей самого алгоритма распознавания(U6).

Увеличить защищенность алгоритма можно следующим образом:

1) Анализ нескольких ракурсов изображения объекта. Проанализировав не один кадр, а к примеру временной ряд и по временному ряду оценивать подлинность пользователя. Недостатком такого метода является необходимости в дополнительном времени на проверку.

2) Совершать определенных действий пользователем. Он схож с прошлым пунктов, только анализируется не любое изменение ракурса, а подобающее поведение человека требуемым действиям.

3) Проверка фона изображения. Данная мера защиты актуальна при расположении камеры в определенном месте. Если камера будет детектировать факт того, что фон отличается от обычного, то это может значить, что используется фотография или видео объекта.

4) Искажение при движении. Если злоумышленник использует устройства для того, что показать изображение камере, то могут возникать блики и искажения изображения, такое можно задетектировать. Также есть возможно обнаружить использование злоумышленником распечатанной фотографии, с помощью того, что в каждой камере имеется линза и при движении объекта меняется фокусное расстояние, в линзе изображение будет искажаться, и, проведя анализ всех этих изменений в изображении, можно понять, что перед камерой плоское изображение.

5) Распределение освещения. Распределение освещения на объекте и на окружении объекта позволит узнать, является ли настоящим объект, находящийся перед камерой.

6) Добавление отличительных черт объекту. Если добавить объекту оригинальную цветную наклейку, то это в значительной мере повысит вероятно верно распознать объект.

7) Использование специализированных камер. К примеру ИК-камер или 3D-камер. Заключается в эксплуатации специального оборудования. Лучший результат дает комбинирование одновременно несколько типов таких камер. Такие камеры не требуют ничего особенного со стороны пользователей, но их использование требует больших затрат.

8) Дополнительные способы идентификации. Использование сразу несколько биометрических способ распознавания пользователей. К примеру добавить распознавание по отпечаткам пальцев.

В таблице ниже будет представлен сравнительный анализ эффективности мер защиты. В ходе анализа было выявлено, что дополнительные способы идентификации перекрывают все уязвимости алгоритма распознавания лиц, но этот способ крайне финансово затратный. Другой не менее эффективной мерой защиты является совершение определенных действий пользователем, эта мера перекрывает 3 из 6 уязвимостей, которые были выявлены в ходе анализа. При этом является простой в реализации и низкой по стоимости.

Для алгоритмов распознавания лиц, такие способ защиты как добавление отличительных черт и проверка фона или распределение освещения, являются самыми оптимальными и перекрывают 3 из 5 выявленных уязвимостей.

Таблица -1. Сравнительный анализ эффективности мер защиты

Методы защиты	**Перекрываемые уязвимости**	**Простота реализации**	**Стоимость**
Анализ нескольких ракурсов изображения	U1	Низкая	Низкая
Совершение определенных действий пользователем	U1,U2,U3	Низкая	Низкая
Проверка фона	U1,U2	Средняя	Низкая
Искажение при движении	U1,U2	Высокая	Низкая
Распределение освещения	U1,U2	Средняя	Низкая
Добавление отличительных черт объекту	U5	Низкая	Низкая
Использование специальных камер(ИК, 3D)	U1,U2	Низкая	Высокая
Дополнительные способы идентификации	U1,U2,U3,U4,U5,U6	Низкая	Высокая

3. ПРОГРАММНАЯ РАЗРАБОТКА ПРОЕКТА

3.1. Формирование базы данных

После того, как мы выбрали все необходимое программное обеспечение, нам необходимо загрузить библиотеки, которые будут использоваться при работе с алгоритмом.

Открываем командную строку, весь процесс установки библиотек проходят через нее.

- pip install cv2 — библиотека компьютерного зрения OpenCV.
- pip install pillow — добавляет возможность обработки изображений в интерпретатор Python.
- pip install cmake — используется для управления процессом компиляции программного обеспечения с помощью простых файлов конфигурации, которые не зависят от платформы компилятора.
- pip install numpy — используется для численных обработок в алгоритме.
- pip install dlib — библиотека машинного обучения, функционал которой будет активно эксплуатироваться в ходе работы с алгоритмом.
- pip install imutils — библиотека для работы с изображениями. В ходе разработки алгоритма будет использоваться для перевода изображения в необходимый нам формат.
- pip install pickle — реализует бинарные протоколы для социализации данных. В нашем случае будет превращать информацию, считанную с изображения в поток байтов и обратно в объекты.

Первым этапом в разработке алгоритма было принято сделать формирование базы данных из положительных и негативных изображений для тренировки своего собственного HAAR- каскада.

Для удобства мной был написан алгоритм, который работает следующим образом:

Первоначальным этапом были прописаны на каких настройках будет работать камера. Стоит отметить, что эти настройки можно менять в зависимости от необходимости.

Мои настройки выбраны для оптимального формирования базы данных.

```
myPath = 'Res/images' #Картинки с лицомом
cameraNo = 0
cameraBrightness = 190
moduleVal = 10
minBlur = 500
grayImage = False
SaveData = True
imgWidth = 180
imgHeight = 120




#########

global countFolder
cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)
cap.set(10,cameraBrightness)
```

Рисунок 15 – Настройки камеры

- myPath — путь в котором будут сохранятся наши изображения.
- cameraNo — это номер нашей камеры в системе. При необходимости есть возможность подключить другие устройства и просто заменить номер, чтобы также успешно использовать алгоритм.
- cameraBrightness — это настройка яркости камеры, которая будет выводится на экран.

Остальные параметры ориентированы по большей части на разрешение в котором камера будет выводится к нам в рабочее пространство, а также остальные необходимые настройки, как например минимальный показать блюра, который влияет на «смазанность» изображения.

- cap = cv2.VideoCapture(0) — выводит камеру на дисплей.

```
def saveDataFunc():
    global countFolder
    countFolder = 0
    while os.path.exists(myPath+ str(countFolder)):
        countFolder = countFolder + 1
    os.makedirs(myPath + str(countFolder))

if SaveData:saveDataFunc()
```

Рисунок 16 – указание пути сохранения изображений

Функция «saveDataFunc» выполняет задачу фиксацию пути в который будут сохраняться изображения, снятые на вебкамеру нашего устройства.

```
while True:

    success, img = cap.read()
    img = cv2.resize(img,(imgWidth,imgHeight))
    if grayImage: img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    if SaveData:
        blur = cv2.Laplacian(img,cv2.CV_64F).var()
        if count % moduleVal ==0 and blur > minBlur:
            nowTime = time.time()
            cv2.imwrite(myPath+ str(countFolder)+
                        '/'+ str(countSave)+" "+ str(int(blur))+ " "+str(nowTime)+".png",img)
            countSave+=1
        count += 1
        cv2.imshow("Image",img)

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

Рисунок 17 – Изменение изображения под необходимый формат

- img = cv2.resize(img,(imgWidth,imgHeight)) — изменение изображения под заданный нами раннее формат.

Это необходимо сделать для удобства хранения большого количества файлов, которые мы будем использовать при формировании HAAR-каскада.

Также в части кода происходит конвертация изображений в необходимый для чтения и поиска ключевых точек формат BGR2GRAY.

Также для того, что оптимизировать этот алгоритм, нужно было задать определенные ограничения для отброса некачественных изображений, как раз таки тут нам понадобился параметр блюра, который автоматически отбрасывает некачественные изображения.

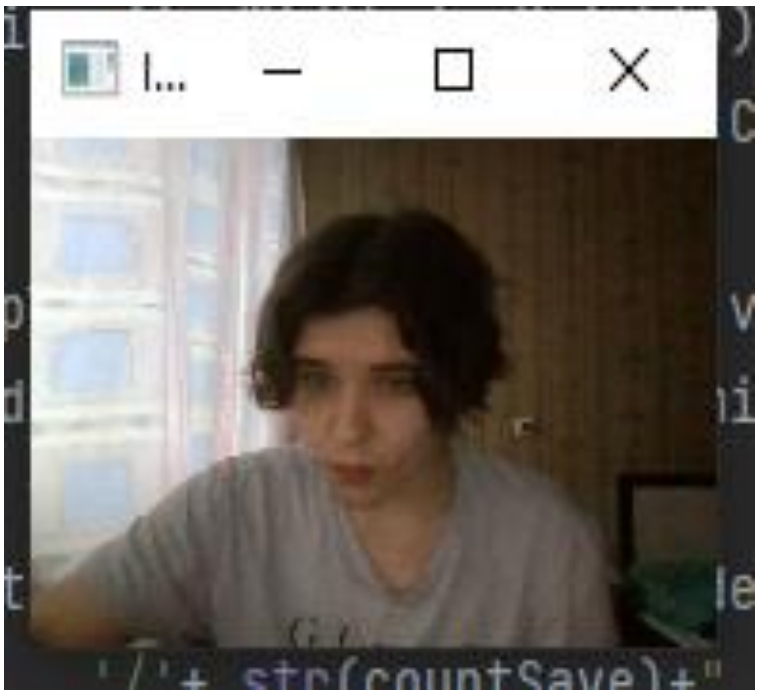

Рисунок 18 – Вывод камеры для фиксации изображений

В ходе этого этапа мы формируем базу данных из позитивных и негативных изображений нашего лица/лиц.

Важным критерием для формирования негативных изображений является факт того, что на фотографиях не должно быть нашего лица. Оно должно находится только на положительных образцах.

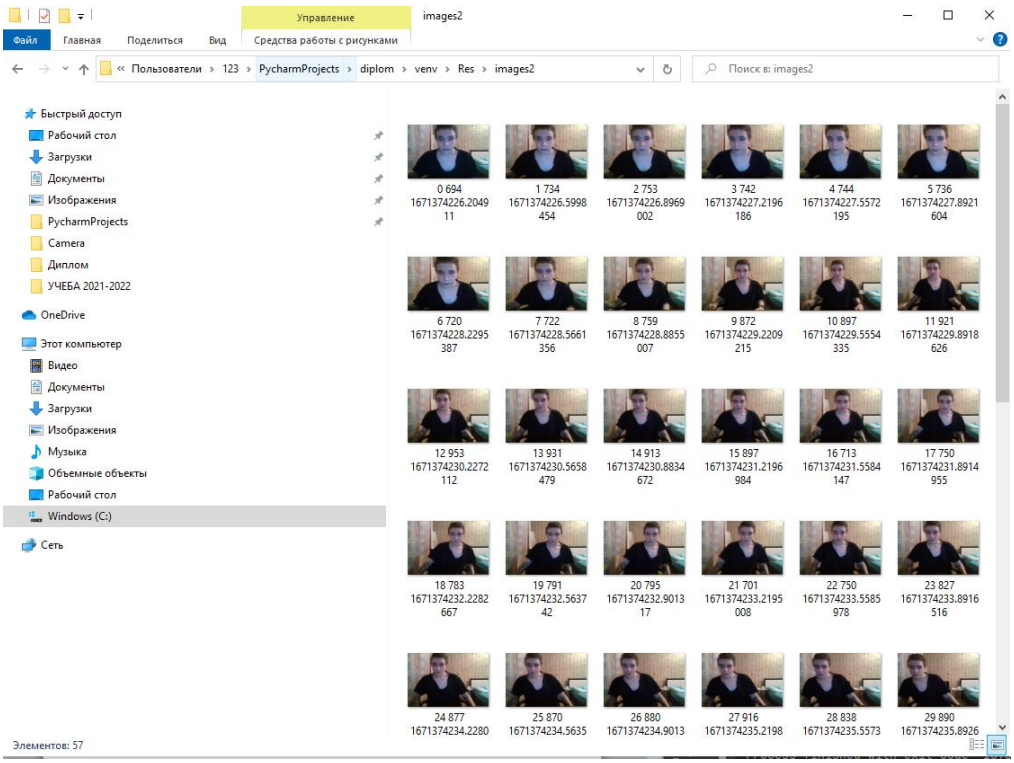

Рисунок 19 – Сохраненные алгоритмом фотографии

3.2. Создание собственного XAAP-каскада для распознавания лиц

Для формирования каскада будет использоваться программа Cascade Trainer GUI.

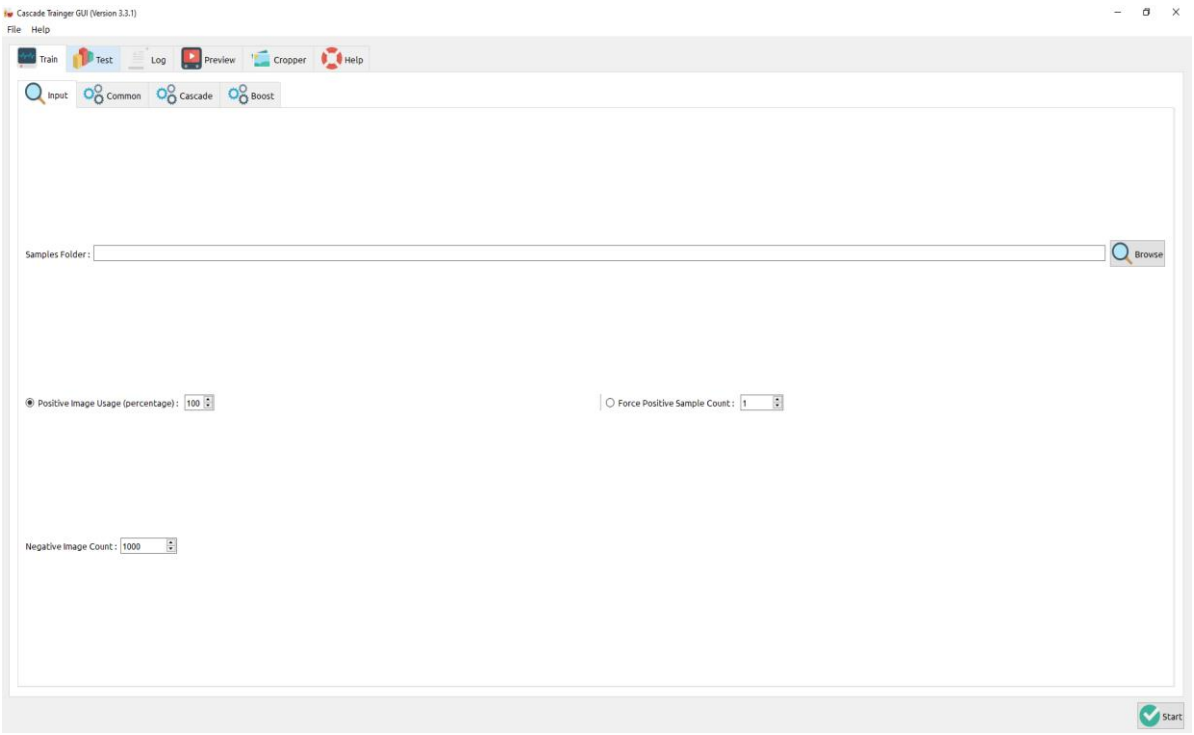

Рисунок 20 – Рабочий интерфейс программы

Для любой задачи обнаружения нам понадобится набор положительных и негативных образцов.

Функция каскадного классификатора при работе с OpenCV это показывать технологии на что смотреть на изображениях.

Сколько изображений на нужно, зависит от множества факторов, которые включают в себя:

- Качество изображений;
- Объект, который мы хотим распознать;
- Метод создания образцов;
- Мощность процессора.

Для своего обучения были использованы 60 положительных образцов и 40 отрицательных.

Мы создаем позитивные и негативные образцы с помощью описанного выше алгоритма и сохраняем их в соответствующие папки.

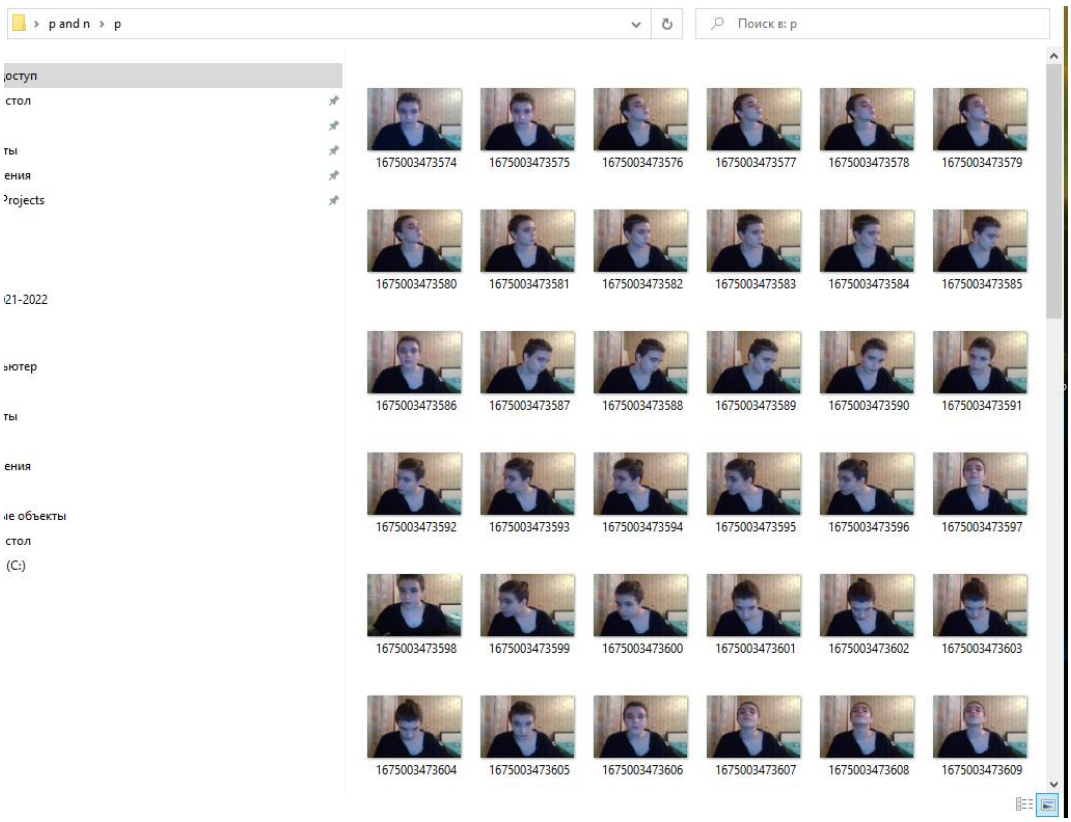

Рисунок 21 – Позитивные изображения

Положительные образы должны находиться в папке с именем «p». Положительными образцами являются изображения, содержащие объект, который мы хотим, чтобы наш алгоритм обнаружил.

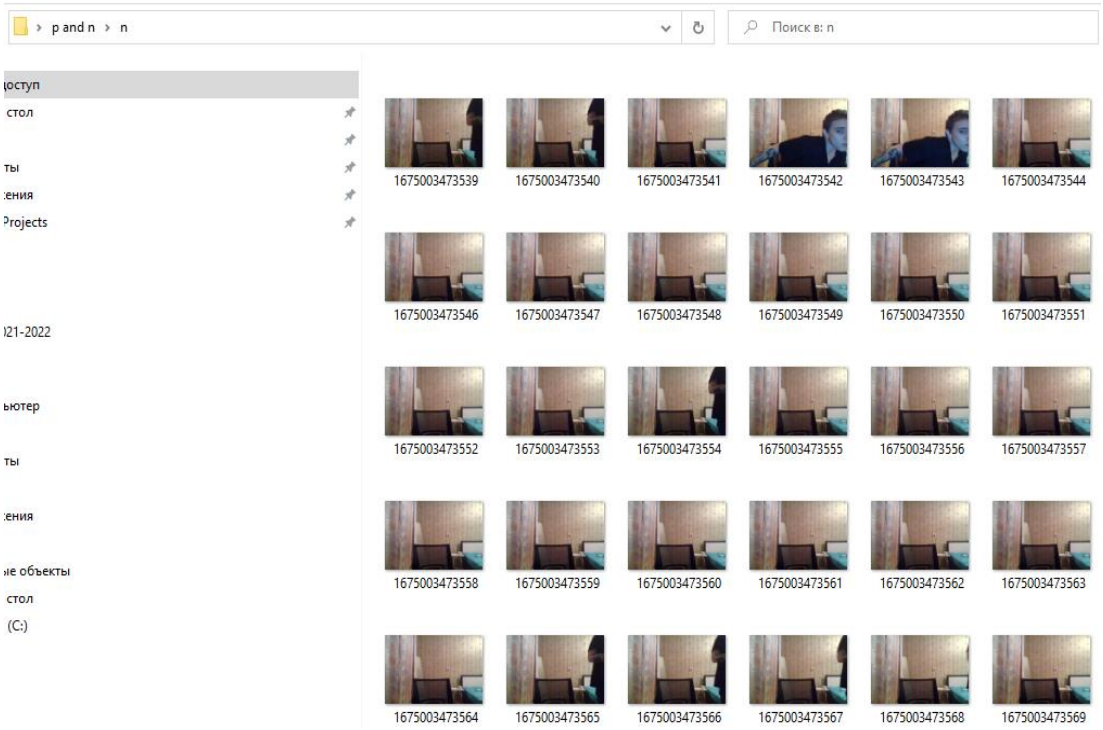

Рисунок 22 – Негативные изображения

Негативные образы — это изображения, которые не должны включать объект, которым мы хотим обнаружить. Это не означает, что отрицательные образы являются чем-то совершенно другим.

Вообще отрицательный образ — это образ, в котором содержится часть объекта, которую мы хотим обнаружить. Процент этой части должен быть примерно 20% от оригинала объекта, который мы хотим обнаружить.

Помня об этом, мы можем взять части лица, некоторые части фона, чтобы создать отрицательные образы.

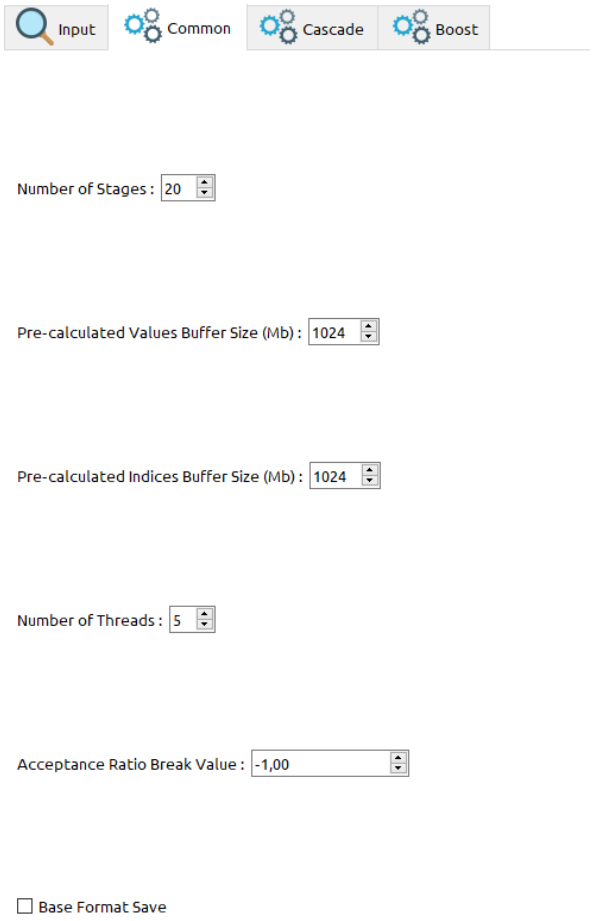

Рисунок 23 – Настройки формирования каскада

В этом разделе мы устанавливаем итерации обучения. В общем, большее количество итерация улучшают точность обнаружения классификатора. Но с увеличением количества итераций увеличивается требуемое время для построения и обучения классификатора.

Затем мы устанавливаем размер буфера предварительного расчета равным скорости тренировочного процесса.

Мы можем назначить столько памяти, сколько нам будет необходимо. Однако в этом случае все упирается в ресурсы нашего компьютера.

Также в программе мы можем установить тип функции HAAR или LBP.

Классификаторы HAAR очень точны, но требуют гораздо больше времени для обучения. Классификаторы LBP быстрее, но при обучении каскада для определения лица, точно будет играть ключевую роль.

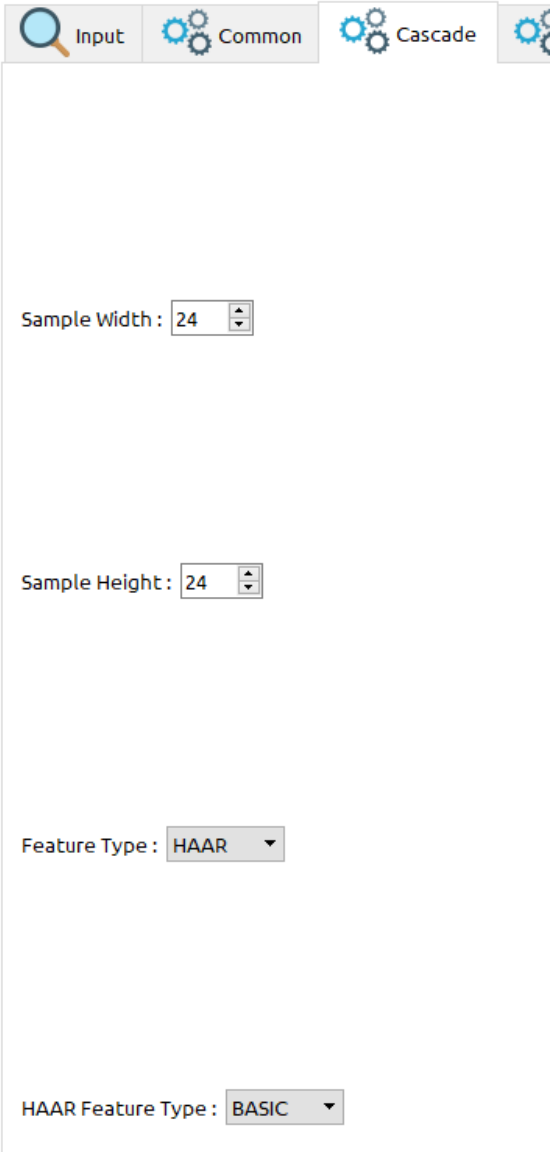

Рисунок 24 – Выбор типа функции

После того, как все параметры заданы, нажимаем кнопку «Start», чтобы начать обучения нашего каскадного классификатора.

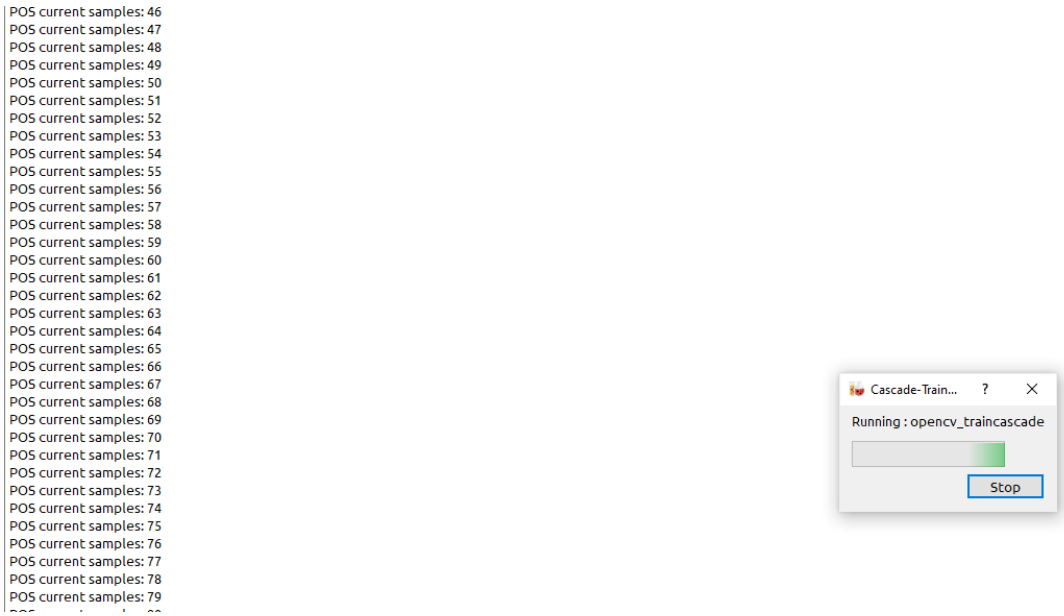

Рисунок 25 – Журнал тренировки

После тренировки классификатора в выбранной нами папке появится папка с обученным классификатором, который мы и будем в дальнейшем использовать в нашем алгоритме.

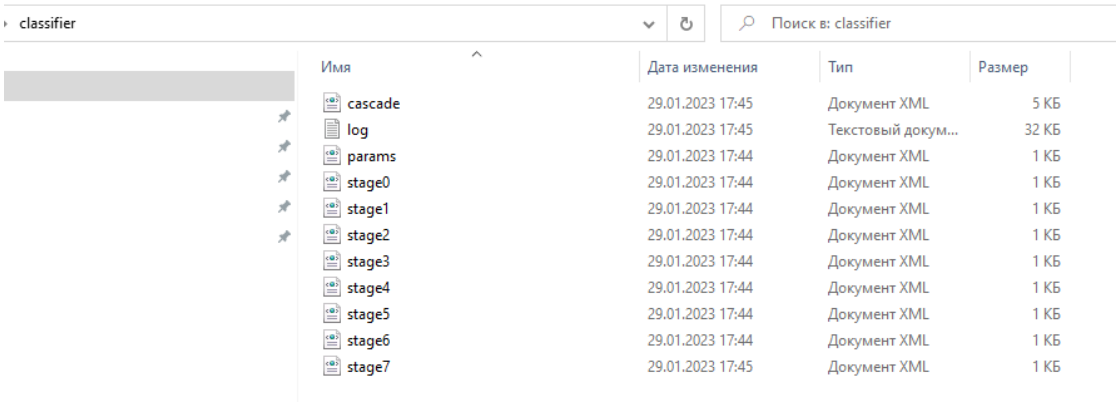

Рисунок 26 – Готовый классификатор

Также в этой же папке появятся файлы со стадиями, которые проходил наш классификатор в ходе его обучения.

Их также можно посмотреть в логах и ориентируясь на это редактировать базы с положительными/негативными образами или же корректировать внутренние настройки программы.

3.3. Создание алгоритма распознавания лиц

Следующим этапом нашей работы будет создания непосредственно самого алгоритма создания лиц. Однако он также будет включать в себя 2 этапа.

Первым этапом будет создания файла с ключевыми точками лица определенного человека.

```
imagePaths = list(paths.list_images('Image'))
knownEncodings = []
knownNames = ["ilya"]
```

Рисунок 27 – Известные пользователи

В этой части мы задаем папку с изображениями для идентифицируемого пользователя и задаем известных для алгоритма людей.

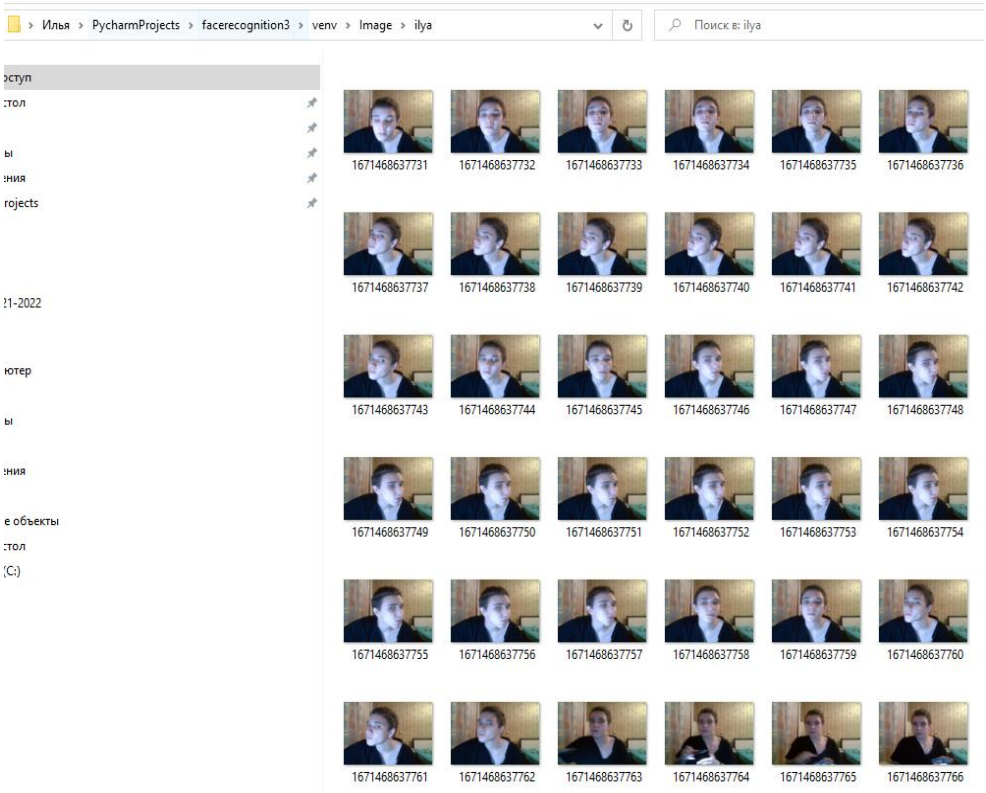

Рисунок 28 – База данных идентифицируемого пользователя

Данное разграничение необходимо для реализации системы свой/чужой. Так в последствии будет видно, что пользователи, не входящие в базу данных алгоритма, будут отображаться, как Unknown(неизвестный).

```
for (i, imagePath) in enumerate(imagePaths):
    name = imagePath.split(os.path.sep)[-2]
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    boxes = face_recognition.face_locations(rgb, model='hog')
    encodings = face_recognition.face_encodings(rgb, boxes)
    for encoding in encodings:
        knownEncodings.append(encoding)
        knownNames.append(name)
data = {"encodings": knownEncodings, "names": knownNames}
f = open("face_enc", "wb")
f.write(pickle.dumps(data))
f.close()
```

Рисунок 29 – Цикл для перебора изображений и записи их в текстовый файл

Первым дело мы извлекаем имя человека из соответствующей папки с помощью следующей команды:

For (i , imagePath) in enumerate(imagePaths):

name = imagePath.split(os.path.sep)[-2]

Далее мы загружаем изображения из папки и конвертируем его из BGR (OpenCV ordering) в dlib ordering(RGB):

rgb = cv2.cvtColor(image, cv2.COLOR_BRG2RGB)

Далее используя библиотеку face_recogniton для обнаружения основных элементов лица. После это мы вычисляем эмбендинги для каждого лица, которое было обнаружено на фотографиях.

Эмбендинг — перевод данных в текстовый формат, который будет в последствии считывать компьютер. Использование данного способа обусловлено тем, что это в значительной мере повышает скорость работы алгоритма, так как компьютер будет брать данные о положении лица из заранее описанного источника и применять их в реальном времени.

Затем мы сохраняем данные в файл, которые в последствие будет считывать алгоритм.

- face_enc — файл с текстовой интерпретацией ключевых точек лица на фотографии.

```
cascPathface = os.path.dirname(cv2.__file__) + "/data/cascade.xml"

faceCascade = cv2.CascadeClassifier(cascPathface)

data = pickle.loads(open('face_enc', "rb").read())

print("Streaming started")
video_capture = cv2.VideoCapture(0)
```

Рисунок 30 – Чтение каскада и файла с точками лица

Следующее, что необходимо сделать это загрузить обученный нами каскад Хаара в каскадный классификатор с помощью команды:

cascPathface = os.path.dirname(cv2.__file__) + «/data/cascade.xml»

Затем загрузить известные лица и вложения, которые мы сохраняли в файл face_enc.

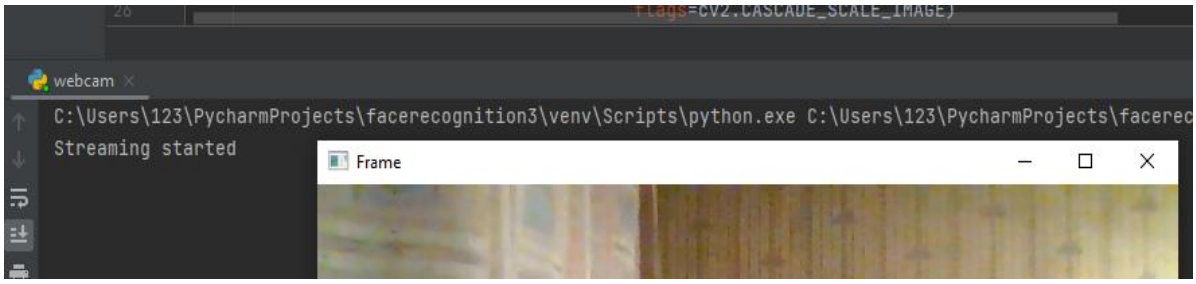

Рисунок 31 – Захват камеры и надпись о том, что программа начала свою работу

Чтобы начать считывание данных с видеотрансляции, нам нужно осуществить захват видеофайла из потока и начать его конвертировать в необходимый формат в реальном времени.

```
while True:

    ret, frame = video_capture.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray,
                                         scaleFactor=1.1,
                                         minNeighbors=5,
                                         minSize=(60, 60),
                                         flags=cv2.CASCADE_SCALE_IMAGE)


    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    encodings = face_recognition.face_encodings(rgb)
    names = [ilya]
```

Рисунок 32 – Чтение изображения с веб-камеры и его конвертация

Теперь, когда алгоритм начал считывать детали нашего лица в реальном времени, необходимо задать ему задачу по сравнению кодировок, которые он видит в реальном времени с кодировками, которые мы задали в текстовом формате для определенных пользователей.

Это позволит алгоритму реализовать в себе систему свой/чужой, по скоку после прочтения и анализа кодировок, алгоритм сможет определить записан ли пользователь в нашу базу данных.

Если алгоритмом пользуется идентифицированный пользователь, записанный в базу данных, то будет выведено его имя, а если человек, который не записан, то будет выведено «Unknown».

```
for encoding in encodings:

    matches = face_recognition.compare_faces(data["encodings"],
                                             encoding)
    name = "Unknown"
```

Рисунок 33 – Если нет совпадений в кодировках

В случае совпадения кодировок, нам нужно провести поиск и сохранить их, как распознанные лица.

После выполнения этого действия будет проведена проверка во внутренней базе данных на совпадение с пользователем с определенным именем.

Проверка осуществляется по индексам, которые были сохранены в текстовый файл ранее.

Проверяются все пользователи и выводится имя того с кем было найдено наибольшее количество совпадений и обновляется список имен.

```
if True in matches:

    matchedIdxs = [i for (i, b) in enumerate(matches) if b]
    counts = {}
```

Рисунок 34 – Проверка совпадений пользователей во внутренней базе данных

```
        for i in matchedIdxs:

            name = data["names"][i]

            counts[name] = counts.get(name, 0) + 1

        name = max(counts, key=counts.get)
```

Рисунок 35 – Нахождение определённого пользователя

Если в ходе работы всех действий был определен человек на веб-камере, то алгоритм начинает фиксацию на лице, рисуя прямоугольник, параметры которого также задаются вручную.

Если в ходе сравнения всех кодировок, мы находим пользователя, записанного в базу данных, то он будет выводится с заданным ему именем.

Если нет и пользователь системе неизвестен, то будет выведена плашка «Unknown».

```
        names.append(name)

        for ((x, y, w, h), name) in zip(faces, names):

            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(frame, name, (x, y), cv2.FONT_HERSHEY_SIMPLEX,
                        0.75, (0, 255, 0), 2)
    cv2.imshow("Frame", frame)
    if cv2.waitKey(1) &  0xFF == ord('q'):
        break
video_capture.release()
cv2.destroyAllWindows()
```

Рисунок 36 – Фиксация пользователя

После прохождения всех вышеперечисленных действий, система является готовой к распознаванию данных. Запускаем её в нашей среде разработки. Через непродолжительное время включится веб-камера, которая готовая к работе.

Смотрим в объектив камеры, если мы занесены в созданную базу данных, то над обведенной рамкой будет показываться имя пользователя.

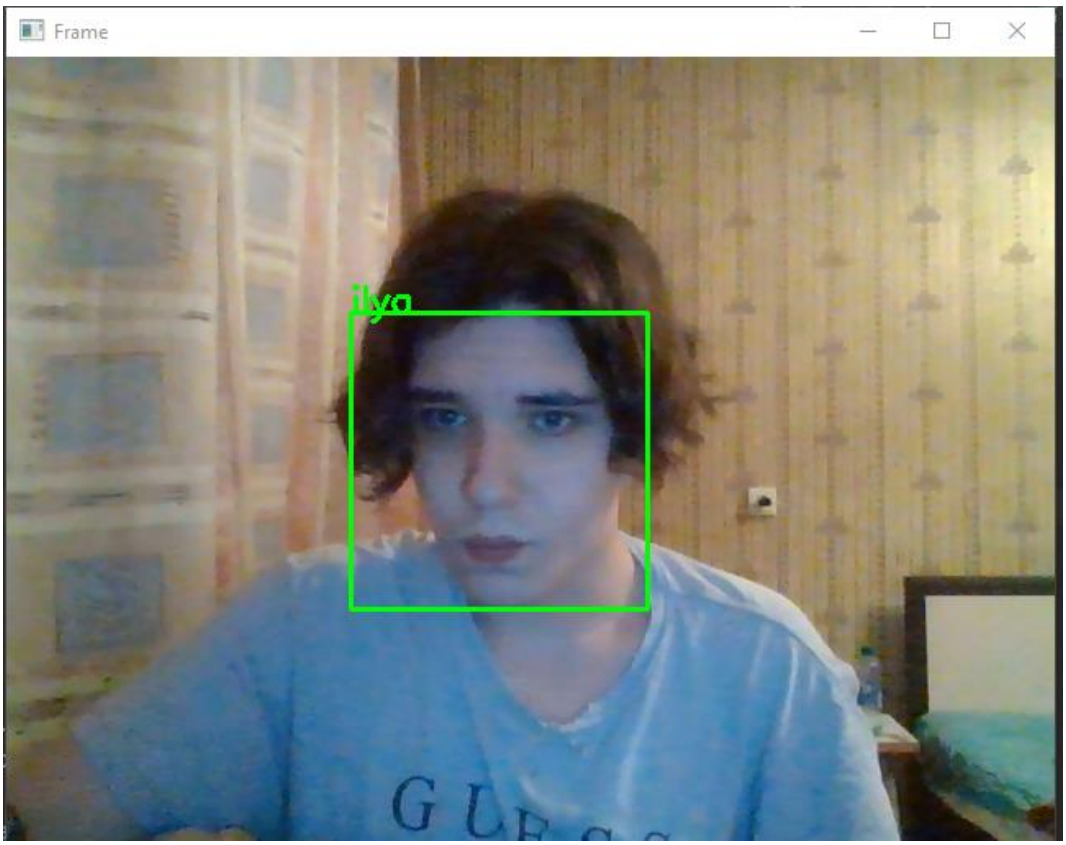

Рисунок 37 – Распознавание пользователя занесенного в базу данных

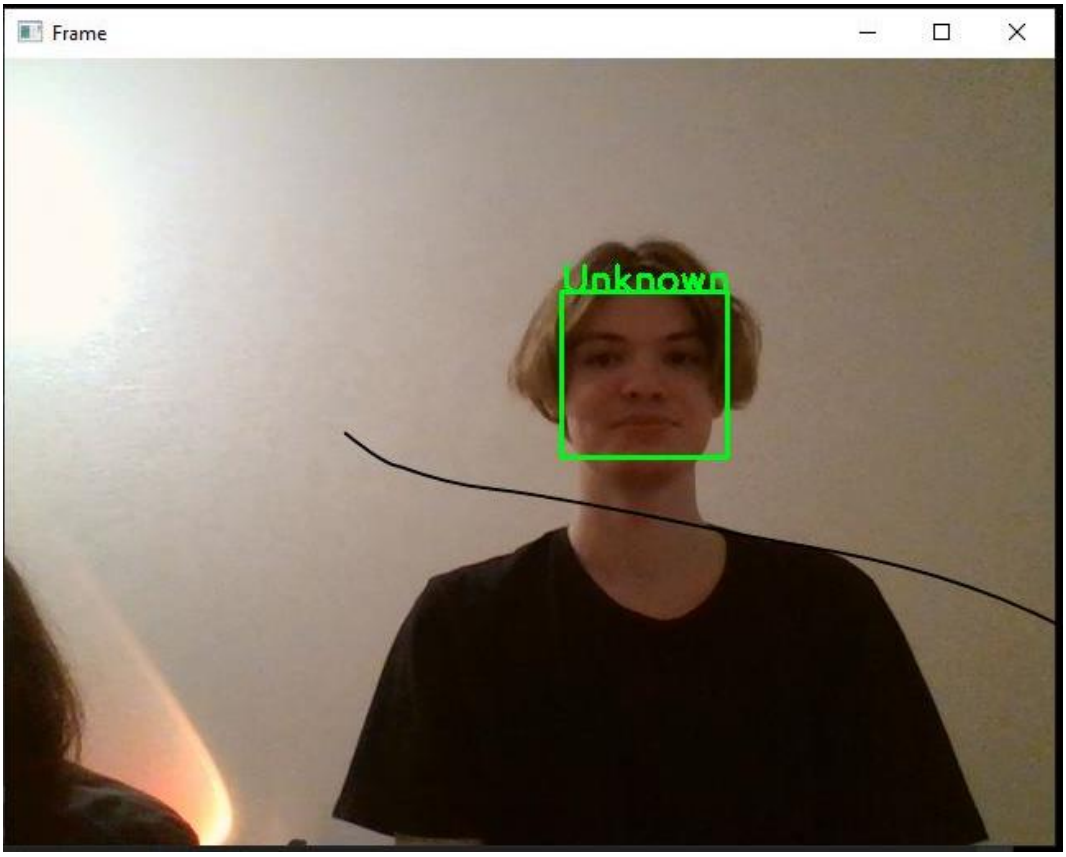

Рисунок 38 – Распознавание неизвестного для базы данных пользователя

Одной из главной задач этой ВКР остается обеспечение безопасности распознавания. Хоть в этой системе и есть значительное количество плюсов, но минусы не исключаются, как и в любой другой системе.

У алгоритма есть ряд неточностей, как например: Распознавание пользователя со сделанной фотографии, будь то распечатанная или же демонстрируемая с другого электронного устройства(телефон).

Для повышения фактора защиты, необходимо ввести дополнительные способы аутентификации зарегистрированных пользователей.

Многими компаниями или сервисами сейчас используется технология двухфакторной или многофакторной аутентификации. Это активно эксплуатируется на пропускных пунктах или же при обеспечение банковской безопасности. Подобного рода системы, могут быть представлены как с использованием одного метода или же использовать сразу несколько способов идентификации. Эти системы получили название гибридных.

Если в ходе распознавания по лицу используются ещё и пароли или пропуски, злоумышленник может разузнать о дополнительном факторе защиты и попытаться осуществить несанкционированный доступ к информации.

Если же используется сразу несколько факторов в рамках одного метода, то сделать это будет близким к невозможности.

Алгоритм распознавания по лицу, хоть и является недостаточно надежным, имеет некоторые достоинства.Встроенные в библиотеки инструменты, также могут позволять определять жесты человека. Если задать алгоритму определенные шаблоны жестов, которые можно использовать для получения дополнительного фактора идентификации.

Можно задать определенному человеку, определенный жест или последовательность жестов.

Четко выверенную последовательность из жестов предугадать крайне трудно, даже учитывая минусы алгоритма.

ЗАКЛЮЧЕНИЕ

При написании дипломной работы был разработан алгоритм распознавания лиц и реализован в среде разработки.

В ходе выполнения были получены данных о видах и принципах работы алгоритмов распознавания лиц, были найдены преимущества его и недостатки, а также проведен анализ основных уязвимостей и способы повышения его защищенности. На основе всех этих данных был выбран наиболее актуальный способ реализации алгоритма, программное обеспечение и язык программирования для реализации проекта.

На основе изученных технологий была обучена собственная база данных для определения лиц идентифицированных пользователей. Произведено тестирование работоспособности алгоритма и оценка его безопасности.

После всего этого, на основе полученной информации разработана методика обеспечения безопасности идентификации, на основе знаний об основных минусах и уязвимостях полученного алгоритма.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ Р 50.1.053-2005. Информационные технологии. Основные термины и определения в области технической защиты информации : дата введения 2006-01-01. – М. : Стандартинформ, 2005.

2. Попов, Г. А. Разработка программного комплекса видеоконтроля объектов или лиц на территории организации / Г. А. Попов, Т. А. Попова // Актуальные вопросы информационной безопасности регионов в условиях перехода России к цифровой экономике : материалы VII Всероссийской научнопрактической конференции, г. Волгоград, 26–27 апр. 2018 г. – Волгоград : Изд-во ВолГУ, 2018. – С. 285–289.

3. PYTHONIST [Электронный ресурс]. – Режим доступа: https://pythonist.ru/raspoznavanie-licz-pri-pomoshhi-python-i-opencv/#facedetection. – Дата доступа: 23.12.2022.

4. Кухарев, Георгий Биометрические системы: Методы и средства идентификации личности человека / Георгий Кухарев. – : Политехника, 2001. – 240 с.

5. Wikipedia [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Теория_распознавания_образов. – Дата доступа: 23.12.2022.

6. Real Python [Электронный ресурс]. – Режим доступа: https://realpython.com/face-recognition-with-python/. – Дата доступа: 24.12.2022.

7. OpenCV [Электронный ресурс]. – Режим доступа: https://opencv.org/. – Дата доступа: 25.12.2022.

8. Вапник, В.Н Теория распознавания образов (статистические проблемы обучения) / В.Н Вапник. – : Наука, 1974. – 416 с.

9. Face Detection and Recognition Using Hidden Markov Models [Электронный ресурс]. – Режим доступа: http://www.anefian.com/research/nefian98_face.pdf. – Дата доступа:

10. Face Recognition by Elastic Bunch Graph Matching [Электронный ресурс]. – Режим доступа: https://www.face-rec.org/algorithms/ebgm/wisfelkrue99-facerecognition-jainbook.pdf. – Дата доступа: .

11. DeepFace Closing the Gap to Human [Электронный ресурс]. – Режим доступа: https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf. – Дата доступа: .

12. CASCADE TRAINER GUI [Электронный ресурс]. – Режим доступа: https://amin-ahmadi.com/cascade-trainer-gui/. – Дата доступа: .

13. Метод Виолы-Джонса (Viola-Jones) как основа для распознавания лиц [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/133826/. – Дата доступа: .

ПРИЛОЖЕНИЕ 1

Программа для формирования базы данных:

```
import cv2
import os
import time
###############
myPath = 'Res/images' #Картинки с лицомом
cameraNo = 0
cameraBrightness = 190
moduleVal = 10
minBlur = 500
grayImage = False
SaveData = True
imgWidth = 180
imgHeight = 120
#########
global countFolder
cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)
cap.set(10,cameraBrightness)
count = 0
countSave = 0
def saveDataFunc():
    global countFolder
    countFolder = 0
    while os.path.exists(myPath+ str(countFolder)):
        countFolder = countFolder + 1
```

```
    os.makedirs(myPath + str(countFolder))
if SaveData:saveDataFunc()
while True:
    success, img = cap.read()
    img = cv2.resize(img,(imgWidth,imgHeight))
    if grayImage: img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    if SaveData:
        blur = cv2.Laplacian(img,cv2.CV_64F).var()
        if count % moduleVal ==0 and blur > minBlur:
            nowTime = time.time()
            cv2.imwrite(myPath+ str(countFolder)+
                        '/'+ str(countSave)+" "+ str(int(blur))+ "
"+str(nowTime)+".png",img)
            countSave+=1
        count += 1
        cv2.imshow("Image",img)
    if ord('q') == cv2.waitKey(1)  &  0xFF:
        break
cap.release()
cv2.destroyAllWindows()
```

ПРИЛОЖЕНИЕ 2

Программа для обучения алгоритма:

```
from imutils import paths
import face_recognition
import pickle
import cv2
import os
# в директории Images хранятся папки со всеми изображениями
imagePaths = list(paths.list_images('Image'))
knownEncodings = []
knownNames = ["ilya"]
# перебираем все папки с изображениями
for (i, imagePath) in enumerate(imagePaths):
    # извлекаем имя человека из названия папки
    name = imagePath.split(os.path.sep)[-2]
    # загружаем изображение и конвертируем его из BGR (OpenCV ordering)
    # в dlib ordering (RGB)
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    # используем библиотеку Face_recognition для обнаружения лиц
    boxes = face_recognition.face_locations(rgb, model='hog')
    # вычисляем эмбеддинги для каждого лица
    encodings = face_recognition.face_encodings(rgb, boxes)
```

```
    # loop over the encodings
    for encoding in encodings:
        knownEncodings.append(encoding)
        knownNames.append(name)
# сохраним эмбеддинги вместе с их именами в формате словаря
data = {"encodings": knownEncodings, "names": knownNames}
# для сохранения данных в файл используем метод pickle
f = open("face_enc", "wb")
f.write(pickle.dumps(data))
f.close()
```

ПРИЛОЖЕНИЕ 3

Алгоритм распознавания с веб-камеры:

```
import face_recognition
import imutils
import pickle
import time
import cv2
import os
#найти путь к файлу xml, содержащему файл haarcascade
cascPathface = os.path.dirname(cv2.__file__) +
"/data/haarcascade_frontalface_alt2.xml"
# загрузить harcaascade в каскадный классификатор
faceCascade = cv2.CascadeClassifier(cascPathface)
# загрузить известные лица и вложения, сохраненные в последнем файле
data = pickle.loads(open('face_enc', "rb").read())
print("Streaming started")
video_capture = cv2.VideoCapture(0)
# зацикливаемся на кадрах из потока видеофайла
while True:
    # захватить кадр из потокового видеопотока
    ret, frame = video_capture.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray,
                                         scaleFactor=1.1,
                                         minNeighbors=5,
                                         minSize=(60, 60),
                                         flags=cv2.CASCADE_SCALE_IMAGE)
    # преобразовать входной кадр из BGR в RGB
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    # эмбеддинг лица для лица на входе
```

```
    encodings = face_recognition.face_encodings(rgb)
    names = []
    # эмбеддинг лица для лица на входе
    # у нас есть несколько вложений для нескольких лиц
    for encoding in encodings:
        # Сравните кодировки с кодировками в data["encodings"]
        # Совпадения содержат массив с логическими значениями и значением
True для встраивания, которое близко соответствует
        # и False для остальных
        matches = face_recognition.compare_faces(data["encodings"],
                                                 encoding)
        # установить имя =неизвестно, если ни одна кодировка не совпадает
        name = "Unknown"
        # проверяем, нашли ли мы совпадение
        if True in matches:
            matchedIdxs = [i for (i, b) in enumerate(matches) if b]
            counts = {}
            for i in matchedIdxs:
                name = data["names"][i]
                counts[name] = counts.get(name, 0) + 1
            name = max(counts, key=counts.get)
        # обновить список имен
        names.append(name)
        for ((x, y, w, h), name) in zip(faces, names):
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(frame, name, (x, y), cv2.FONT_HERSHEY_SIMPLEX,
                        0.75, (0, 255, 0), 2)
    cv2.imshow("Frame", frame)
    if cv2.waitKey(1) &  0xFF == ord('q'):
        break
```

```
video_capture.release()
cv2.destroyAllWindows()
```