МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО» ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

Квасов Устин Анатольевич

**Система контроля присутствия студентов на занятиях с использованием технологий компьютерного зрения**

Выпускная квалификационная работа (уровень бакалавриата)

Направления подготовки 09.03.04 Программная инженерия

Научный руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. А. Руденко

Доцент кафедры

К ЗАЩИТЕ ДОПУСКАЮ:

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Милюков

Симферополь, 2023

**Реферат**

Квасов У.А. Система контроля присутствия студентов на занятиях с использованием технологий компьютерного зрения. Выпускная квалификационная работа (уровень бакалавриата) по специальности 09,03,04 Программная инженерия, Кафедра программной инженерии и моделирования Физико- технического института Крымского федерального университета им. В. И. Вернадского. – Симферополь, 2023. – 78 с., 20 рис., 1 прил., 10 ист.

Объектом исследования работы является применение технологий машинного зрения в практических ситуациях. В случае данной работ при отметке посещаемости студентов.

Целью работы является написание программы для осуществления данной задачи.

В данной работе будут разобраны алгоритмы для отслеживания объектов и распознавания лиц, а также технологии их применяющие.

Результатом данной работы является готовая программа и описания технологий для ее создания

Данная работа несет теоретическую ценность для тех, кто хочет узнать о технологиях машинного зрения.

Областью применения данной работы можно считать предприятия (заведения) где необходимо проводить регистрацию посещения сотрудниками (и не только).

**Оглавление**

[Введение 5](#_Toc136872722)

[Задачи машинного зрения: 5](#_Toc136872723)

[Распознавание 6](#_Toc136872724)

[Идентификация 6](#_Toc136872725)

[Обнаружение 6](#_Toc136872726)

[Распознавание текста 6](#_Toc136872727)

[Оценка движения 7](#_Toc136872728)

[Восстановление сцены 7](#_Toc136872729)

[Восстановление изображений 7](#_Toc136872730)

[Глава 1 Теоретические основы 9](#_Toc136872731)

[1.1 Теория и История алгоритмов распознавания лиц 9](#_Toc136872732)

[1.1.1Алгоритмы распознавания 9](#_Toc136872733)

[1.1.2 История алгоритмов распознавания лиц. 11](#_Toc136872734)

[ LBPH Алгоритм (1996) 13](#_Toc136872735)

[1.2 Теория и история алгоритмов отслеживания объектов 16](#_Toc136872736)

[1.2.1 Алгоритмы обнаружения объектов 16](#_Toc136872737)

[1.2.2 История алгоритмов обнаружения лиц 18](#_Toc136872738)

[ Алгоритм Виола-Джонса (2001): 18](#_Toc136872739)

[1.3 Сравнительный анализ методов 25](#_Toc136872740)

[1.3.1Сравнительный анализ методов обнаружения объектов 25](#_Toc136872741)

[1.4 Выбранные методы 29](#_Toc136872742)

[Глава 2 Анализ современных методов и инструментов 32](#_Toc136872743)

[2.1 Анализ архитектуры ПО для распознавания лиц 32](#_Toc136872744)

[2.2 Библиотеки 34](#_Toc136872745)

[2.2.1 Обзор библиотек реализующих методы обнаружения объектов: 34](#_Toc136872746)

[2.2.2 Обзор библиотек реализующих методы распознавания лица: 36](#_Toc136872747)

[2.3 Сравнение библиотек и примеры имплементации 37](#_Toc136872748)

[2.3.1 Сравнение библиотек для отслеживания объектов 38](#_Toc136872749)

[2.3.1.1 Примеры имплементации 41](#_Toc136872750)

[2.3.2 Сравнение библиотек для распознавания лиц 47](#_Toc136872751)

[2.3.2.1 Примеры имплементации 50](#_Toc136872752)

[Глава 3 Проект 55](#_Toc136872753)

[3.1 Разработка архитектуры проекта 55](#_Toc136872754)

[3.2 Разработка интерфейса и программных модулей 57](#_Toc136872755)

[Завершение: 72](#_Toc136872756)

[Используемая литература 74](#_Toc136872757)

[Приложение 75](#_Toc136872758)

Введение

Машинное зрение — это научное направление в области искусственного интеллекта, в частности робототехники, и связанные с ним технологии получения изображений объектов реального мира, их обработки и использования полученных данных для решения разного рода прикладных задач без участия (полного или частичного) человека.

Исходя из общей тенденции к автоматизации процессов не только производства, но и других операций (в том числе и повседневных дел) можно смело утверждать об актуальности данного направления.

В подтверждение своей позиции рассмотрим области применения машинного зрения. Также рассмотрим какие задачи решает применение данной технологии.

Машинное зрение сосредотачивается на применении в промышленном комплексе. Примером может являться использование автономных роботов и систем визуальной проверки. Подразумевается, что технологии датчиков изображения связаны с обработкой видеоданных для управления роботом.

Обработка изображений и анализ изображений в основном сосредоточены на работе с 2D изображениями, т.е. как преобразовать одно изображение в другое. Например, попиксельные операции увеличения контрастности, операции по выделению краёв, устранению шумов или геометрические преобразования, такие как вращение изображения. Данные операции предполагают, что обработка/анализ изображения действуют независимо от содержания самих изображений.

Задачи машинного зрения:

1. Распознавание
2. Идентификация
3. Обнаружение
4. Распознавание текста
5. Восстановление 3D формы по 2D изображениям
6. Оценка движения
7. Восстановление сцены
8. Восстановление изображений
9. Выделение на изображениях структур определенного вида, сегментация изображений
10. Анализ оптического потока

Распознавание

Классическая задача в компьютерном зрении, обработке изображений и машинном зрении это определение содержат ли видеоданные некоторый характерный объект, особенность или активность.

Эта задача может быть достоверно и легко решена человеком, но до сих пор не решена удовлетворительно в компьютерном зрении в общем случае: случайные объекты в случайных ситуациях.

Один или несколько предварительно заданных или изученных объектов или классов объектов могут быть распознаны (обычно вместе с их двухмерным положением на изображении или трехмерным положением в сцене).

Идентификация

Задача при которой распознается объект обладающий набором информации. Как пример может служить распознавание лица или номера авто.

Обнаружение

Производится проверка на наличие определенного условия.

Распознавание текста

Производится обработка и анализ изображения содержащее текст. Оптическое распознавание знаков – есть распознавание символов печатного или рукописного текста.

Оценка движения

Отвечает за обработку последовательного ряда изображений (видеоряда) с целью оценки скорости движения объекта. Используется в отслеживании реальных объектов (по типу авто) или отслеживание виртуальных точек движения (при помощи этого создается движение камеры в 3D сценах).

Восстановление сцены

Задачей является восстановление трехмерной сцены из набора изображений с разных углов.

Восстановление изображений

Здесь задачей является обработка изображения для достижения необходимого результата. Здесь может использоваться как и удаление шумов так и восстановления участков поврежденных изображений

Исходя из актуальности целью работы является разработка программы позволяющей производить отметку посещаемости студента при помощи машинного зрения. Это позволит упростить или же полностью упразднить нынешний способ данной процедуры

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить теоретические основы
2. Анализировать современные методы и инструменты
3. Разработать архитектуру программы
4. Разработка интерфейса и программных модулей
5. Провести тестирование и анализ результатов

Глава 1 Теоретические основы

* 1. Теория и История алгоритмов распознавания лиц

1.1.1Алгоритмы распознавания лиц направлены на идентификацию и проверку лиц на основе их лицевых характеристик. Общая теория распознавания лиц включает два основных этапа: обнаружение лица и представление/верификация лица.

Ниже приводится обзор общей теории алгоритмов распознавания лиц:

**Обнаружение лица**:

Первым шагом в распознавании лиц является обнаружение присутствия и расположения лиц в кадре изображения или видео. Этот шаг включает в себя поиск областей во входных данных, которые, вероятно, содержат лица. Для обнаружения лиц могут использоваться различные алгоритмы, включая каскад Хаара, гистограмму ориентированных градиентов (HOG) и подходы на основе глубокого обучения, такие как конволюционные нейронные сети (CNN).

После обнаружения лица алгоритм определяет граничную область лица или ориентиры (ключевые точки), которые определяют положение и форму лица.

**Представление лица и извлечение признаков**:

После обнаружения лица следующим шагом является извлечение характеристик или представлений из обнаруженных лиц. Целью является создание компактного и отчетливого представления, которое отражает уникальные характеристики каждого лица.

* 1. **Традиционные методы**: Традиционные методы распознавания лиц используют созданные вручную признаки, такие как локальные бинарные паттерны (LBP), собственные лица или лица Фишера. Эти методы преобразуют изображение лица в низкоразмерный вектор признаков, который отражает основные характеристики лица.
  2. **Методы на основе глубокого обучения**: Модели глубокого обучения, в частности, конволюционные нейронные сети (КНС), произвели революцию в распознавании лиц. CNN могут изучать иерархические представления непосредственно из необработанных значений пикселей, что позволяет создавать более точные и дискриминационные представления лиц. Такие модели, как FaceNet и VGGFace, используют глубокие архитектуры CNN для изучения вкраплений лиц, которые являются компактными векторными представлениями лиц.

**Верификация/идентификация лица:**

После получения представлений лица алгоритм выполняет задачи верификации или идентификации лица.

* 1. Верификация лица: При проверке лица алгоритм определяет, принадлежат ли два представления лица одному и тому же человеку или нет. Он вычисляет метрику сходства или расстояния между двумя вкраплениями лица и сравнивает ее с пороговым значением для принятия решения.
  2. Идентификация лица: При идентификации лица алгоритм сравнивает данное представление лица с базой данных известных представлений лица, чтобы определить личность человека. Обычно это включает вычисление сходства или расстояния между запрашиваемым лицом и лицами из базы данных и выбор наиболее близкого соответствия.

1.1.2 История алгоритмов распознавания лиц. От ранних экспериментов до разработки сложных моделей глубокого обучения, алгоритмы распознавания лиц прошли долгий путь в своей способности идентифицировать и проверять людей на основе их черт лица. Давайте рассмотрим основные вехи в истории алгоритмов распознавания лиц:

**Ранние подходы (1960-1990-е годы):**

В 1960-х годах Вуди Бледсоу, Хелен Чан Вольф и Чарльз Биссон разработали систему распознавания лиц, которая использовала компьютер для анализа и сравнения черт лица. Эта система стала первой значительной попыткой автоматизировать распознавание лиц. Однако из-за ограниченности вычислительных мощностей и доступных наборов данных прогресс в этот период был медленным.

**Собственные лица (1990-е годы):**

В 1990-х годах Сирович и Кирби представили концепцию собственных лиц - метод, основанный на анализе главных компонент (PCA). Eigenfaces представляет лица как линейную комбинацию собственных векторов, что позволяет уменьшить размерность и эффективно распознавать лица. Этот подход проложил путь для будущих разработок алгоритмов распознавания лиц.

**Анализ локальных признаков (2000-е годы):**

В начале 2000-х годов алгоритмы распознавания лиц перешли к локальному анализу признаков. Новаторская работа Пола Виолы и Майкла Джонса представила алгоритм Виолы-Джонса, который использовал Хаар-подобные признаки и каскадные классификаторы для обнаружения лиц на изображениях. Этот алгоритм заложил основу для обнаружения лиц в реальном времени в различных приложениях.

**Статистические модели (2000-е годы):**

Еще один значительный прогресс в распознавании лиц произошел с внедрением статистических моделей. Одним из популярных подходов была модель активного внешнего вида (AAM), предложенная Кутсом, Эдвардсом и Тейлором. AAM моделирует вариации формы и текстуры лица, обеспечивая надежное распознавание и анализ выражения лица.

**Революция глубокого обучения (2010-е годы):**

Появление глубокого обучения произвело революцию в области распознавания лиц. Глубокие нейронные сети, в частности конволюционные нейронные сети (CNN), продемонстрировали исключительную производительность в различных задачах компьютерного зрения, включая распознавание лиц. Система DeepFace, разработанная командой Facebook AI Research (FAIR), достигла выдающейся точности благодаря обучению глубокой нейронной сети на крупномасштабном наборе данных.

**DeepFace и FaceNet (2014-2015 гг.):**

DeepFace, представленная командой FAIR в 2014 году, использовала глубокую CNN для отображения изображений лица в высокоразмерном пространстве признаков, что позволило обеспечить точную проверку лица. В 2015 году FaceNet от Google расширил границы распознавания лиц, внедрив триплетную функцию потерь, которая позволяет генерировать высокодискриминантные вкрапления лиц.

**Одномоментное обучение и генеративные модели (с 2016 года по настоящее время):**

Последние достижения в алгоритмах распознавания лиц были сосредоточены на одномоментном обучении и генеративных моделях. Одним из примеров является разработка генеративных адверсариальных сетей (GAN), которые применяются для создания реалистичных и разнообразных изображений лиц. Кроме того, такие модели, как Face ID в айфонах Apple, используют методы глубокого обучения для создания 3D-карт лиц для безопасной и удобной аутентификации.

LBPH Алгоритм (1996)

Алгоритм Local Binary Patterns Histogram (LBPH) - это широко используемый метод распознавания лиц и анализа текстуры. Впервые он был представлен Тимо Ахоненом, Абденуром Хадидом и Матти Пиетикайненом в 2002 году. Алгоритм LBPH известен своей простотой и эффективностью, особенно в сценариях, когда доступный набор данных невелик.

Алгоритм LBPH работает с полутоновыми изображениями, улавливая локальную структуру изображения. Он описывает текстуру данного пикселя, сравнивая его с соседними и кодируя результат в двоичный шаблон. Затем эти двоичные шаблоны объединяются в гистограмму, которая отражает общую текстуру изображения.

Рассмотрим шаги алгоритма:

1. **Параметры: LBPH использует 4 параметра:**

Радиус: радиус используется для построения круговой локальной двоичной схемы и представляет собой радиус вокруг центрального пикселя. Обычно он равен 1.

Соседи: количество точек выборки для построения кругового локального бинарного шаблона. Помните: чем больше точек выборки, тем выше вычислительные затраты. Обычно задается значение 8.

Grid X: количество ячеек в горизонтальном направлении. Чем больше ячеек, тем мельче сетка, тем выше размерность результирующего вектора признаков. Обычно задается значение 8.

Grid Y: количество ячеек в вертикальном направлении. Чем больше ячеек, тем мельче сетка, тем выше размерность результирующего вектора признаков. Обычно задается равным 8.

2. **Обучение алгоритма:** Сначала нам нужно обучить алгоритм. Для этого нам нужно использовать набор данных с изображениями лиц людей, которых мы хотим распознать. Нам также необходимо задать идентификатор (это может быть номер или имя человека) для каждого изображения, чтобы алгоритм использовал эту информацию для распознавания входного изображения и выдачи выходного. Изображения одного и того же человека должны иметь одинаковый идентификатор.

3. **Применение операции LBP:** Первым вычислительным шагом LBPH является создание промежуточного изображения, которое лучше описывает исходное изображение, выделяя черты лица. Для этого алгоритм использует концепцию скользящего окна, основанную на параметрах радиуса и соседей.

Вот пошаговое описание алгоритма LBPH:

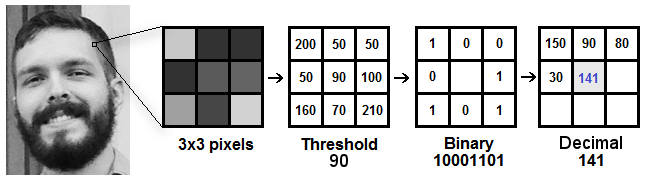


Рис 1.1 *(LBPH алгоритм)*

**Сегментация изображения:**

Входное изображение делится на более мелкие области, называемые ячейками. Каждая ячейка обычно содержит квадратную или круглую область пикселей.

**Извлечение характеристик:**

Для каждого пикселя в ячейке LBPH сравнивает значение его интенсивности с окружающими соседями. Обычно выбирается 8 соседей, образующих круг вокруг центрального пикселя. В результате сравнения интенсивности получается двоичный код: 1, если интенсивность соседа равна или больше, чем у центрального пикселя, и 0 в противном случае.

**Вычисление гистограммы:**

Двоичные коды, полученные от всех пикселей в ячейке, объединяются в двоичное число. Затем это двоичное число преобразуется в десятичное представление. Десятичные значения, полученные для всех ячеек, используются для построения гистограммы, где каждый бином представляет уникальное десятичное значение.

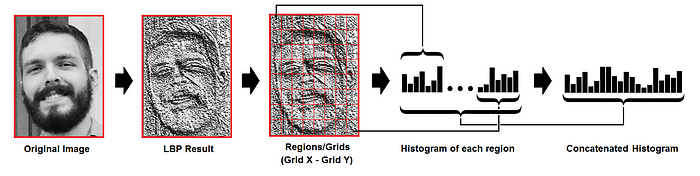


Рис 1.2*(LBPH алгоритм)*

**Представление изображения:**

Гистограммы, полученные из всех ячеек, объединяются для формирования окончательного вектора признаков, который представляет текстуру всего изображения.

**Классификация:**

Полученный вектор признаков может быть использован для задач классификации или распознавания с помощью различных алгоритмов машинного обучения, таких как k-nearest neighbors (KNN) или support vector machines (SVM).

* 1. Теория и история алгоритмов отслеживания объектов

1.2.1 Алгоритмы обнаружения объектов направлены на идентификацию и определение местоположения интересующих объектов в изображениях или видеокадрах. Общая теория обнаружения объектов включает в себя два ключевых компонента: генерацию предложений регионов и классификацию этих предложений для определения наличия и местоположения объектов.

Ниже приводится обзор общей теории алгоритмов обнаружения объектов:

**Генерация предложений регионов:**

Первым шагом в обнаружении объектов является генерация предложений регионов, которые представляют собой потенциальные регионы-кандидаты, которые могут содержать объекты. Для генерации предложений регионов могут использоваться различные методы, такие как скользящие окна, селективный поиск или методы на основе якорей.

**Скользящие окна:**

Методы скользящего окна предполагают сканирование изображения или различных масштабов изображения с помощью окна фиксированного размера. В каждой позиции окна извлекаются характеристики, и классификатор определяет, содержит ли регион объект или нет. Этот подход требует больших вычислительных затрат, но может быть эффективен, когда объекты имеют фиксированное соотношение сторон.

**Селективный поиск:**

Селективный поиск - это алгоритм предложения региона, который сегментирует изображение на основе различных признаков изображения, таких как цвет, текстура и форма. Он генерирует иерархию предложений регионов, которые постепенно становятся более тонкими, с целью захвата объектов в различных масштабах и ориентациях

**Методы, основанные на якорях:**

Якорные методы, широко используемые в детекторах на основе глубокого обучения, таких как Faster R-CNN и YOLO, используют набор предопределенных якорных ящиков с различными соотношениями сторон и масштабами. Эти якорные ящики размещаются в различных местах изображения, и модель предсказывает, присутствует ли объект в пределах каждого якорного ящика.

**Классификация и локализация:**

После создания предложений регионов следующим шагом является классификация и локализация объектов в этих предложениях. Это включает в себя определение класса объекта (например, "человек", "автомобиль", "кошка") и точную локализацию его положения с помощью ограничивающих ячеек.

**Классификация**: Для классификации объектов могут использоваться различные модели машинного обучения или глубокого обучения. Эти модели обучаются на помеченных наборах данных, где они учатся распознавать различные категории объектов на основе извлеченных признаков из предложений регионов.

**Локализация**: Локализация объекта включает в себя оценку координат ограничительной рамки, которая плотно охватывает объект в пределах предложения региона. Модель учится предсказывать координаты ограничительной рамки относительно местоположения предложения.

**Постобработка и уточнение:**

После классификации и локализации для уточнения результатов обнаружения объектов можно применить шаги постобработки. Эти шаги часто включают в себя немаксимальное подавление (NMS) для устранения избыточных или перекрывающихся ограничительных рамок и применение порогов уверенности для отсеивания обнаружений ниже определенного балла уверенности.

* + 1. История алгоритмов обнаружения лиц (объектов).

Рассмотрим основные вехи в истории алгоритмов обнаружения лиц:

**Подходы, основанные на правилах:**

На заре компьютерного зрения для обнаружения лиц в основном использовались методы, основанные на правилах. Эти подходы опирались на правила или эвристику, разработанные вручную, для определения таких черт лица, как глаза, нос и рот. Однако из-за сложности и изменчивости лиц эти методы были ограничены в своей точности и надежности.

Алгоритм Виола-Джонса (2001):

Алгоритм Виолы-Джонса, представленный Полом Виолой и Майклом Джонсом в 2001 году, произвел революцию в распознавании лиц. В нем использовалась концепция Хаар-подобных характеристик и каскадных классификаторов для достижения производительности в режиме реального времени. Алгоритм использовал обученный каскад простых классификаторов Хаара для быстрого удаления нелицевых областей изображения, а затем более сложные классификаторы для точного определения лиц.

**Классификаторы Хаара** - классификаторы, которые использовались в первом детекторе лиц в реальном времени. Классификатор Хаара, или каскадный классификатор Хаара, - это программа машинного обучения для обнаружения объектов, которая идентифицирует объекты на изображениях и видео.

Подробное описание классификаторов Хаара можно найти в статье Пола Виолы и Майкла Джонса "Rapid Object Detection using a Boosted Cascade of Simple Features.

**Создание каскадного классификатора Хаара**

1. Алгоритм можно объяснить в четыре этапа:
2. Вычисление характеристик Хаара
3. Создание интегральных образов
4. Использование Adaboost
5. Реализация каскадных классификаторов

Важно помнить, что этот алгоритм требует большого количества положительных изображений лиц и отрицательных изображений нелиц для обучения классификатора, как и другие модели машинного обучения.

**Вычисление характеристик Хаара**

Первым шагом является сбор признаков Хаара. Признак Хаара - это, по сути, вычисления, которые выполняются для смежных прямоугольных областей в определенном месте окна обнаружения. Вычисления включают суммирование интенсивностей пикселей в каждой области и вычисление разницы между этими суммами. Ниже приведены примеры признаков Хаара.

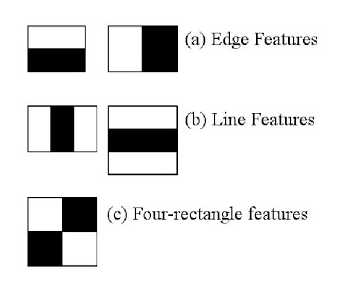


Рис. 1.3 (*признаки Хаара*)

Эти особенности может быть трудно определить для большого изображения. Именно здесь в игру вступают интегральные изображения, поскольку при использовании интегрального изображения количество операций сокращается.

**Создание интегральных изображений**

Интегральные изображения, по сути, ускоряют вычисление этих характеристик Хаара. Вместо того чтобы вычислять каждый пиксель, вместо этого создаются подпрямоугольники и создаются ссылки на массивы для каждого из этих подпрямоугольников. Затем они используются для вычисления характеристик Хаара.

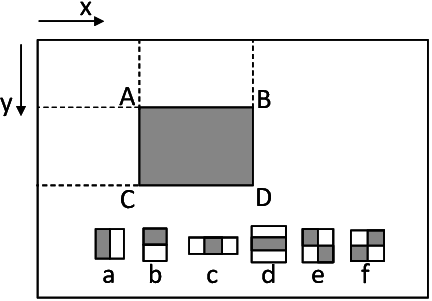


Рис. 1.4 (*интегральные изображения*)

Важно отметить, что почти все признаки Хаара не будут иметь значения при обнаружении объекта, поскольку важны только признаки объекта. Однако как определить лучшие признаки, которые представляют объект, из сотен тысяч признаков Хаара? Именно здесь в игру вступает Adaboost.

**Обучение с помощью Adaboost**

Adaboost, по сути, выбирает лучшие признаки и обучает классификаторы использовать их. Он использует комбинацию "слабых классификаторов" для создания "сильного классификатора", который алгоритм может использовать для обнаружения объектов.

Слабые классификаторы создаются путем перемещения окна по входному изображению и вычисления характеристик Хаара для каждого подраздела изображения. Эта разница сравнивается с выученным порогом, который отделяет необъекты от объектов. Поскольку это "слабые классификаторы", для формирования сильного классификатора требуется большое количество признаков Хаара.

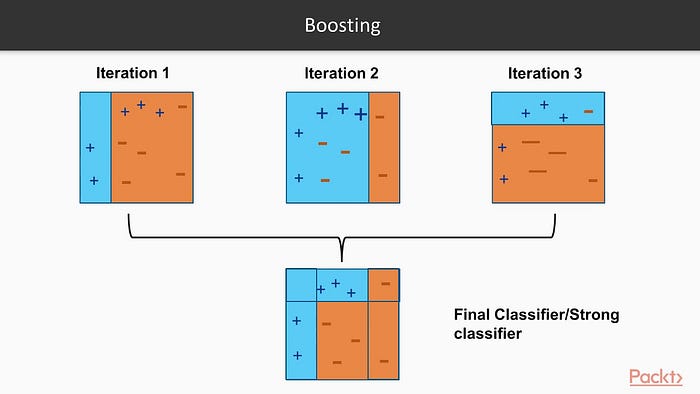


Рис. 1.5(*adaboost*)

На последнем этапе эти слабые классификаторы объединяются в сильный классификатор с помощью каскадных классификаторов.

**Реализация каскадных классификаторов**

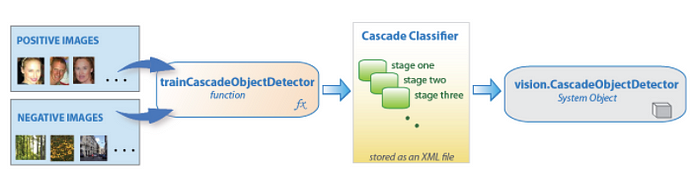


Рис. 1.6(*реализация*)

Каскадный классификатор состоит из ряда этапов, где каждый этап представляет собой набор слабых обучаемых. Слабые обучаемые обучаются с помощью бустинга, что позволяет получить высокоточный классификатор на основе среднего предсказания всех слабых обучаемых.

На основе этого предсказания классификатор принимает решение либо указать, что объект был найден (положительный результат), либо перейти к следующей области (отрицательный результат). Этапы разработаны таким образом, чтобы как можно быстрее отбрасывать отрицательные выборки, поскольку большинство окон не содержат ничего интересного.

Важно добиться максимально низкого показателя ложных отрицаний, поскольку классификация объекта как не-объекта сильно ухудшит работу вашего алгоритма обнаружения объектов.

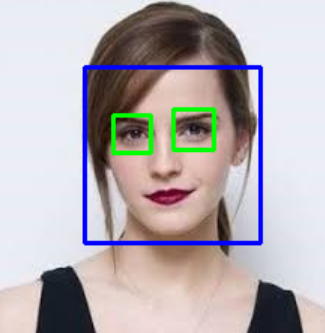


Рис. 1.6 (*результат*)

**Гистограмма ориентированных градиентов (HOG) (2005):**

Navneet Dalal и Bill Triggs представили метод Histogram of Oriented Gradients (HOG) в 2005 году. Этот метод вычисляет градиенты в изображении и представляет локальную информацию о форме и текстуре. Характеристики HOG затем использовались с алгоритмами машинного обучения, такими как машины опорных векторов (SVM), для обнаружения лиц.

**Революция глубокого обучения (2012 год - настоящее время):**

Появление моделей глубокого обучения оказало глубокое влияние на обнаружение лиц. Глубокие нейронные сети, в частности, конволюционные нейронные сети (CNN), продемонстрировали исключительную производительность в различных задачах компьютерного зрения, включая распознавание лиц. Модели глубокого обучения могут автоматически изучать иерархические представления лиц, что делает их очень эффективными в обнаружении сложных моделей.

**Конволюционные нейронные сети на основе регионов (R-CNN) (2014):**

R-CNN, предложенная Россом Гиршиком и др. в 2014 году, ознаменовала значительный прогресс в обнаружении объектов, включая обнаружение лиц. R-CNN разделила задачу обнаружения на два этапа: генерация предложения региона и тонкая настройка с помощью CNN. Этот метод достиг впечатляющей точности, но был вычислительно дорогим.

**Single Shot MultiBox Detector (SSD) (2016):**

SSD, представленный Вэй Лю и др. в 2016 году, был нацелен на повышение скорости обнаружения объектов. Он объединил преимущества методов предложения областей с производительностью в реальном времени, используя набор ограничительных рамок по умолчанию и применяя несколько конволюционных слоев для предсказания классов объектов и уточнения координат ограничительных рамок.

**You Only Look Once (YOLO) (2016):**

YOLO, предложенная Джозефом Редмоном и др. в 2016 году, представила систему обнаружения объектов в реальном времени, которая достигла впечатляющей скорости и точности. YOLO рассматривает обнаружение объектов как проблему регрессии и делит входное изображение на сетку, каждая ячейка которой отвечает за предсказание объектов и их соответствующих ограничительных рамок.

**EfficientDet (2019):**

EfficientDet, разработанный Mingxing Tan и Quoc V. Le в 2019 году, расширил границы обнаружения объектов, представив эффективные и масштабируемые модели. В ней использовалась комбинированная техника масштабирования, которая оптимизировала глубину, ширину и разрешение сетевой архитектуры для достижения самой современной производительности при различных ограничениях ресурсов.

* 1. Сравнительный анализ методов

1.3.1Сравнительный анализ методов обнаружения объектов

**Алгоритм Виола-Джонса:**

Плюсы:

Работа в режиме реального времени: Алгоритм Виола-Джонса был специально разработан для обнаружения лиц в режиме реального времени, что делает его подходящим для приложений, требующих высокой скорости обработки.

Устойчивость к изменениям: Алгоритм включает в себя каскад классификаторов, которые постепенно отфильтровывают нелицевые области, что помогает повысить устойчивость к изменениям в освещении, позе и выражении лица.

Относительно низкие вычислительные требования: По сравнению с более сложными моделями глубокого обучения, алгоритм Виолы-Джонса является вычислительно эффективным и может работать на устройствах с ограниченными ресурсами.

Минусы:

Ограниченная точность: Хотя алгоритм Виолы-Джонса хорошо работает во многих сценариях, он может испытывать трудности при более сложных изменениях лица или окклюзиях из-за того, что он опирается на простые признаки, подобные Хаару.

Отсутствие тонкой локализации: Алгоритм обеспечивает определение границ лица, но не обеспечивает детальной локализации ориентиров лица, что ограничивает его применение в задачах, требующих точного анализа лица.

**Гистограмма ориентированных градиентов (HOG):**

Плюсы:

Более высокая точность, чем у Виола-Джонса: Методы на основе HOG могут захватывать более сложную информацию о форме и текстуре, что делает их более точными в обнаружении лиц с различными вариациями.

Подходит для умеренных требований к ресурсам: Хотя методы на основе HOG не так эффективны, как Виола-Джонса, они все же могут обеспечить достаточно высокую производительность на современном оборудовании.

Минусы:

Медленнее, чем Виола-Джонс: Методы на основе HOG в целом медленнее, чем Виола-Джонса, особенно при использовании в сочетании с традиционными классификаторами машинного обучения.

Ограниченная устойчивость к экстремальным вариациям: Методы на основе HOG могут не справляться с экстремальными изменениями позы, не фронтальными лицами или окклюзивными лицами из-за того, что они полагаются на информацию о локальном градиенте.

**Подходы на основе глубокого обучения (например, R-CNN, YOLO):**

Плюсы:

Современная точность: Модели глубокого обучения достигли значительной точности в распознавании лиц, особенно при обучении на крупномасштабных наборах данных

Лучшая обобщенность: Модели глубокого обучения могут изучать сложные характеристики и адаптироваться к широкому диапазону вариаций внешнего вида лица, позы и условий освещения.

Минусы:

Более высокие вычислительные требования: Модели глубокого обучения требуют больших вычислительных затрат и обычно нуждаются в более мощном оборудовании или специализированных ускорителях.

Данные для обучения и вычислительные ресурсы: Модели глубокого обучения часто требуют больших объемов аннотированных учебных данных и значительных вычислительных ресурсов для обучения, которые могут быть доступны не во всех сценариях.

1.2.2 Сравнительный анализ методов распознавания лица

**LBPH (Local Binary Patterns Histograms):**

Плюсы:

Простота и вычислительная эффективность: LBPH - это относительно простой и эффективный с вычислительной точки зрения метод распознавания лиц. Он работает путем извлечения локальных бинарных паттернов из изображения лица и построения гистограммного представления. Такая простота делает его пригодным для использования в условиях ограниченных ресурсов.

Устойчивость к изменениям: LBPH известен своей устойчивостью к изменениям выражения лица, освещенности и незначительным изменениям позы. Он захватывает текстурные паттерны в локальных областях, что делает его более устойчивым к определенным изменениям.

Относительно небольшой набор данных для обучения: LBPH часто может достичь приемлемой производительности при меньшем количестве обучающих образцов по сравнению с более сложными моделями глубокого обучения.

Минусы:

Ограниченная дискриминационная способность: LBPH может не улавливать высокодискриминантные признаки, которые различают похожих людей. Она может не справляться со сложными вариациями, большими изменениями позы или окклюзиями, поскольку в основном полагается на локальные текстурные паттерны.

Чувствительность к шуму и качеству изображения: LBPH чувствителен к шуму и качеству изображения, поскольку работает на уровне пикселей. Плохое качество изображения или шум могут повлиять на точность локальных бинарных паттернов и, соответственно, на эффективность распознавания.

**Подходы на основе глубокого обучения (например, FaceNet, VGGFace):**

Плюсы:

Современная точность: Модели распознавания лиц на основе глубокого обучения, такие как FaceNet и VGGFace, достигли выдающейся точности и могут изучать высокодискриминативные представления лиц.

Лучшая обобщенность: Модели глубокого обучения способны улавливать сложные и абстрактные черты лица, что делает их более эффективными в работе с вариациями и позволяет достичь высокой эффективности распознавания.

Сквозное обучение: Модели глубокого обучения могут изучать представления признаков непосредственно из необработанных значений пикселей, что устраняет необходимость в ручной разработке признаков.

Минусы:

Необходимость в большом наборе обучающих данных: Для достижения оптимальной производительности моделям глубокого обучения часто требуются большие аннотированные наборы данных. Получение и маркировка таких наборов данных может занять много времени и ресурсов.

Повышенные вычислительные требования: Модели глубокого обучения требуют больших вычислительных затрат и обычно требуют более мощного оборудования или специализированных ускорителей для достижения производительности в реальном времени.

Чувствительность к смещениям в наборе данных: Модели глубокого обучения могут быть чувствительны к предубеждениям, присутствующим в обучающих данных, что потенциально может привести к проблемам справедливости и предубеждения при распознавании лиц.

1.4 Выбранные методы

В результате анализа и сравнения методов обнаружения и распознавания лиц была выбрана связка из алгоритма Виола-Джонса(Хаар Касады или же Haarcascade) и LBPH. Это обусловлено следующим:

Надежное распознавание лиц:

Каскады Хаара эффективны при обнаружении лиц в изображении или видео кадре. Они используют обученный классификатор для определения областей, которые могут содержать лица, на основе Хаароподобных характеристик. Включив каскады Хаара в систему, вы сможете обеспечить точное и надежное обнаружение лиц даже при различных условиях освещения, позах и выражениях лица.

Производительность в реальном времени:

Каскады Хаара известны своей вычислительной эффективностью, что позволяет распознавать лица в режиме реального времени. Это делает их подходящими для приложений, требующих быстрой и оперативной обработки, таких как системы видеонаблюдения или мониторинга в реальном времени.

Локальное представление лица на основе текстуры:

LBPH - это простой, но эффективный метод представления лица. Он извлекает локальные бинарные паттерны из областей лица, создавая гистограммное представление, которое отражает текстурные паттерны в каждой области. Это представление кодирует дискриминационную информацию о лице, что делает его пригодным для задач распознавания.

Устойчивость к некоторым изменениям:

LBPH известен своей устойчивостью к изменениям выражения лица, освещенности и незначительным изменениям позы. Он фокусируется на захвате локальных текстурных паттернов, которые, как правило, более устойчивы к изменениям по сравнению с другими признаками, такими как форма или цвет. Эта устойчивость повышает способность системы распознавать людей в различных условиях.

Небольшой набор обучающих данных:

LBPH часто может достичь приемлемой производительности даже при относительно небольшом количестве обучающих образцов. Это делает ее преимуществом в сценариях, где сбор и маркировка крупномасштабного набора данных может быть сложной задачей или отнимать много времени.

Вычислительная эффективность:

Каскады Хаара и LBPH являются вычислительно эффективными методами. Они не требуют больших вычислительных ресурсов, что делает их пригодными для применения на устройствах с ограниченными ресурсами или в системах с ограниченной вычислительной мощностью.

Вывод: Комбинируя каскады Хаара для обнаружения лиц и LBPH для представления и распознавания лиц, можно получить преимущества надежного и эффективного обнаружения лиц, точного распознавания лиц в режиме реального времени и компактного представления, передающего важную информацию о текстуре лица. Эта комбинация особенно полезна в приложениях, где требуется быстрое и надежное распознавание лиц, например, в системах контроля доступа, отслеживания посещаемости или идентификации личности. Что и является целью разработки

Глава 2 Анализ современных методов и инструментов

* 1. Анализ архитектуры ПО для распознавания лиц

Архитектура программы распознавания лиц может варьироваться в зависимости от конкретной реализации и требований. Однако вот общая архитектура, которая обычно используется:

**Ввод данных**: Программное обеспечение получает входные данные, содержащие изображения или видеокадры, которые необходимо обработать для распознавания лиц. Эти данные могут поступать из различных источников, таких как камеры, видеофайлы или наборы данных изображений.

**Обнаружение лица:** Первым шагом в конвейере распознавания лиц является обнаружение лица. Этот модуль идентифицирует и локализует присутствие лиц во входных данных. Для обнаружения лиц могут использоваться такие методы, как каскады Хаара, HOG или модели на основе глубокого обучения. Результатом работы этого модуля является расположение и размер обнаруженных лиц.

**Выравнивание и предварительная обработка лица**: После обнаружения лиц может быть использован модуль выравнивания лиц для нормализации изображений лиц путем их выравнивания на основе лицевых ориентиров или других геометрических преобразований. Этот шаг помогает уменьшить отклонения, связанные с позой головы, и повышает точность последующей обработки. Кроме того, для повышения качества и согласованности изображений лица могут применяться такие этапы предварительной обработки, как нормализация, изменение размера и выравнивание гистограммы.

**Извлечение признаков:** На этапе извлечения признаков создается представление для каждой области лица, чтобы передать ее уникальные характеристики. Для извлечения признаков могут использоваться различные алгоритмы, включая традиционные методы, такие как LBPH, Eigenfaces, или методы глубокого обучения с использованием сверточных нейронных сетей (CNN). Результатом этого этапа является вектор признаков или вставка, в которой закодированы отличительные атрибуты лица.

**База данных лиц или хранилище вкраплений**: Сгенерированные вкрапления или векторы признаков лица обычно хранятся в базе данных лиц или в хранилище вкраплений. Эта база данных служит в качестве эталона для сравнения на этапе распознавания. Каждое хранимое вкрапление связано с идентификационной меткой или метаданными.

**Распознавание лица**: На этапе распознавания система сравнивает извлеченные черты лица из входного изображения или видеокадра с хранящимися в базе данных вкраплениями лица. Для сравнения и классификации могут использоваться различные алгоритмы, такие как евклидово расстояние, косинусное сходство или классификаторы SVM (Support Vector Machine). Система определяет идентичность входного лица путем поиска наиболее похожего или ближайшего соседа в базе данных.

**Вывод и принятие решений**: Последний этап включает в себя принятие решений на основе результатов распознавания лиц. Система может выводить распознанную личность, баллы доверия или дополнительные метаданные, связанные с идентифицированными лицами. В зависимости от приложения могут быть инициированы дальнейшие действия, такие как контроль доступа, отслеживание посещаемости или персонализация пользовательского опыта.

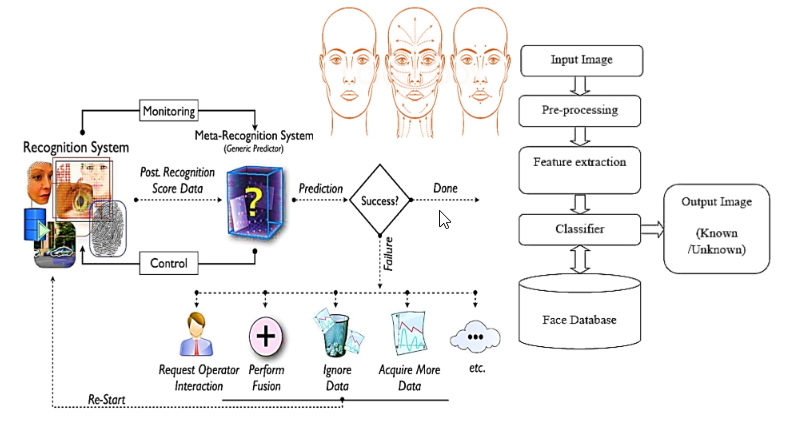


Рис 2.1 (*Архитектура ПО*)

2.2 Библиотеки

* + 1. Обзор библиотек реализующих методы обнаружения объектов:

**OpenCV:**

OpenCV (Open Source Computer Vision Library) - это широко используемая библиотека компьютерного зрения, которая предоставляет различные алгоритмы и функции, включая обнаружение объектов. Она предлагает предварительно обученные модели для обнаружения объектов, такие как каскады Хаара и детекторы на основе HOG. OpenCV также обеспечивает поддержку обнаружения объектов на основе глубокого обучения с использованием таких фреймворков, как TensorFlow и PyTorch.

**TensorFlow Object Detection API:**

TensorFlow Object Detection API - это мощный фреймворк, построенный поверх TensorFlow, который упрощает реализацию моделей обнаружения объектов. Он предлагает коллекцию предварительно обученных моделей (например, SSD, Faster R-CNN) и предоставляет инструменты для обучения и оценки. API поддерживает как традиционные методы машинного обучения, так и подходы на основе глубокого обучения.

**PyTorch:**

PyTorch - это популярный фреймворк глубокого обучения, который обеспечивает эффективную реализацию различных задач компьютерного зрения, включая обнаружение объектов. Он предлагает предварительно обученные модели, такие как Faster R-CNN, Mask R-CNN и YOLO, а также обеспечивает гибкость для создания собственных архитектур обнаружения объектов. PyTorch позволяет легко интегрироваться с другими библиотеками и инструментами компьютерного зрения.

**Detectron2**:

Detectron2 - это высокопроизводительная модульная система обнаружения объектов, построенная на базе PyTorch. Он предоставляет коллекцию самых современных моделей обнаружения объектов и позволяет настраивать и дорабатывать их. Detectron2 предлагает богатый набор функциональных возможностей для обучения, вывода и оценки, что делает его подходящим для исследований и развертывания на производственном уровне.

**YOLO (You Only Look Once):**

YOLO - это алгоритм обнаружения объектов, известный своей производительностью в режиме реального времени. Существует несколько реализаций YOLO, таких как Darknet, Darknet PyTorch и YOLOv5, которые предоставляют предварительно обученные модели и код для обнаружения объектов. Эти реализации часто предлагают баланс между скоростью и точностью

**MXNet:**

MXNet - это гибкая система глубокого обучения, которая поддерживает задачи обнаружения объектов. Он предоставляет предварительно обученные модели, такие как SSD (Single Shot MultiBox Detector) и Faster R-CNN, а также предлагает инструменты для обучения и вывода. MXNet известен своим эффективным управлением памятью и часто используется в производственных системах.

* + 1. Обзор библиотек реализующих методы распознавания лица:

**OpenCV:**

OpenCV (Open Source Computer Vision Library) - это широко используемая библиотека компьютерного зрения, которая предлагает различные возможности распознавания лиц. Она предоставляет предварительно обученные модели для распознавания лиц, такие как каскады Хаара и модели на основе глубокого обучения. OpenCV также предлагает функции для распознавания лиц с помощью таких методов, как LBPH (Local Binary Patterns Histograms) и Eigenfaces.

**dlib:**

dlib - это библиотека C++, известная своими надежными возможностями обнаружения и распознавания лиц. Она предлагает предварительно обученные модели для обнаружения лиц и оценки лицевых ориентиров, а также инструменты для распознавания лиц с помощью таких методов, как гистограмма ориентированных градиентов (HOG) и глубокое метрическое обучение.

**FaceNet:**

FaceNet - это модель распознавания лиц на основе глубокого обучения, разработанная компанией Google. TensorFlow предоставляет реализацию FaceNet, включая предварительно обученные модели, для задач распознавания лиц. Она предлагает точные и дискриминационные вкрапления лиц, которые могут быть использованы для верификации или идентификации лиц.

**VGGFace:**

VGGFace - это модель глубокого обучения, разработанная Группой визуальной геометрии (VGG) Оксфордского университета. Она предоставляет предварительно обученные модели для распознавания лиц с помощью сверточных нейронных сетей (CNN). VGGFace достигает высокой точности и может быть использован для задач верификации или идентификации лица.

**PyTorch и TensorFlow:**

PyTorch и TensorFlow - популярные платформы глубокого обучения, которые предоставляют инструменты для распознавания лиц. Они предлагают предварительно обученные модели, такие как FaceNet, VGGFace и различные архитектуры CNN, для задач распознавания лиц. Эти фреймворки обеспечивают гибкость при построении собственных моделей распознавания лиц и предлагают эффективные вычисления на GPU.

**ArcFace:**

ArcFace - это модель глубокого обучения, специально разработанная для задач распознавания лиц. Она обеспечивает мощное и дискриминационное встраивание лица путем введения функции потерь на основе угловой маржи. Реализации ArcFace доступны в популярных фреймворках глубокого обучения, таких как TensorFlow и PyTorch.

2.3 Сравнение библиотек и примеры имплементации

2.3.1 Сравнение библиотек для отслеживания объектов

**OpenCV:**

Простота использования: OpenCV известен своей простотой и удобством использования. Он предоставляет удобный API и исчерпывающую документацию, что делает его доступным для разработчиков разного уровня подготовки.

Скорость: OpenCV разработан как эффективный и оптимизированный для производительности инструмент. Он включает в себя высоко оптимизированные алгоритмы, реализованные на C++, что обеспечивает быстрое выполнение задач по обработке изображений и видео.

Другие факторы: OpenCV имеет широкий спектр функциональных возможностей, помимо обнаружения объектов, включая манипулирование изображениями, извлечение признаков и калибровку камеры. Он поддерживает различные языки программирования, включая C++, Python и Java. OpenCV также имеет большое сообщество, что обеспечивает регулярное обновление, исправление ошибок и поддержку.

**TensorFlow Object Detection API:**

Простота использования: TensorFlow Object Detection API предлагает высокоуровневый API, который упрощает процесс реализации моделей обнаружения объектов. Он предоставляет предварительно обученные модели и инструменты для обучения и оценки. Однако API может иметь более сложную кривую обучения по сравнению с OpenCV, особенно если вы новичок в TensorFlow.

Скорость: Скорость работы TensorFlow Object Detection API зависит от архитектуры модели и аппаратных настроек. TensorFlow поддерживает аппаратное ускорение с помощью GPU и TPU, что может значительно повысить скорость выводов.

Другие факторы: TensorFlow - это широко распространенный фреймворк глубокого обучения с обширным сообществом и экосистемой. Он предлагает обширную документацию, учебники и ресурсы. API для обнаружения объектов предоставляет широкий спектр предварительно обученных моделей, включая такие популярные, как Faster R-CNN и SSD, что делает его подходящим для различных задач обнаружения объектов.

**PyTorch:**

Простота использования: PyTorch известен своим интуитивно понятным и питоническим синтаксисом, что делает его относительно простым в изучении и использовании. Он предоставляет динамический вычислительный граф, что обеспечивает большую гибкость при проектировании и отладке моделей. Однако, по сравнению с OpenCV, PyTorch может иметь более сложную кривую обучения, если вы новичок в использовании фреймворков глубокого обучения.

Скорость: Скорость PyTorch зависит от реализации модели и аппаратной настройки. Он поддерживает GPU-ускорение, что может повысить скорость вычислений. Кроме того, PyTorch постоянно оптимизирует свою производительность, и в последних версиях достигнуты значительные успехи в скорости и эффективности.

Другие факторы: PyTorch завоевал популярность благодаря своей гибкости, позволяющей исследователям и разработчикам легко экспериментировать с новыми идеями. Он предлагает богатую экосистему с различными предварительно обученными моделями и самыми современными архитектурами. PyTorch также имеет активное сообщество и обширную документацию.

**Detectron2:**

Простота использования: Detectron2 основан на PyTorch и предоставляет API более высокого уровня для задач обнаружения объектов. По сравнению со своим предшественником, Detectron, он имеет более упрощенную и модульную конструкцию. API упрощает процесс построения и обучения моделей обнаружения объектов, но все же может потребовать некоторого знакомства с PyTorch.

Скорость: Detectron2 использует преимущества оптимизации PyTorch и обеспечивает эффективное обучение и выводы. Он поддерживает GPU-ускорение, что может значительно ускорить время обработки.

Другие факторы: Detectron2 широко используется в исследовательском сообществе благодаря своей модульной конструкции и поддержке передовых алгоритмов обнаружения объектов. Он предоставляет богатый набор предварительно обученных моделей, а его модульная структура позволяет легко настраивать и экспериментировать. Detectron2 также имеет активное сообщество и постоянно развивается.

**YOLO (You Only Look Once):**

Простота использования: Реализации YOLO могут различаться по простоте использования в зависимости от конкретной библиотеки или фреймворка, который вы выберете. В целом, YOLO имеет простой конвейер, что делает его относительно легким для понимания и внедрения.

Скорость: YOLO славится своей производительностью в режиме реального времени. Она достигается за счет того, что обнаружение объектов рассматривается как единая регрессионная задача, что приводит к быстрым вычислениям. Реализации на основе YOLO сильно оптимизированы для скорости.

Другие факторы: YOLO часто предпочитают использовать в приложениях, требующих обнаружения объектов в реальном времени, например, в системах видеонаблюдения или автономного вождения. Он обеспечивает хороший баланс между точностью и скоростью. Реализации YOLO доступны в различных фреймворках, включая Darknet, фреймворки на основе Darknet, такие как Darkflow, и варианты, реализованные в TensorFlow и PyTorch.

**MXNet:**

Простота использования: MXNet предоставляет простой в использовании высокоуровневый API, что делает его доступным для новичков. Он имеет удобный синтаксис, схожий с другими фреймворками глубокого обучения, такими как TensorFlow и PyTorch.

Скорость: MXNet разработан для повышения скорости и эффективности использования памяти. Он использует передовые методы оптимизации для обеспечения быстрого времени выполнения, что делает его подходящим как для задач обучения, так и для задач вывода.

Другие факторы: MXNet известен своей масштабируемостью и возможностями распределенных вычислений. Он поддерживает GPU-ускорение и обеспечивает беспрепятственное развертывание на различных устройствах, включая облачные среды и пограничные устройства. MXNet имеет активное сообщество и предлагает обширную документацию и учебные пособия.

2.3.1.1 Примеры имплементации

**OpenCV:**

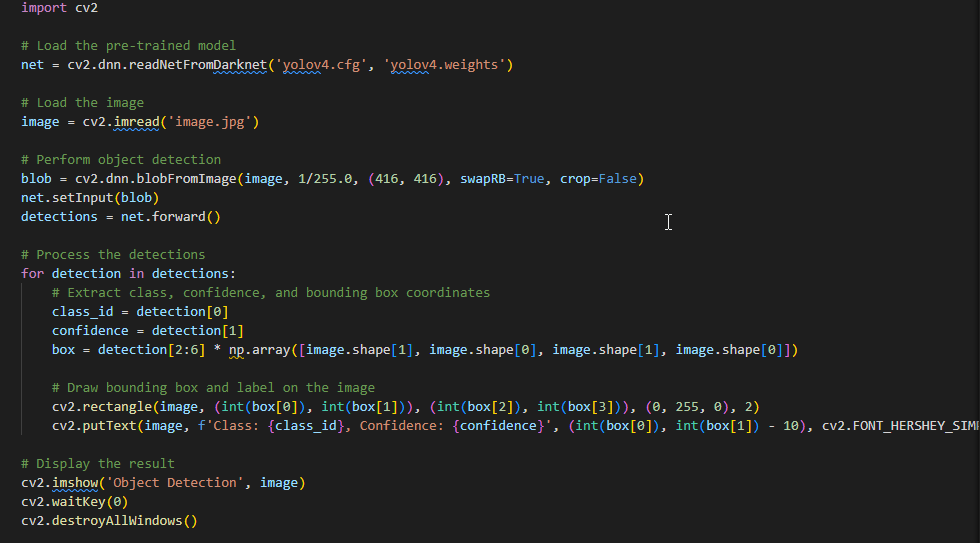


Рис. 2.2

**TensorFlow Object Detection API:**

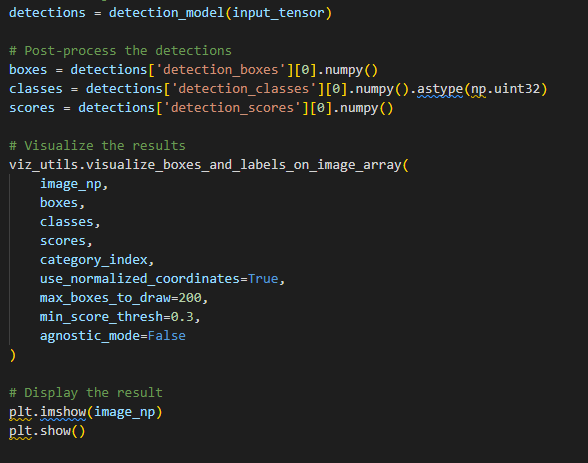


Рис 2.3

**PyTorch:**

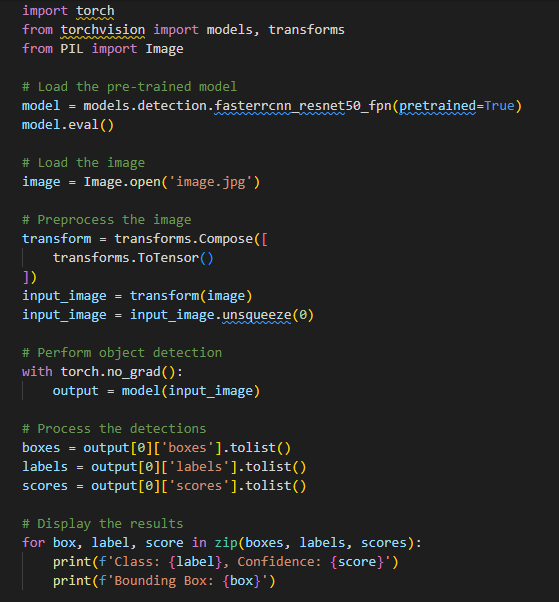


Рис.2.4

**Detectron2:**

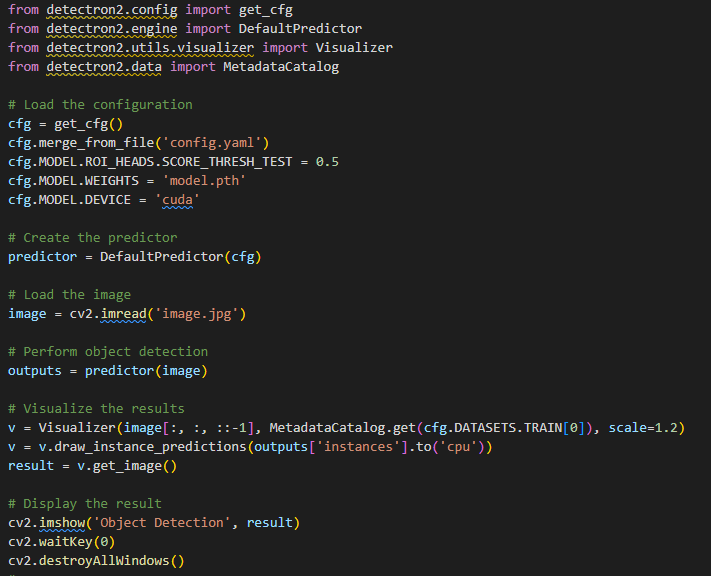


Рис 2.5

**MXNet:**

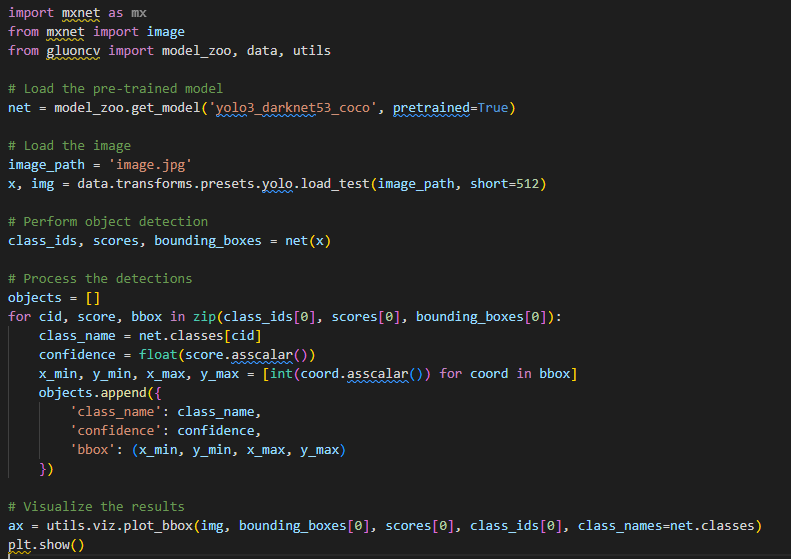


Рис 2.6

Вывод: В целом, у каждой библиотеки/фреймворка есть свои сильные и слабые стороны:

OpenCV известна своей простотой использования, обширной функциональностью и эффективной производительностью.

TensorFlow Object Detection API предлагает высокоуровневый API и обширную экосистему, что делает его подходящим для пользователей TensorFlow.

PyTorch обеспечивает гибкость и интуитивно понятный синтаксис, его предпочитают исследователи и разработчики.

Detectron2 является модульным, настраиваемым и широко используется в исследовательском сообществе для решения сложных задач обнаружения объектов.

Реализации YOLO, такие как Darknet и варианты в TensorFlow и PyTorch, отлично подходят для приложений обнаружения объектов в реальном времени.

MXNet делает акцент на скорости, эффективности использования памяти и масштабируемости, а также на бесшовных вариантах развертывания.

2.3.2 Сравнение библиотек для распознавания лиц

**OpenCV:**

Простота использования: OpenCV предоставляет возможности обнаружения и распознавания лиц с помощью своих предварительно обученных моделей и функций. Он предлагает простой и понятный API, что делает его относительно легким в использовании для базовых задач распознавания лиц.

Скорость: Алгоритмы распознавания лиц OpenCV оптимизированы для производительности, обеспечивая обработку в реальном или близком к реальному времени в зависимости от аппаратной конфигурации.

Другие факторы: OpenCV предоставляет дополнительные функции, помимо распознавания лиц, такие как обнаружение лиц, обработка изображений и задачи компьютерного зрения. Он имеет обширную документацию, большое сообщество и поддерживает множество языков программирования.

**dlib:**

Простота использования: dlib предлагает полный набор инструментов для распознавания лиц, включая предварительно обученные модели и утилиты для обнаружения лиц, определения ориентиров и извлечения признаков. Он имеет хорошо документированный API, но для эффективного использования может потребоваться некоторое знакомство с C++ или Python.

Скорость: dlib известен своей эффективностью и скоростью. В ней используются высоко оптимизированные алгоритмы, позволяющие распознавать лица в реальном времени на мощном оборудовании.

Прочие факторы: модели распознавания лиц dlib обучаются на относительно небольшом наборе данных, что может повлиять на производительность при решении некоторых задач. Тем не менее, это надежная и универсальная библиотека для задач, связанных с лицами, с поддержкой различных платформ и языков программирования.

**FaceNet:**

Простота использования: FaceNet, основанная на глубоком обучении, требует некоторого знакомства с глубокими нейронными сетями и фреймворками, такими как TensorFlow или PyTorch. Она включает в себя обучение или использование предварительно обученных моделей для распознавания лиц. Реализация может быть сложной и требовать специальных знаний.

Скорость: FaceNet может требовать больших вычислительных затрат, особенно на этапе вывода. Однако при оптимизированной реализации и аппаратном ускорении он может достичь производительности в реальном времени.

Другие факторы: FaceNet известен своей высокой точностью и способностью генерировать вкрапления лица, которые передают уникальные черты лица. Она широко применяется в научных исследованиях и используется в различных приложениях для распознавания лиц.

**VGGFace:**

Простота использования: VGGFace предоставляет предварительно обученные модели глубокого обучения для распознавания лиц, что позволяет пользователям использовать существующие архитектуры для задач идентификации лиц. Для эффективного использования требуется некоторое знакомство с такими фреймворками глубокого обучения, как Keras или TensorFlow.

Скорость: Скорость работы VGGFace зависит от выбранной архитектуры модели, реализации и аппаратного обеспечения. Она может быть не так оптимизирована для работы в реальном времени по сравнению с другими библиотеками/фреймворками.

Другие факторы: Модели VGGFace известны своей высокой точностью и способностью к обобщению. Они были обучены на крупных массивах данных распознавания лиц, что делает их пригодными для различных приложений распознавания лиц.

**PyTorch и TensorFlow:**

Простота использования: И PyTorch, и TensorFlow предоставляют высокоуровневые API и обширную документацию, что делает их доступными для пользователей с разным уровнем знаний. Они предлагают широкий спектр предварительно обученных моделей распознавания лиц и утилит.

Скорость: Скорость распознавания лиц с помощью PyTorch и TensorFlow зависит от архитектуры модели, реализации и аппаратного обеспечения. С помощью оптимизации и аппаратного ускорения можно добиться производительности в реальном или близком к реальному времени.

Другие факторы: PyTorch и TensorFlow - широко распространенные фреймворки глубокого обучения с обширной экосистемой. Они обеспечивают гибкость, позволяя исследователям и разработчикам настраивать и экспериментировать с моделями распознавания лиц. Они имеют активные сообщества, обширные ресурсы и поддержку различных платформ и языков программирования.

**ArcFace:**

Простота использования: ArcFace, основанная на глубоком обучении, требует знакомства с глубокими нейронными сетями и фреймворками, такими как TensorFlow или PyTorch. Она предполагает обучение или использование предварительно обученных моделей для распознавания лиц. Реализация может быть сложной и требовать специальных знаний.

Скорость: ArcFace разработан для высокопроизводительного распознавания лиц. Благодаря оптимизированной реализации и аппаратному ускорению он может достичь производительности в реальном времени, в зависимости от аппаратной конфигурации.

Другие факторы: ArcFace известен своей самой высокой точностью в задачах распознавания лиц. В нем используется угловая функция потерь softmax для повышения дискриминационной способности вкраплений, что делает его подходящим для сложных сценариев распознавания лиц.

* + - 1. Примеры имплементации

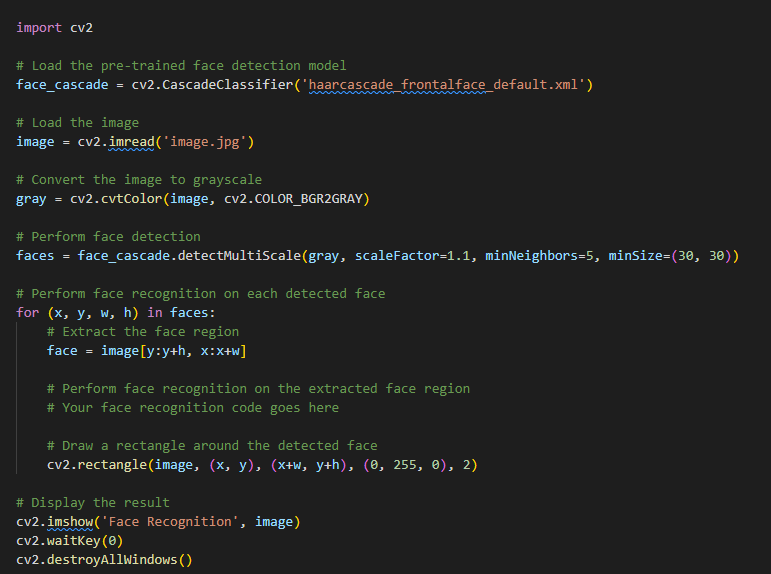
**OpenCV:**

Рис.2.7

**Dlib:**

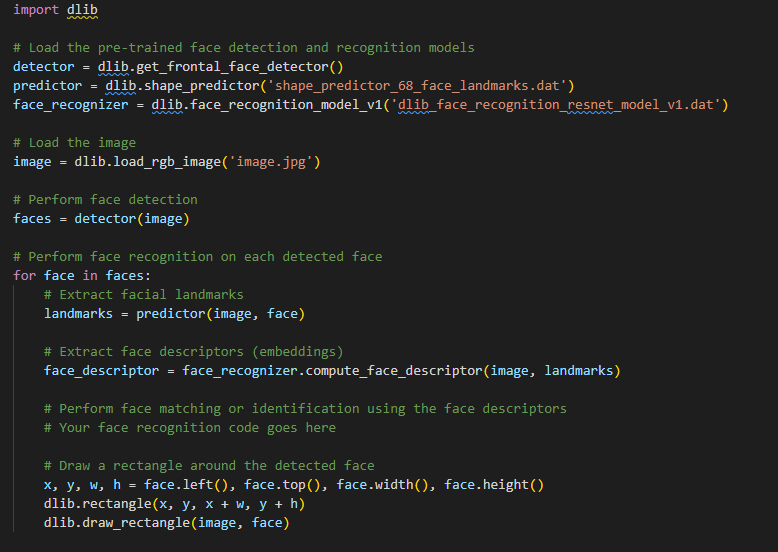


Рис2.8

**FaceNet (TensorFlow):**

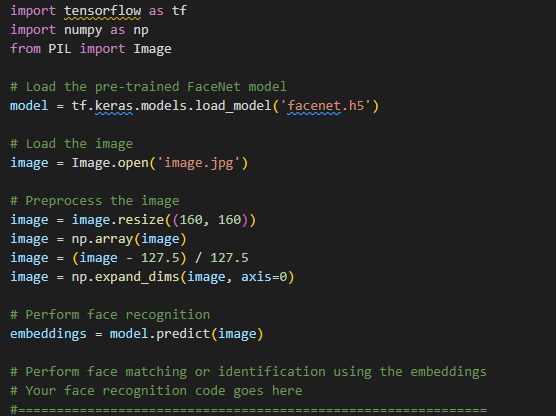


Рис2.9

**VGGFace (Keras):**

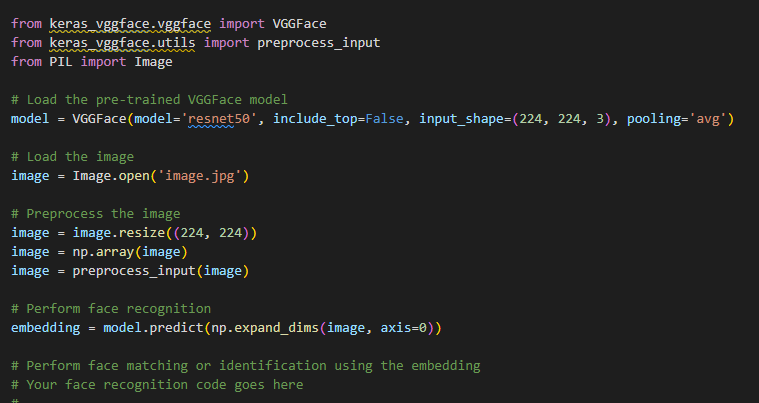


Рис 2.10

**ArcFace (PyTorch):**

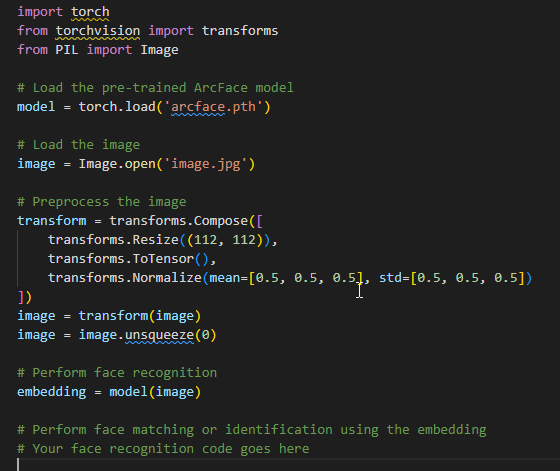


Рис 2.11

Вывод: В итоге для написания программы была выбрана библиотек OpenCV. Библиотека содержит в себе все необходимые инструменты для выполнения необходимых задач. А простота имплементации, обширная документация и поддержка сообщества вокруг данного модуля поможет исправить возникающие сложности

Глава 3 Проект

* 1. Разработка архитектуры проекта

Для разработки архитектуры проекта прибегнем к UML и средствам его использующим. В данном случае используется Visual Paradigm.

Общая структура программы выглядит следующим образом

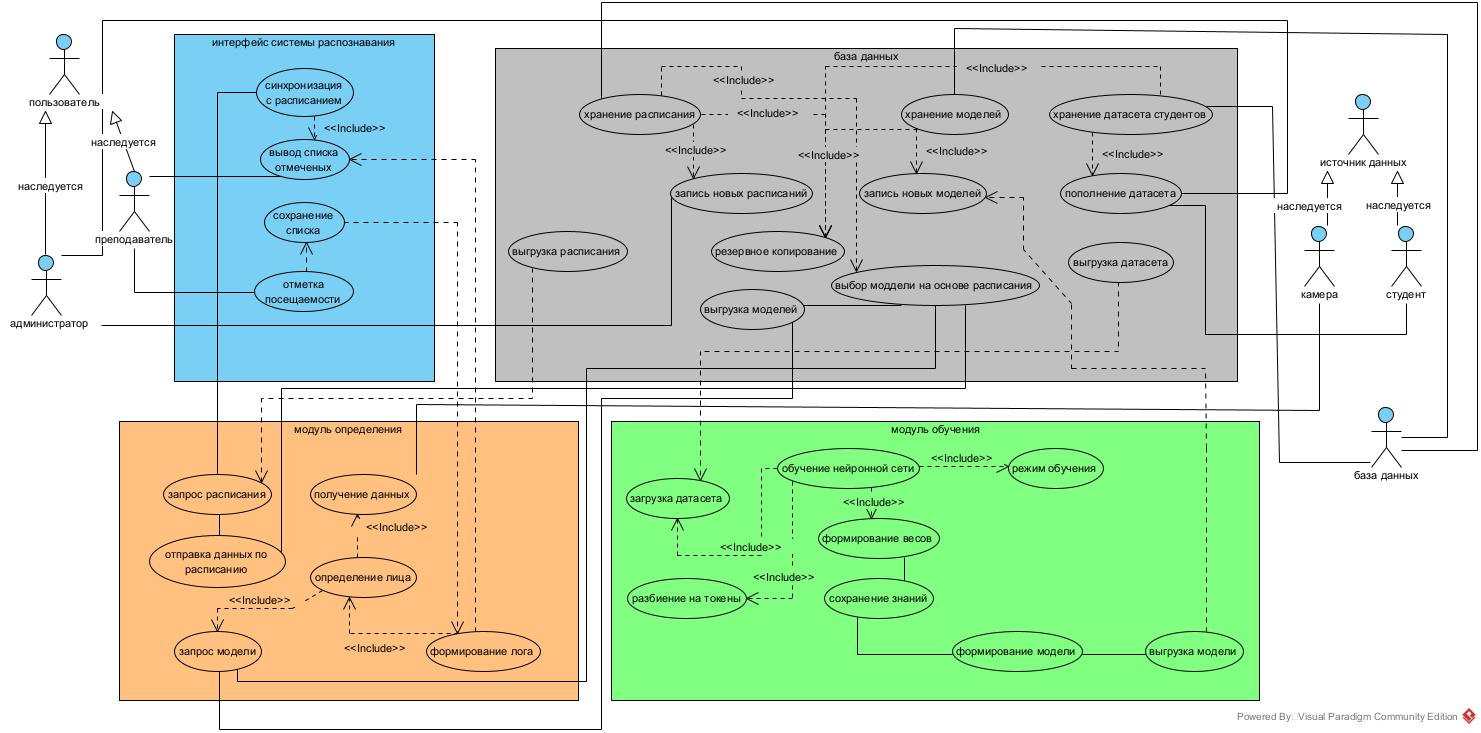


Рис 3.1(*структура программы*)

На диаграмме показано роли учителя(администратора) и студента в данной программе где учитель выступает в качестве оператора системы и ученики играют роль источника данных как на начальном этапе, так и в процессе использования программы.

При этом процесс использования программы следующий

1. создаем студента через детали студента
2. после заполнения данных нажать загрузить фото
3. программа возьмет 100 фотографий
4. далее из главного меню нажать тренировать данные (программа пройдет по всем фото и составит классификатор для распознавания)
5. после уже можно сканировать лицо (программа отобразит вид с камеры и обведет лицо зеленым или красным + выведет данные по ученику + составит автолог)
6. далее заходим в отметку посещения и смотрим логи, выгружаем результаты (в виде csv файла)
7. для просмотра лога.csv используем excel делиметр","

Этот же процесс в UML

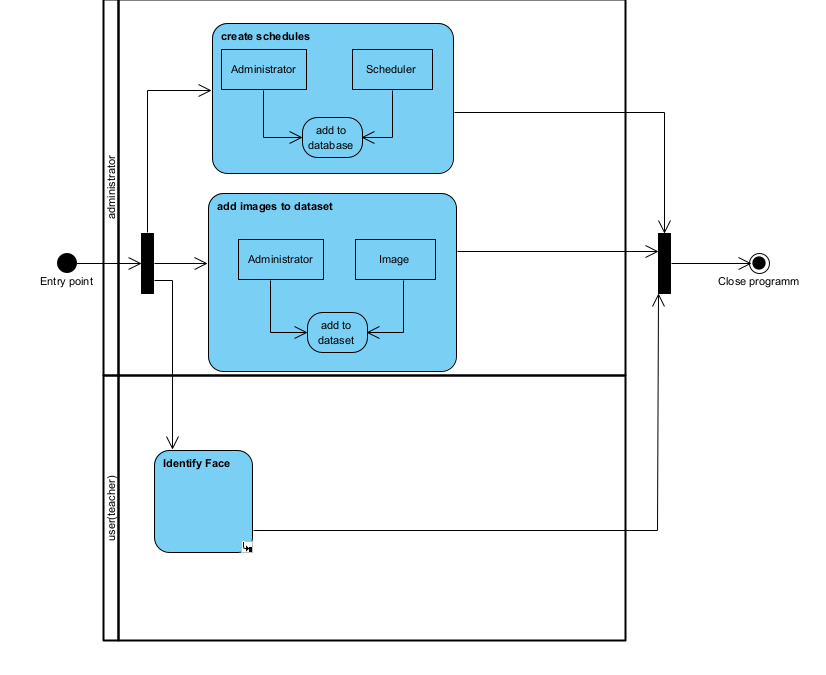


Рис 3.2(*диаграмма использования*)

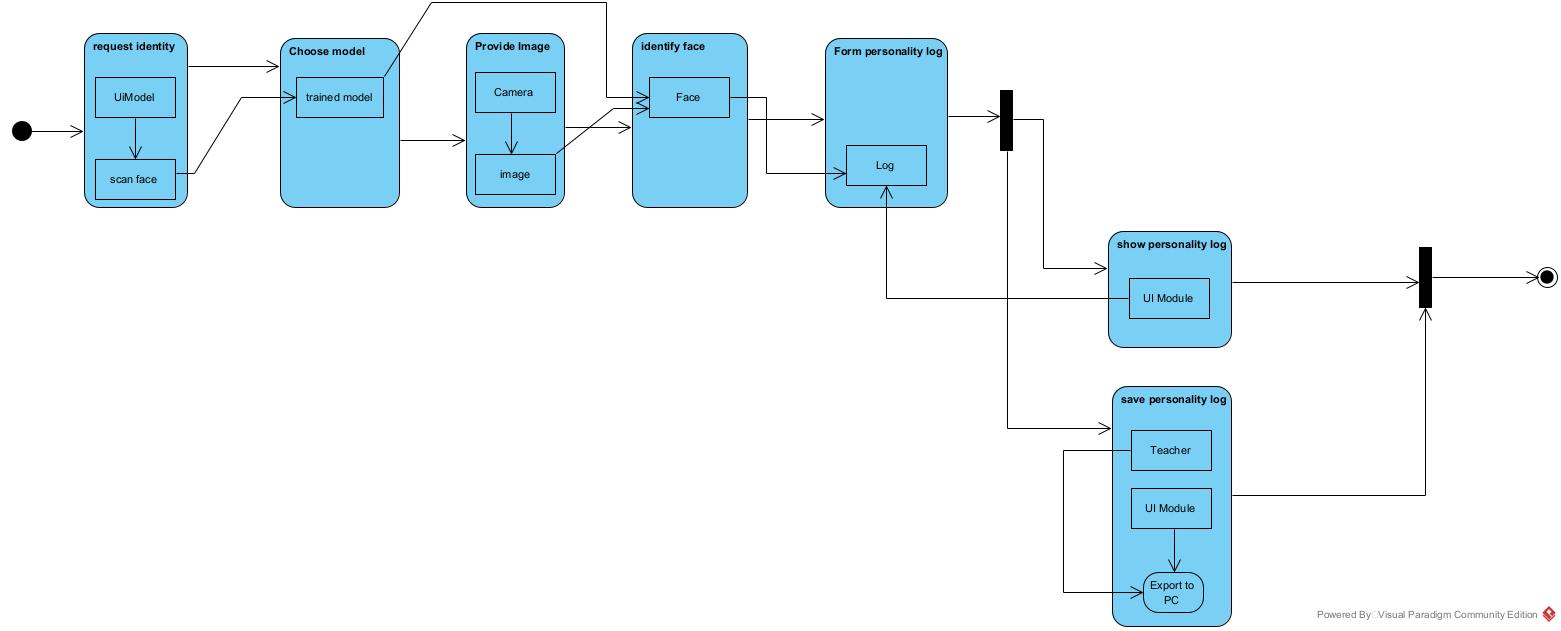


Рис 3.3(*диаграмма использования*)

* 1. Разработка интерфейса и программных модулей

Исходя из модели программы необходимо разработать несколько модулей

1. Форма для заполнения данных студентов для хранения в Базе Данных (далее БД)
2. Модуль для взятия фото (используется haarcascade)
3. Модуль для тренировки данных в classifier (используется LBPH)
4. Модуль для распознавания лица
5. Форма для заполнения журнала
6. Форма для навигации по программе

Начнем с формы навигации:

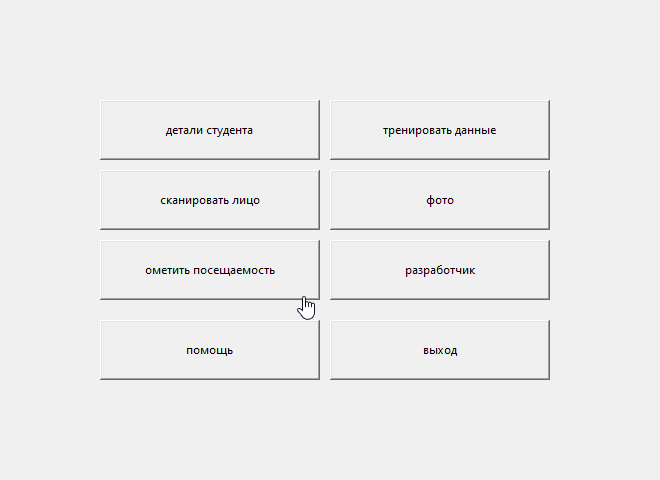


Рис 3.4(*форма навигации*)

Данная форма содержит в себе кнопки для навигации по формам, а так же для сканирования лица и тренировки данных из дополнительных функций имеем кнопку для просмотра фото и помощи

Данная форма создавалась при помощи встроенной графической библиотеки tkinter. Ее код следующий



Рис 3.5(*Код Формы навигации*)

Далее перейдем к форме для заполнения данных студентов

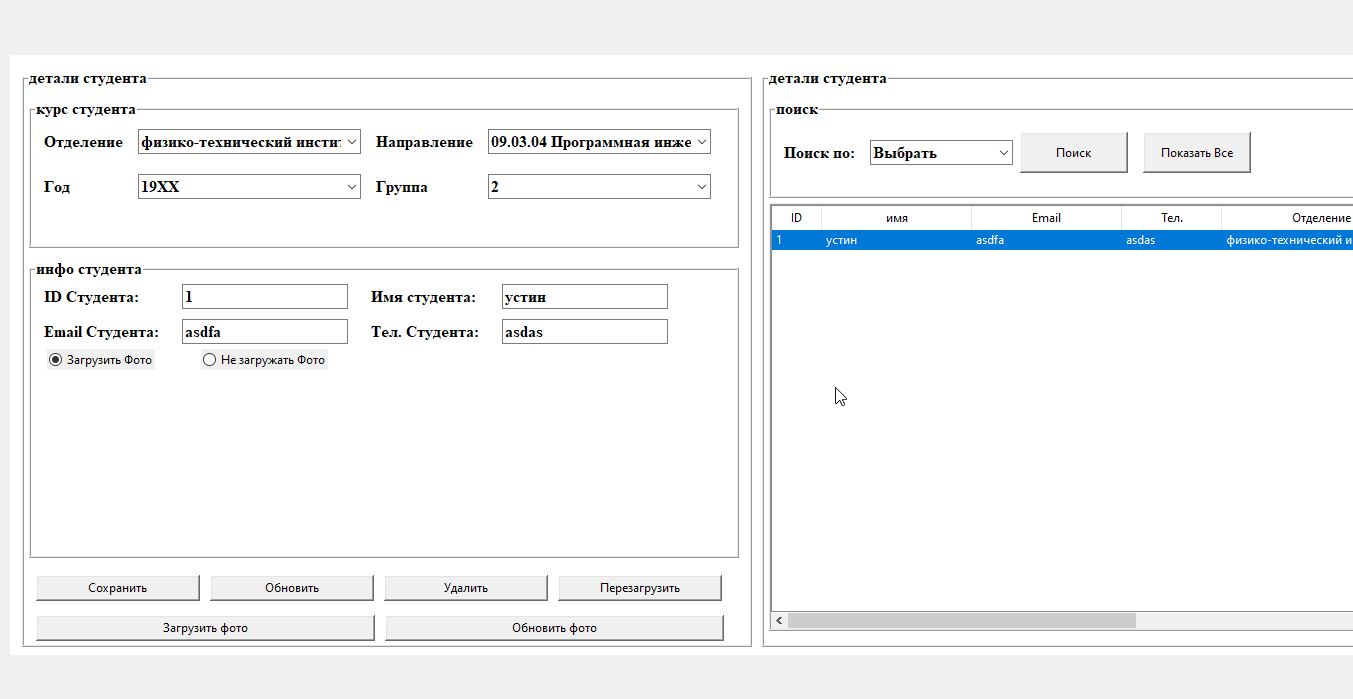


Рис 3.6(*форма заполнения данных студентов*)

Данная форма содержит в себе функционал для манипуляции с данными в БД, а так же для заполнения датасета фото с привязкой к данным.

Для работы этой формы был установлен и сконфигурирован MySQL сервер а также настроена БД и пользователь

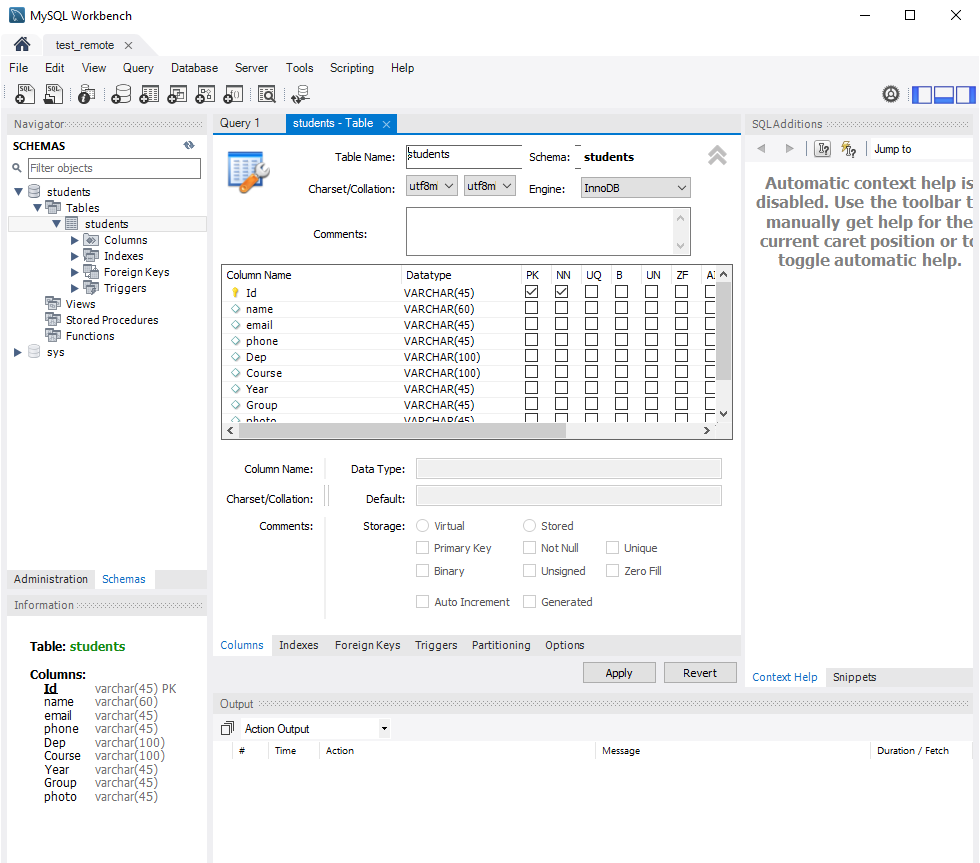


Рис 3.7(*база данных*)

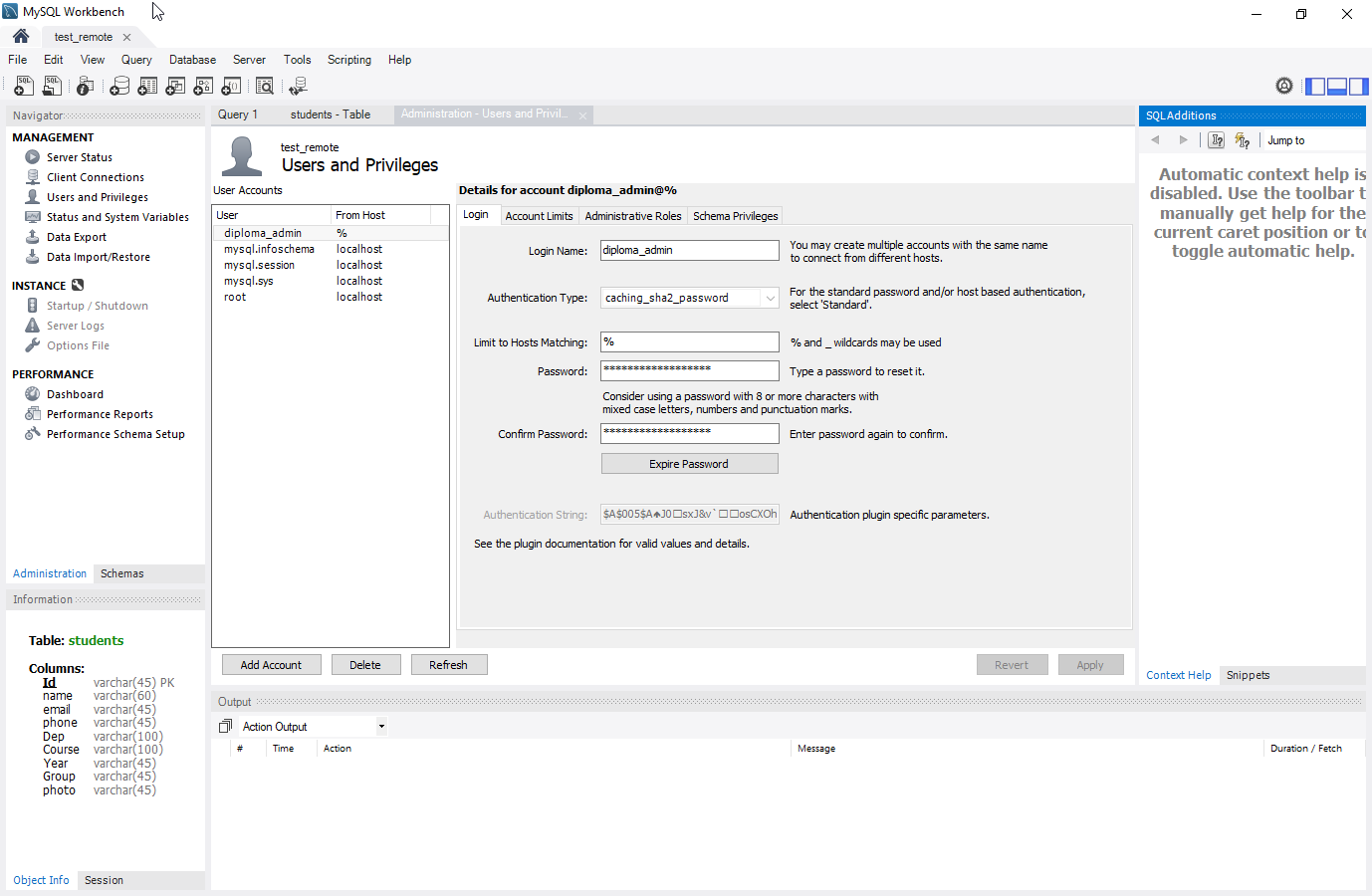


Рис 3.8(*конфигурация пользователя*)

Фрагмент кода, отвечающий за коммуникацию с БД выглядит следующим образом



Рис 3.9(*код программы*)

Для генерации датасета процесс следующий:

1. Для связи позиции из БД с фото мы привязываем фото id студента

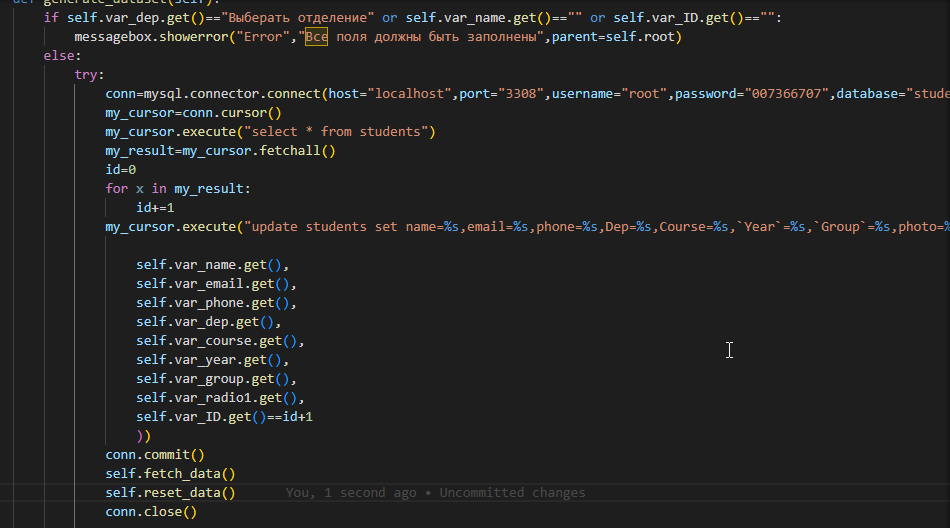


Рис 3.10 (*код программы*)

1. Далее мы получаем feed с камеры



Рис 3.11 (*код программы*)

1. Переводим изображение в серый и проводим обнаружение лица при помощи haarcascade\_classifier.xml (используем здесь default)

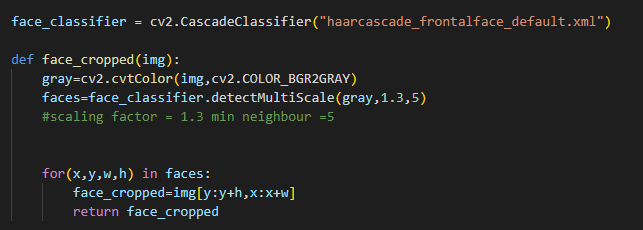


Рис 3.12 (*код программы*)

1. Далее обрезаем изображение по границам найденного лица
2. Растягиваем изображение до 450х450 и сохраняем в папку



Рис 3.13 (*код программы*)

В итоге получаем данное изображение:



Рис 3.14(*результат*)

Следующий шаг - это тренировка классификатора по полученным данным

Для его создания делаем следующее:

Инициализация переменных:

1. data\_dir: Указывает путь к каталогу, в котором находятся обучающие данные.
2. path: Создает список путей к файлам, соединяя data\_dir с каждым именем файла в каталоге.

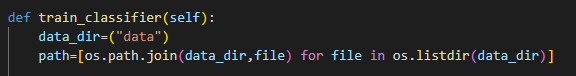


Рис 3.15 (*код программы*)

Инициализирующие списки:

1. faces: Пустой список для хранения изображений лиц.
2. ids: Пустой список для хранения соответствующих идентификаторов лиц.

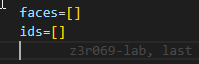


Рис 3.16 (*код программы*)

Циклический просмотр данных для обучения:

1. Итерация по каждому пути файла изображения в path.
2. Открытие каждого изображения с помощью Image.open() PIL и преобразование его в градации серого с помощью метода convert('L').
3. Преобразование изображения PIL в массив NumPy с помощью np.array().
4. Извлечение ID из имени файла изображения путем разделения пути к файлу и извлечения второй части (ID) после точки.
5. Добавление массива NumPy изображения в список лиц.
6. Добавление идентификатора в список ids.
7. Преобразование списков в массивы NumPy:
8. Преобразование списка ids в массив NumPy с помощью np.array(ids).

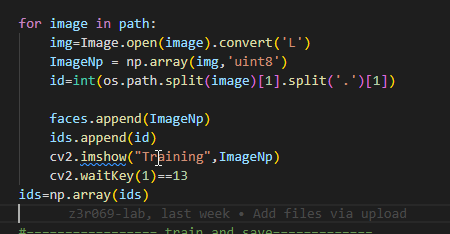


Рис 3.17 (*код программы*)

Обучение классификатора:

1. Создание экземпляра распознавателя лиц LBPH (Local Binary Patterns Histograms) с помощью cv2.face.LBPHFaceRecognizer\_create().
2. Обучение классификатора с помощью метода train(), передавая лица и идентификаторы в качестве аргументов.
3. Сохранение обученного классификатора в файл с именем "classifier.xml" с помощью метода write().
4. Закрытие всех открытых окон OpenCV с помощью метода cv2.destroyAllWindows().

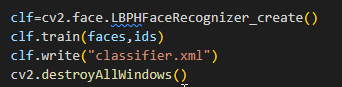


Рис 3.18 (*код программы*)

Распознавание лица:

Вот что делает код:

Открытие и очистка файла посещаемости:

* 1. Открывает файл "attendance.csv" в директории "attendance" в режиме чтения и записи ("r+").
  2. Усекает содержимое файла, чтобы очистить его, используя метод truncate(0).
  3. Закрывает файл с помощью file.close().

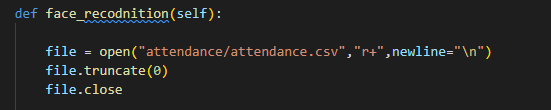


Рис 3.19 (*код программы*)

Определение вспомогательной функции draw\_boundary:

Эта функция рисует границу прямоугольника вокруг обнаруженных лиц и предсказывает их идентификаторы с помощью классификатора.

Она также извлекает дополнительную информацию из базы данных MySQL, используя предсказанный ID.

Если достоверность предсказания выше 77, она отображает ID, имя, факультет и информацию о курсе на изображении и отмечает посещаемость.

Если достоверность предсказания ниже 77, на изображение выводится надпись "Unknown".

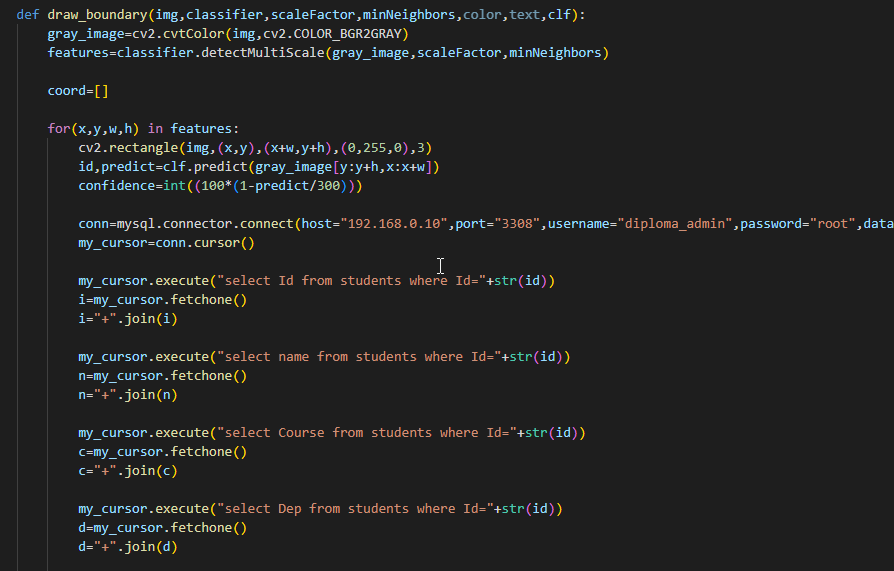


Рис 3.20 (*код программы*)

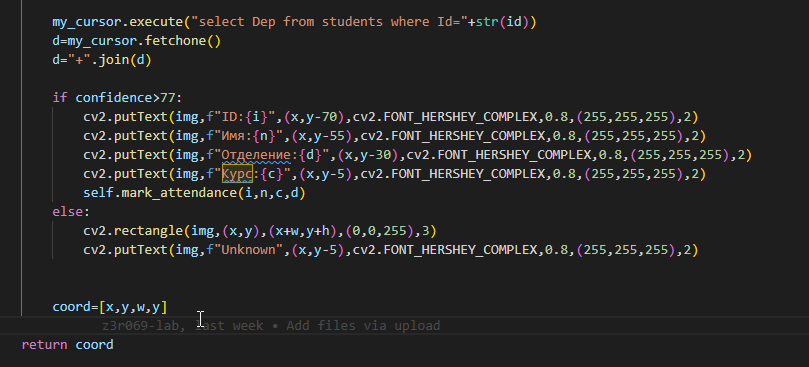


Рис 3.21 (*код программы*)

Определение вспомогательной функции recognize:

Эта функция вызывает draw\_boundary для рисования границ и отображения информации на изображении.

Она принимает на вход изображение, классификатор (clf) и каскад лиц (faceCascade).

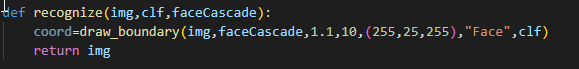


Рис 3.22 (*код программы*)

Загрузка классификатора и каскадного классификатора:

Загружает каскадный классификатор лиц из файла "haarcascade\_frontalface\_default.xml" с помощью cv2.CascadeClassifier.

Создает экземпляр распознавателя лиц LBPH с помощью cv2.face.LBPHFaceRecognizer\_create().

Загружает обученный классификатор из файла "classifier.xml" с помощью clf.read().

Запуск цикла видеозахвата и распознавания:

Открывает объект видеозахвата с помощью cv2.VideoCapture(0) для захвата видео с камеры по умолчанию.

Входит в бесконечный цикл для непрерывного чтения кадров из видеопотока.

Вызывает функцию recognize на каждом кадре для обнаружения и распознавания лиц.

Отображает обработанный кадр изображения с границами и информацией с помощью cv2.imshow().

При нажатии клавиши Enter (код клавиши 13) цикл прерывается.

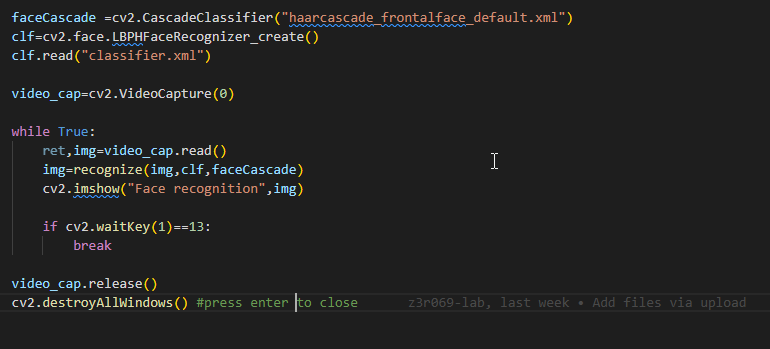


Рис 3.23 (*код программы*)

Освобождение ресурсов и закрытие окон:

Освобождает объект видеозахвата с помощью video\_cap.release().

Закрытие всех окон OpenCV с помощью cv2.destroyAllWindows().

Чтобы выйти из программы, нужно нажать клавишу Enter, когда окно "Распознавание лиц" находится в фокусе.

Результат работы:

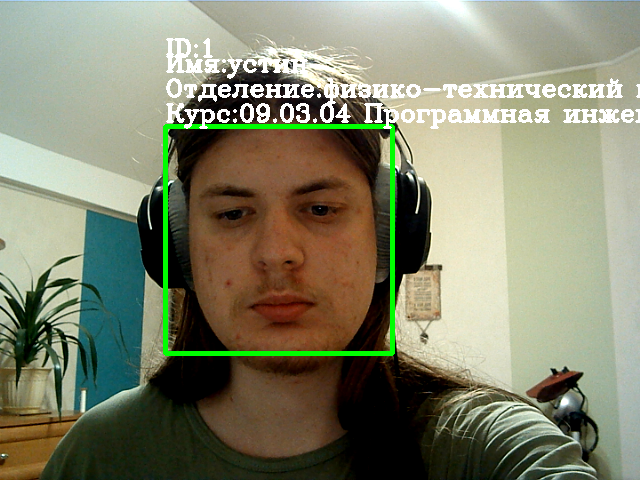


Рис 3.24 (*результат*)

Отметка посещаемости осуществляется в момент сканирования лица. Данные сохраняются в csv формате которые можно открыть в данной форме. А так же при необходимости отредактировать и сохранить

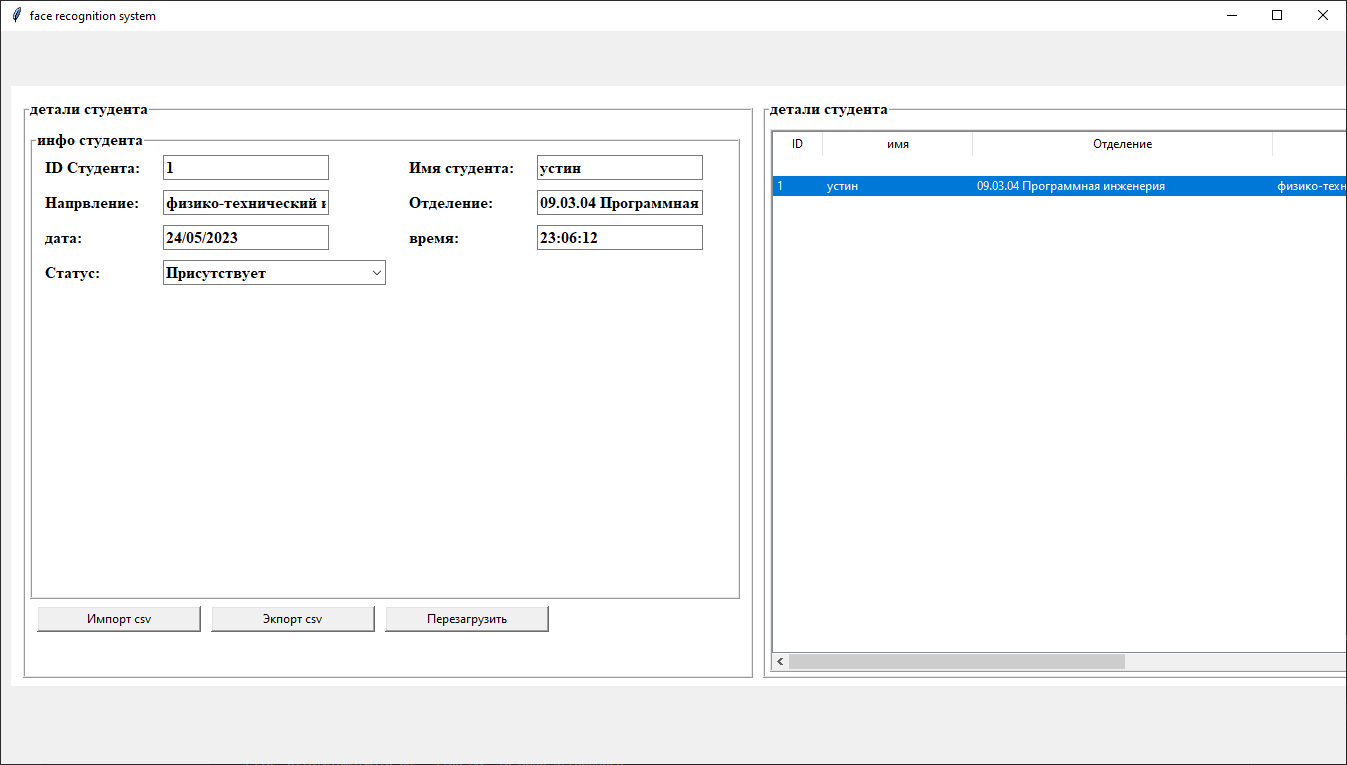


Рис 3.25 (*форма для отметки посещаемости*)

Код сохранения csv выглядит следующим образом

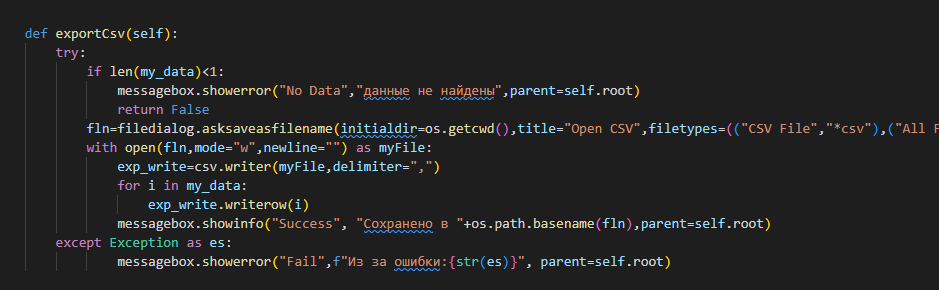


Рис 3.26 (*код программы*)

Результатом работы программы является лог, который можно открыть в Excel.

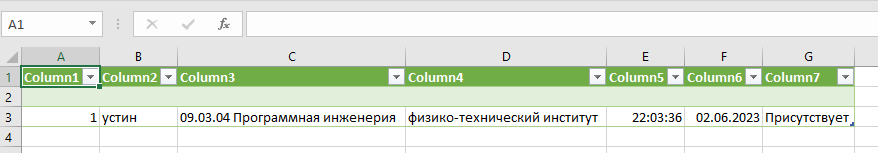


Рис 3.27 (*результат*)

Завершение:

В ходе проведенной работы была достигнута цель по написанию программы при помощи которой можно отметить посещаемость студентов с использованием технологии машинного зрения. С одной стороны, были достигнуты все поставленные задачи, но также всегда имеется место для усовершенствования продукта.

Были рассмотрены различные алгоритмы для распознавания лица и отслеживания объектов. Также било рассмотрено и произведено сравнение технологий, отвечающие за вышеупомянутые алгоритмы.

Работа имеет теоретическую ценность для тех кто желает ознакомиться с технологиями отвечающими за машинное зрение, понять как работают те или иные алгоритмы, в каких случаях их использовать. Также позволяет узнать какие библиотеки отвечают за применение алгоритмов и как их применить на практике. С практической же стороны имеем продукт отвечающий поставленной цели и полный его разбор. Так что те, кто желает повторить данный проект или применить полученные знания для разработки собственной системы.

Используемая литература

1. <https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms>
2. <https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b>
3. <http://www.scholarpedia.org/article/Eigenfaces>
4. <https://en.wikipedia.org/wiki/Local_binary_patterns>
5. <http://www.scholarpedia.org/article/Fisherfaces>
6. DC. He and L. Wang (1990), "Texture Unit, Texture Spectrum, And Texture Analysis", Geoscience and Remote Sensing, IEEE Transactions on, том. 28, стр. 509 - 512.
7. L. Wang and DC. He (1990), "Texture Classification Using Texture Spectrum", Pattern Recognition, Том. 23, No. 8, стр. 905 - 910.
8. T. Ojala, M. Pietikäinen, and D. Harwood (1994), "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions", Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994), Том. 1, стр. 582 - 585.
9. T. Ojala, M. Pietikäinen, and D. Harwood (1996), "A Comparative Study of Texture Measures with Classification Based on Feature Distributions", Pattern Recognition, Том. 29, стр. 51-59.
10. <https://arxiv.org/abs/2207.02696>
11. <https://imageai.readthedocs.io/en/latest/>
12. <https://neptune.ai/blog/object-detection-algorithms-and-libraries>

Приложение

Код программы

Student.py

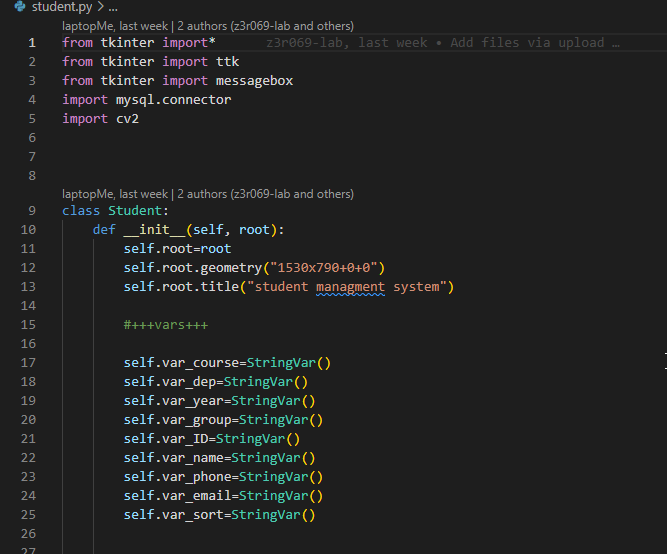


Рис. П1



Рис. П2



Рис. П3



Рис. П4



Рис. П5

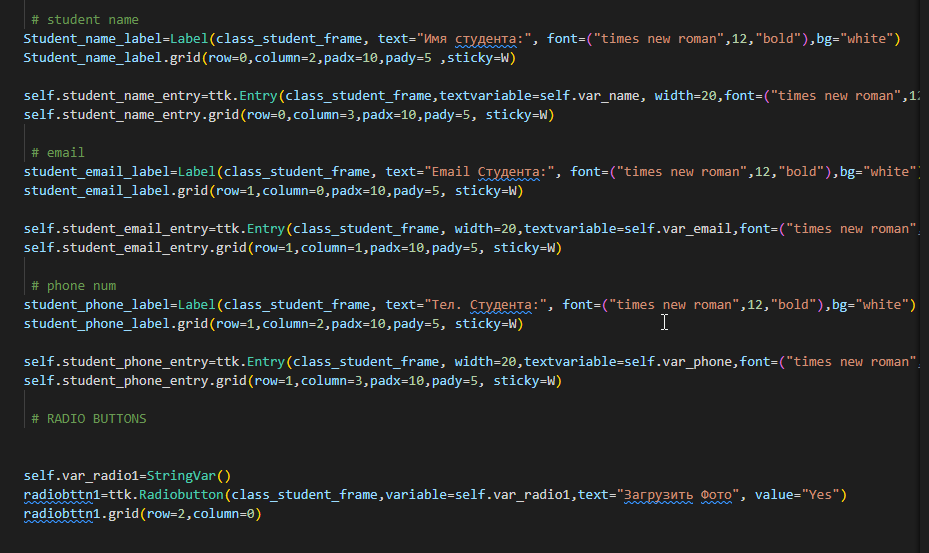


Рис. П6

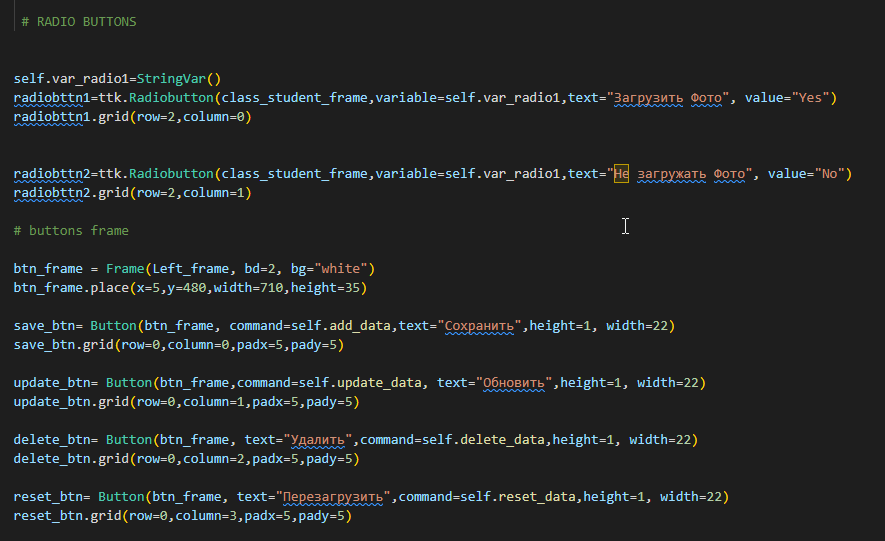


Рис. П7



Рис. П8



Рис. П9

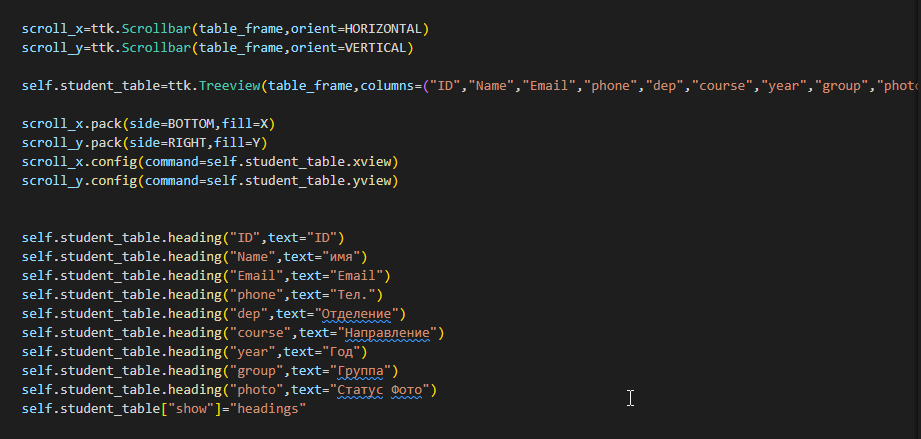


Рис. П10

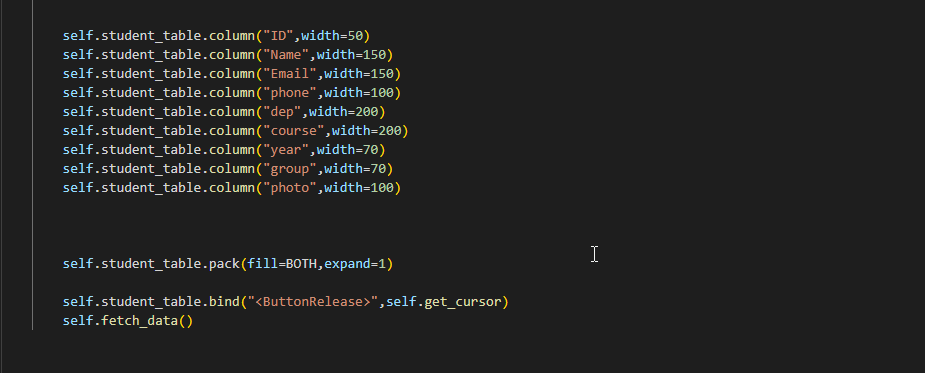


Рис. П11

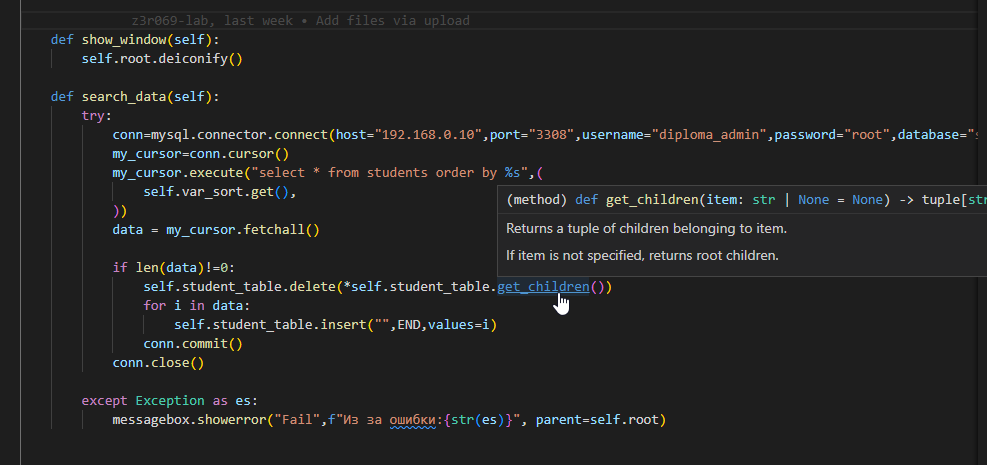


Рис. П12

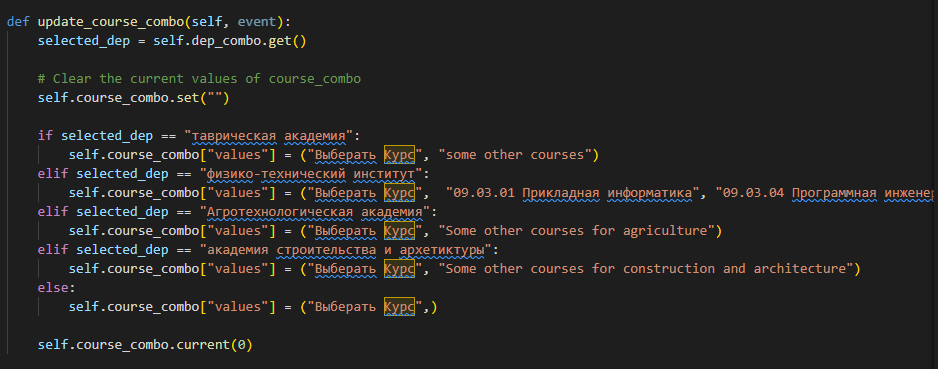


Рис. П13

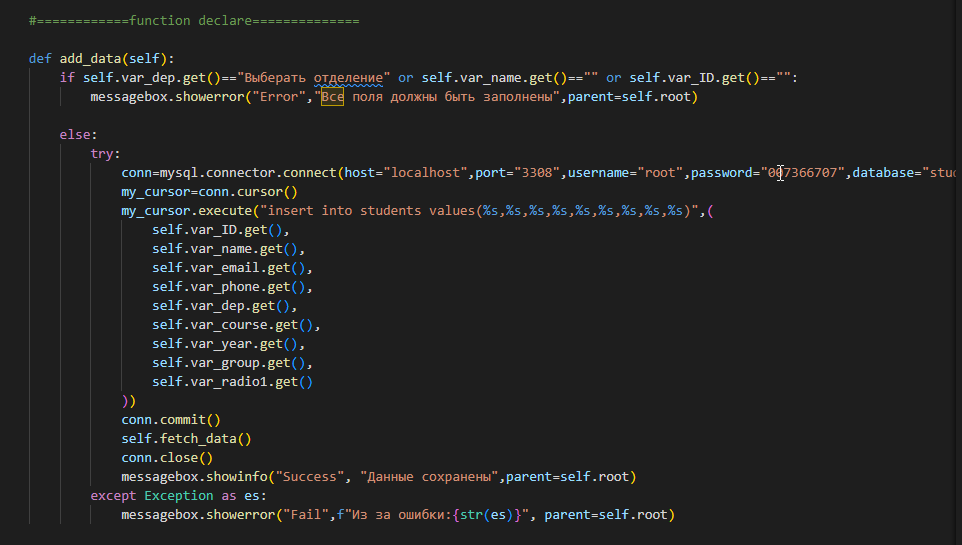


Рис. П14

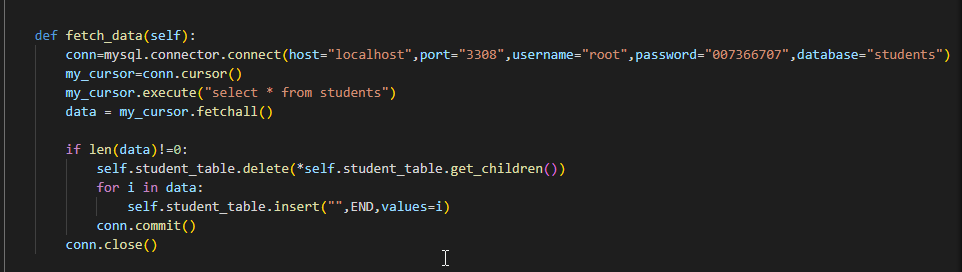


Рис. П15

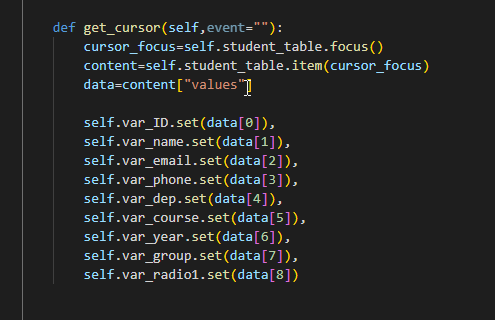


Рис. П16

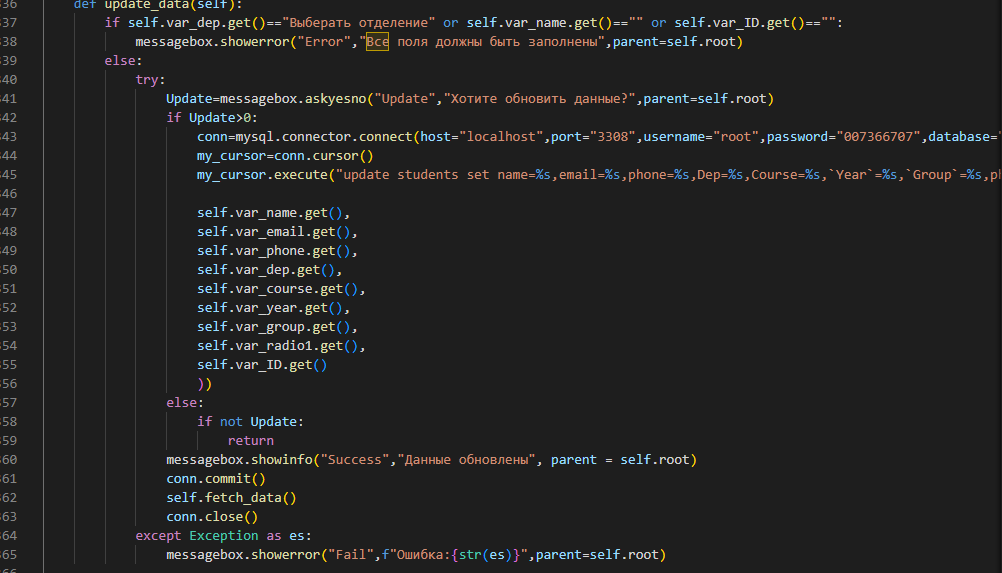


Рис. П17

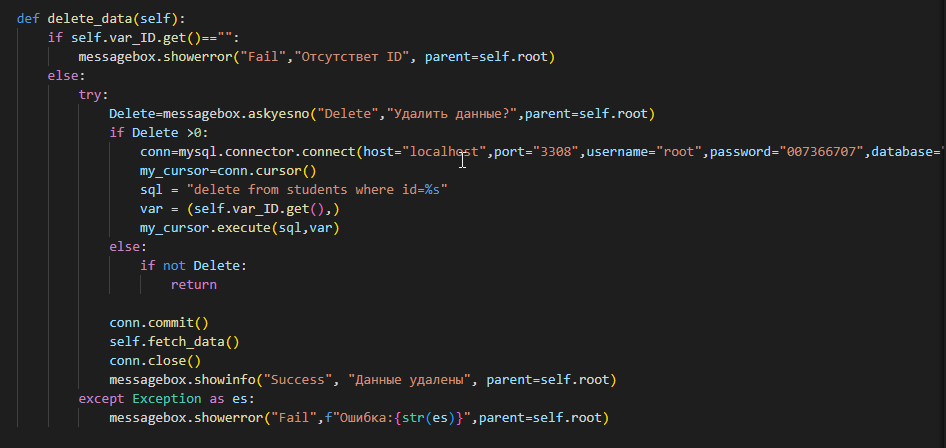


Рис. П18



Рис. П19



Рис. П20

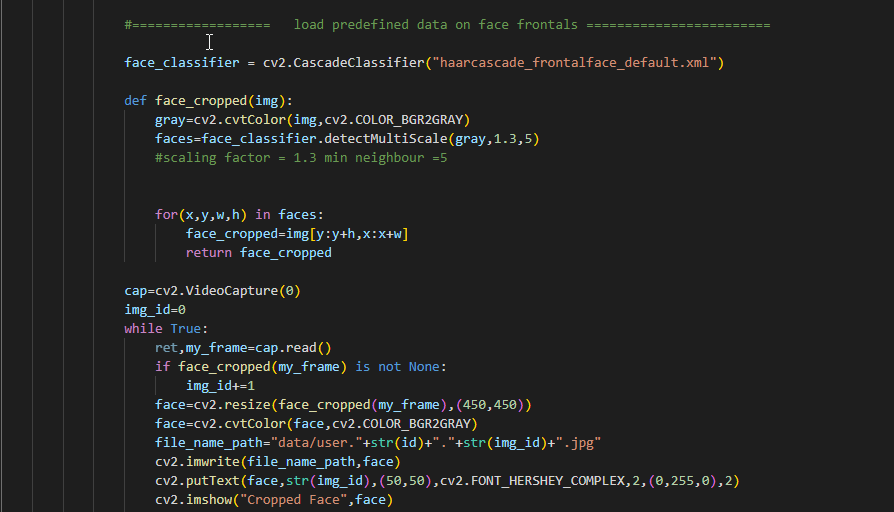


Рис. П21

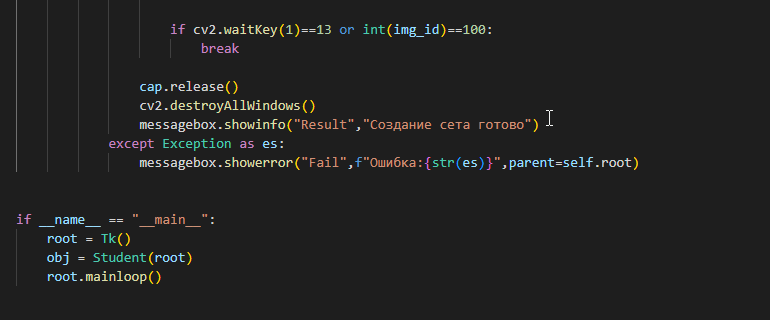


Рис. П22

Main.py



Рис. П23



Рис. П24

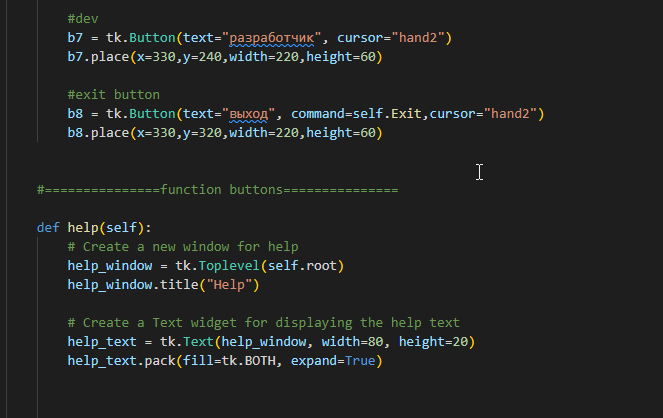


Рис. П25

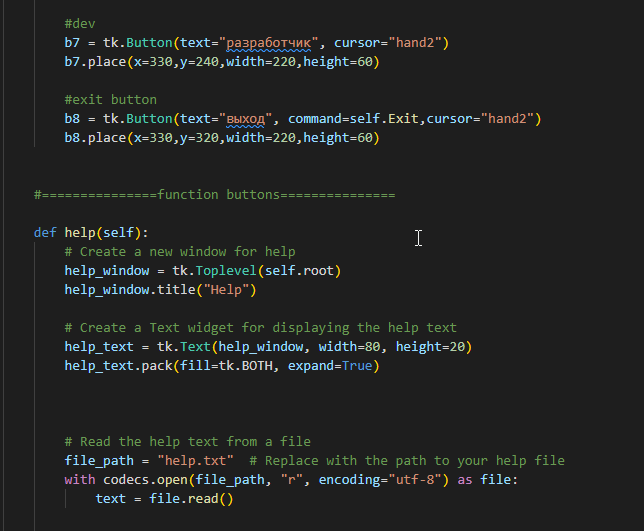


Рис. П26



Рис. П27

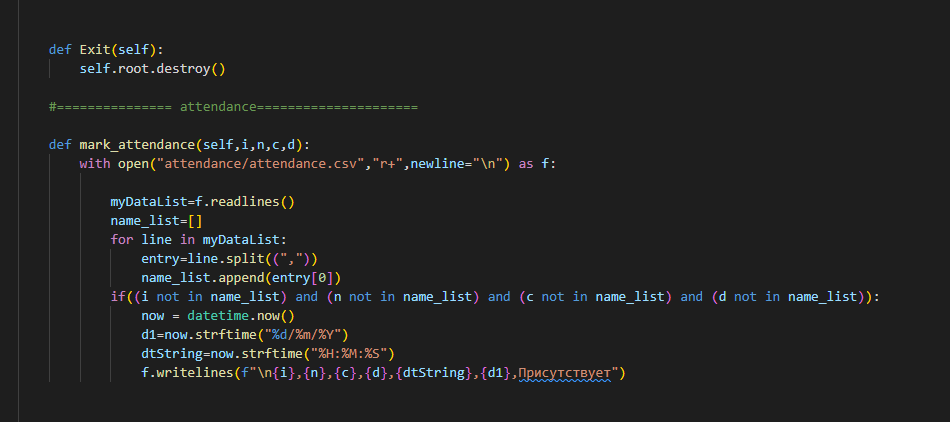


Рис. П28



Рис. П29

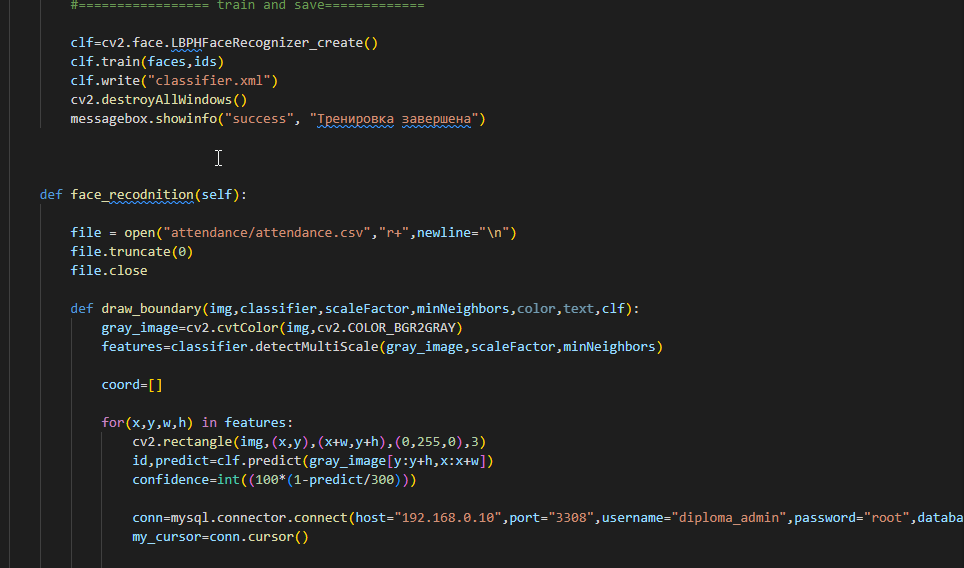


Рис. П30

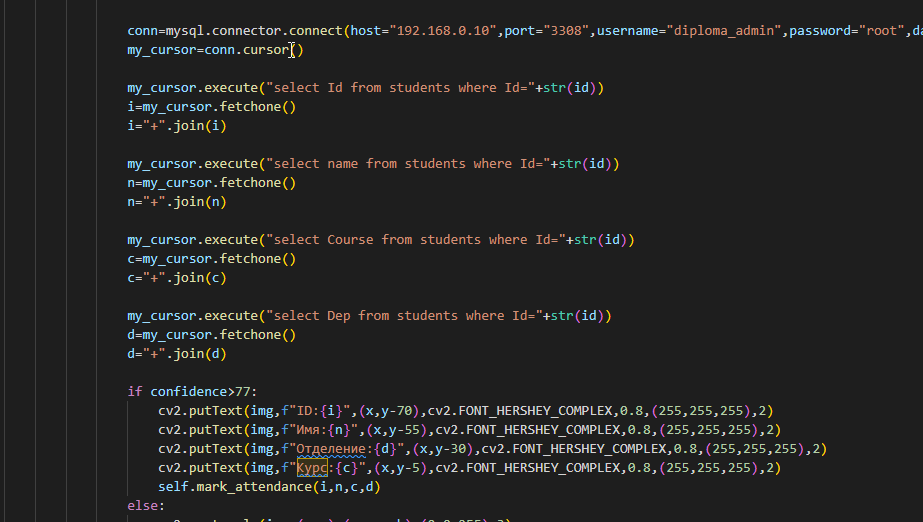


Рис. П31



Рис. П32

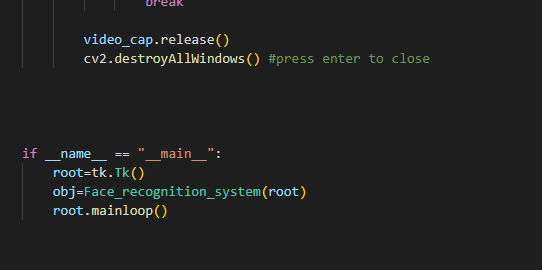


Рис. П33

Attendance.py

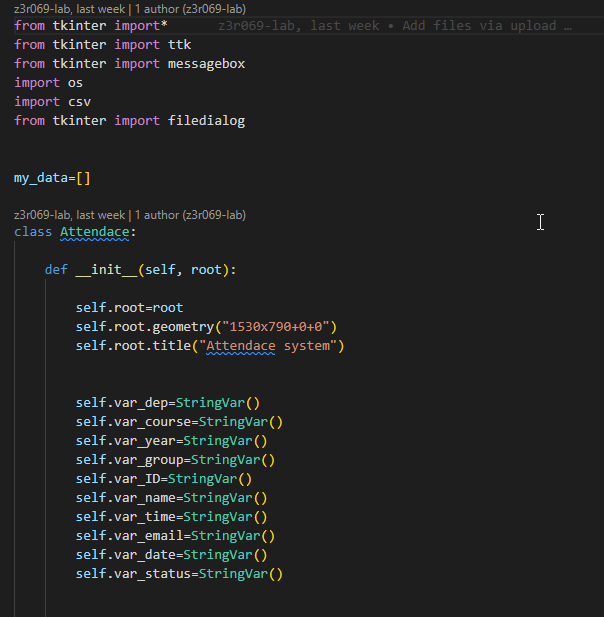


Рис. П34

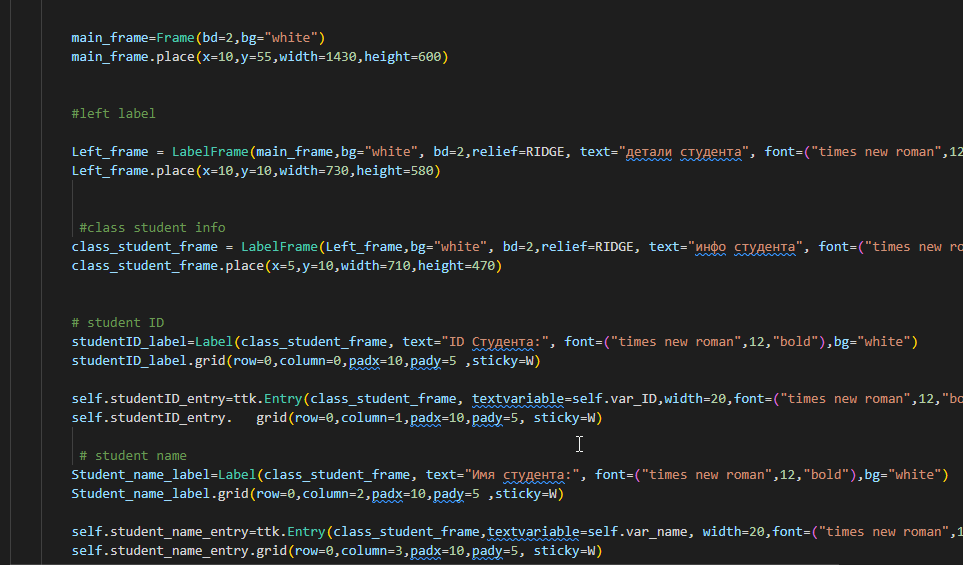


Рис. П35



Рис. П36

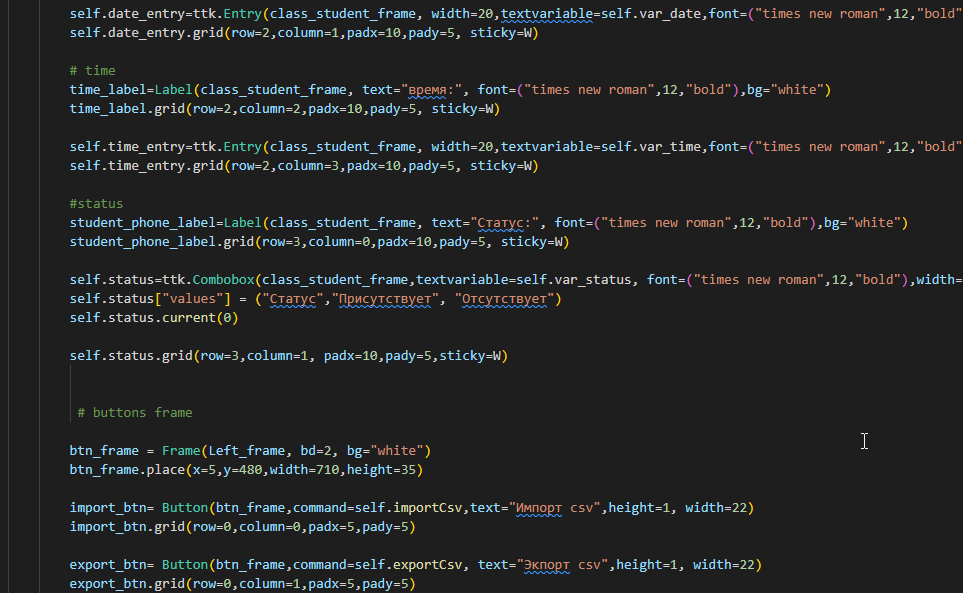


Рис. П37



Рис. П38



Рис. П39

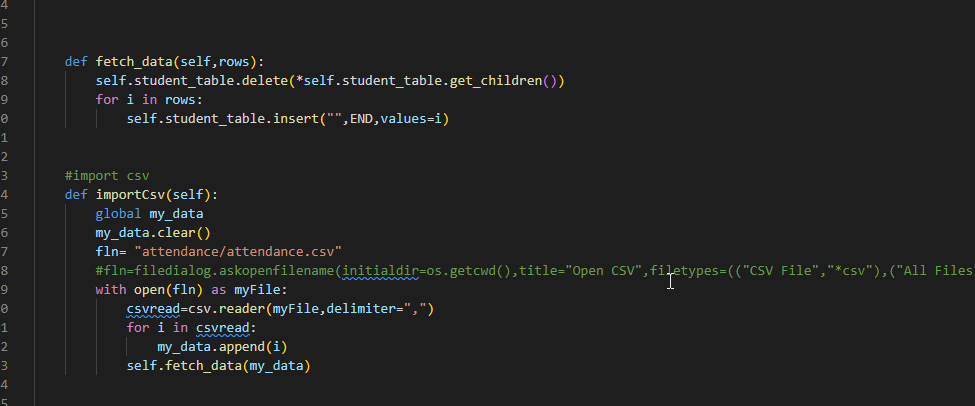


Рис. П40



Рис. П41

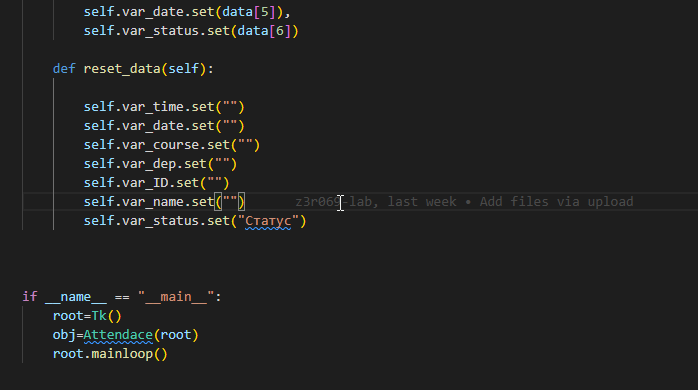


Рис. П42