|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**  **«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**  **(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»** | | | | |
| **Журнал практики** | | | | |
| Институт №3 | «Системы управления, информатика и электроэнергетика» | | | |
|  |  | | | |
| Кафедра | №304 | | Учебная группа | М3О-319Бк-22 |
|  |  | |  |  |
| ФИО обучающегося | | Григорьева Елена Владимировна | | |
|  | |  | | |
| Направление подготовки/  специальность | | 09.03.04 “Программная инженерия” д | | |
|  | | *шифр, наименование направления подготовки/специальности* | | |
|  | |  | | |
| Вид практики | | Производственная | | |
|  | | *учебная, производственная, преддипломная или другой вид практики* | | |

Руководитель практики от МАИ

/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

*(фамилия, имя, отчество) (подпись)*

Обучающийся

Григорьева Е.В. / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_ 2025 г.

*(фамилия, имя, отчество) (подпись) (дата)*

Москва 2025 г.

|  |  |
| --- | --- |
| 1. **Место и сроки проведения практики:** | |
| Наименование организации: | ООО «Открытая Мобильная Платформа» |
| Сроки проведения практики |  |
| дата начала практики: | 30.06.2025 |
| дата окончания практики: | 27.07.2026 |
| 1. **Инструктаж по технике безопасности:** | |

|  |  |  |
| --- | --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ | 30 июня 2025г. |
| *подпись проводившего* | *расшифровка подписи* | *дата проведения* |

1. **Индивидуальное задание обучающегося:**

Нужно найти и реализовать легковесный вариант распознавания лица на C++. Можно поискать готовые библиотеки, но желательно иметь минимум зависимостей. Должно быть 2 режима:

* инициализация лица и сохранение в некую БД (файл)
* распознавание лица и поиск в базе сохраненных -> Результат true/false. В обоих режимах входные данные - видеопоток.

|  |
| --- |
| 1. **План выполнения индивидуального задания обучающегося:** |

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Место проведения** | **Тема** | **Период выполнения** |
| 1. | ОМП | Установочная встреча с руководителем. Выдача задания и обсуждение организационных моментов | 30.06.2025 |
| 2. | ОМП | Анализ готовых решений и библиотек, поиск максимально подходящих зависимостей | 01.07.2025- 07.07.2025 |
| 3. | ОМП | Разработка собственного решения, подключение нужных зависимостей | 08.07.2025- 16.07.2025 |
| 4. | ОМП | Тестирование, внесение правок, анализ трудоспособности | 17.07.2025- 25.07.2025 |

|  |  |  |
| --- | --- | --- |
| **Утверждаю** | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | / / | 30 июня 2025г. |
| *подпись руководителя от МАИ* | *расшифровка подписи* | *дата утверждения\** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | / Аляутдин Р.Т. / | 30 июня 2025г. |
| *подпись руководителя от организации/предприятия* | *расшифровка подписи* | *дата утверждения\** |
| **Ознакомлена** | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | / Григорьева Е.В. / | 30 июня 2025г. |
| *подпись обучающегося* | *расшифровка подписи* | *дата ознакомления\** |

*\*Дата утверждения и ознакомления – дата начала практики*

|  |
| --- |
| 1. **Отзыв руководителя практики от организации/предприятия:** |
| ​Проделана огромная работа для выполнения поставленной задачи. И в полном объёме выполнить её в данные сроки было невозможно. Несмотря на это написанная программа выполняет часть заявленных к ней требований: распознавание лица и сохранение нового лица.  Достоинства работы: - очень подробно расписана мотивация всех принимаемых решений в ходе работы над проектом - очень подробно и достаточно детализировано описана архитектура проекта - код и репозиторий с проектом оформлен аккуратно, разделение на представленные классы логично  - разработан модуль работы с базой данных, обращение к ней и сохранение новых данных  - разработаны модули работы с нейросетями для обработки видеопотока  - оптимально извлекается и обрабатываются признаки для последующего функционала работы проекта  - при разработке студент принимал активное участие в обсуждении используемых модулей с подробной аргументацией своих решений  - выкроены из больших библиотек минимальные зависимости для уменьшения объема сборки  - реализован полный файл для эффективной сборки всех подмодулей проекта  - дополнительно написаны модульные тесты для некоторых функций проекта - проделана очень большая работа для такого промежутка времени ​Недостатки работы: - для всех версий сборки нужно провести анализ занимаемого программой (программой + нужной для её работы библиотекой) места Рекомендованная оценка: отлично.  Руководитель от организации  Аляутдин Р.Т./\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.  *(фамилия, имя, отчество) (подпись) (дата)* |

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»**

1. **Отчет обучающего по практике**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| Институт № 3 | «Системы управления, информатика и электроэнергетика» | | | |
|  |  | | | |
| Кафедра | №304 | | Учебная группа | М3О-319Бк-22 |
|  |  | |  |  |
| ФИО обучающегося | | Григорьева Елена Владимировна | | |
|  | |  | | |
| Направление подготовки/ специальность | | 09.03.04 “ Программная инженерия”\_\_\_ | | |
|  | | *шифр, наименование направления подготовки/специальности* | | |
|  | |  | | |
| Вид практики | | Производственная | | |
|  | | *учебная, производственная, преддипломная или другой вид практики* | | |

Обучающийся

Григорьева Е. В./\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_ 2025 г.

*(фамилия, имя, отчество) (подпись) (дата)*

Москва 2025 г.

Оглавление

[Тема 7](#_Toc204347930)

[Задание 7](#_Toc204347931)

[Назначение программы 7](#_Toc204347932)

[Описание программы 8](#_Toc204347933)

[Функциональные возможности программы 8](#_Toc204347934)

[Структура программы 9](#_Toc204347935)

[Сборка и развёртывание 10](#_Toc204347936)

[Методы и критерии распознавания лица 10](#_Toc204347937)

[Выбор метода детекции лица: YuNet 10](#_Toc204347938)

[MobileFaceNet 11](#_Toc204347939)

[Блок-схема разработанного алгоритма 13](#_Toc204347940)

[Блок-схема алгоритма визуализации 15](#_Toc204347941)

[Назначение и структура классов 16](#_Toc204347942)

[**Класс YuNet** 16](#_Toc204347943)

[**Класс Database** 16](#_Toc204347944)

[Тестирование программы 17](#_Toc204347945)

[Цель и задачи тестирования 17](#_Toc204347946)

[Объект и виды испытаний 17](#_Toc204347947)

[Условия и средства проведения испытаний 17](#_Toc204347948)

[Вывод 19](#_Toc204347949)

[Список использованных источников 19](#_Toc204347950)

[Приложение а. Структура проекта 20](#_Toc204347951)

[Приложение б. Код проекта 21](#_Toc204347952)

[Main.cpp 21](#_Toc204347953)

[Visualizer.cpp 23](#_Toc204347954)

[Приложение с. Результаты тестов 25](#_Toc204347955)

## Тема

Разработка легковесного приложения для детекции и распознавания лиц в видеопотоке с использованием OpenCV и MobileFaceNet на языке C++.

## Задание

Нужно найти и реализовать легковесный вариант распознавания лица на C++. Можно поискать готовые библиотеки, но желательно иметь минимум зависимостей. Должно быть 2 режима:

* инициализация лица и сохранение в некую БД (файл)
* распознавание лица и поиск в базе сохраненных -> Результат true/false. В обоих режимах входные данные - видеопоток.

## Назначение программы

Целью разработки данной программы является создание лёгкого и переносимого C++-приложения для автоматического распознавания лиц в реальном времени с использованием видеопотока с камеры. Программа предоставляет возможность:

1. Регистрировать новое лицо в базе данных;
2. Распознавать лица, уже сохранённые в базе, с последующей идентификацией личности.

Сфера применения подобного программного обеспечения достаточно широка и включает в себя:

* Системы контроля доступа (например, вход в помещения по распознаванию лица);
* Пропускные системы на предприятиях и в учебных заведениях;
* Мониторинг и авторизация в локальных компьютерных системах;
* Базовая демонстрация и прототипирование решений в области компьютерного зрения.

Разработка ведётся с особым акцентом на:

* Минимализм и лёгковесность: исключаются тяжёлые фреймворки и сторонние зависимости, не участвующие в ключевых вычислениях;
* Переносимость и простота сборки: проект организован с использованием CMake и может быть легко собран на любой Unix-платформе
* Простота использования и модульность: структура проекта позволяет гибко изменять модель, формат хранения данных, параметры распознавания и визуализации без переписывания основного кода.

## Описание программы

Разрабатываемая программа представляет собой кроссплатформенное консольное приложение на языке **C++**, предназначенное для **детекции и распознавания лиц** в реальном времени с помощью видеопотока с камеры. Программа разработана с упором на минимализм, переносимость и удобство интеграции в более крупные системы. Она использует модели глубокого обучения (предобученные нейросети) в формате ONNX и библиотеку **OpenCV** для работы с видео и нейросетями.

Программа реализует **двухэтапную архитектуру**:

1. **Детекция лица** — определение координат и ключевых точек лица на изображении;
2. **Распознавание лица** — извлечение вектора признаков и поиск наиболее похожего лица в базе.

Программа имеет два основных режима работы, которые выбираются через аргумент --mode

### Функциональные возможности программы

Разработанное приложение обладает следующим функционалом:

1. Детекция лиц на видеопотоке

* Обнаружение лиц в режиме реального времени с использованием модели YuNet.
* Поддержка захвата видеопотока с камеры.
* Вывод координат лиц и ключевых точек (глаза, нос, рот).

2. Извлечение эмбеддинга (вектора признаков)

* Приведение изображения лица к фиксированному размеру (112x112).
* Использование модели MobileFaceNet для генерации 128-мерного вектора, описывающего лицо.
* Нормализация данных и преобразование в формат, пригодный для сравнения.

3. Регистрация лиц

* Режим --mode=register.
* После детекции пользователь вводит имя в консоли.
* Сохраняется пара: имя + вектор признаков в локальную базу face\_db.json.
* База создаётся автоматически, если не существует.

4. Распознавание лиц

* Режим --mode=identify.
* Сравнение текущего вектора с базой по косинусному сходству.
* Отображение "Access allowed" , если найдено совпадение.
* Если лицо не найдено — отображается сообщение "Access denied".

5. Работа с базой данных

* Использование текстового формата JSON для хранения и чтения базы.
* Поддержка нескольких записей: пары - имя + embedding.
* Поиск наиболее похожего пользователя при распознавании.

Программа имеет гибкие параметры входа, указываемые как аргументы командной строки. Имеется возможность указать:

* пути к ONNX-моделям –model –v-model
* путь к базе данных --database
* режима работы программы --mode
* режим для вывода вспомогательного сообщения –help
* установку бекенда для выполнения DNN (по умолчанию — OpenCV) –backend
* установку целевого устройства --target

В проекте применяется модуль cv::dnn из OpenCV — это компонент, предназначенный для работы с предварительно обученными нейросетями. Он позволяет загружать и использовать модели в формате ONNX без необходимости установки и настройки сторонних глубоких фреймворков. Модуль dnn делает проект компактным, кроссплатформенным и пригодным для развёртывания даже на устройствах с ограниченными ресурсами.

## Структура программы

Программа имеет модульную структуру и включает следующие ключевые компоненты:

Модуль инициализации и конфигурации — обрабатывает аргументы командной строки, загружает модель распознавания лиц (YuNet) и векторизации (MobileFaceNet), задаёт параметры confidence, NMS, размер входного изображения и режим работы.

Модуль детекции лица — использует ONNX-модель YuNet через OpenCV DNN API для нахождения координат лиц и ключевых точек на изображении.

Модуль векторизации признаков (Face Embedding) — извлекает область лица, нормализует изображение и подаёт в MobileFaceNet для получения векторного представления лица.

Модуль базы данных — отвечает за хранение и загрузку эмбеддингов лиц из JSON-файла (в формате face\_db.json), поддерживает добавление новых записей и поиск наиболее похожего лица с использованием косинусного сходства.

Модуль визуализации — отображает кадр с прямоугольниками вокруг лиц, ключевыми точками, уровнем confidence, именем найденного пользователя или предложением зарегистрировать новое лицо.

Главный цикл обработки — захватывает изображение с камеры, выполняет обработку.

## Сборка и развёртывание

Сборка осуществляется через Cmake. Поддерживаются два варианта:

* Локальная сборка — с установленной системой OpenCV >= 4.5.4
* Сборка с зависимостями через Git Submodule — проект автоматически подключает OpenCV и MobileFaceNet как внешние модули.

## Методы и критерии распознавания лица

**YuNet** — это компактный и производительный метод детекции лиц, разработанный командой OpenCV и предоставляемый в рамках OpenCV Zoo. Он основан на anchor-free архитектуре нейросетевого детектора и предназначен для высокоскоростной и точной локализации лиц в изображении. YuNet оптимизирован под работу с модулем FaceDetectorYN из библиотеки OpenCV начиная с версии 4.5.4.

YuNet обеспечивает **компромисс между скоростью, размером модели и точностью,** что делает его отличным выбором для встраиваемых систем, мобильных устройств и легковесных приложений — таких, как мой проект.

YuNet является одностадийным (single-shot) anchor-free детектором:

* не используется генерация якорей (anchor boxes);
* модель напрямую предсказывает положение лица, размер и координаты ключевых точек (глаза, нос, уголки рта).

**Особенности архитектуры:**

* Backbone: модифицированный MobileNet;
* Output head: регрессия координат лиц и landmarks;
* Входное изображение: строго фиксированного размера (обычно 320×320 или 480×640);
* Прямой выход: cv::Mat с данными по координатам, landmarks и confidence.

### Выбор метода детекции лица: YuNet

Для решения задачи локализации лиц в изображении была выбрана модель **YuNet**, опубликованная в OpenCV Zoo. YuNet представляет собой компактную, предварительно обученную сверточную нейросеть, оптимизированную под задачу обнаружения лиц с высокой точностью и минимальной задержкой. Среди ключевых причин выбора YuNet можно выделить следующие:

* **Лёгкость и производительность:** YuNet разработана с акцентом на мобильные и встроенные устройства, и показывает высокую производительность даже при низкой частоте кадров;
* **Наличие ONNX-модели:** Возможность загрузки модели в формате ONNX позволяет использовать универсальные интерфейсы cv::dnn, не прибегая к сторонним фреймворкам;
* **Совместимость с OpenCV:** YuNet интегрирована с OpenCV через API cv::FaceDetectorYN, что упрощает разработку и ускоряет прототипирование;

YuNet использует классическую архитектуру backbone с несколькими уровнями сверток и feature map'ов, а также специализированные предсказатели координат, landmarks и confidence scores. Сжатие и упрощение архитектуры при сохранении точности позволили добиться высокой скорости обработки, даже без использования GPU.

В разработанном программном обеспечении реализованы два ключевых этапа: **детекция лица** (определение области изображения, содержащей лицо) и **распознавание личности** (идентификация лица по базе шаблонов). Каждый из этапов использует свои методы и внутренние критерии для принятия решений.

Для детекции лица применяется модель **YuNet**, предоставляемая в составе библиотеки OpenCV начиная с версии 4.5.4. Эта модель использует сверточную нейронную сеть, специально обученную на задаче локализации лиц на изображениях. YuNet возвращает координаты прямоугольных областей, в которых предположительно находятся лица, а также набор ключевых точек (глаз, нос, уголки рта). Чтобы отфильтровать некорректные и низкокачественные предсказания, в программу встроены критерии, позволяющие оценить достоверность и точность результатов.

Первым из таких критериев является **порог уверенности** (confidence threshold). Каждый прямоугольник сопровождается значением вероятности наличия лица; если это значение ниже установленного порога (в нашем случае — 0.9), результат отбрасывается как сомнительный. Таким образом, снижается количество ложноположительных срабатываний и обеспечивается надежность локализации.

Вторым важным критерием является механизм подавления повторяющихся рамок — **Non-Maximum Suppression** (NMS). YuNet может предсказать несколько рамок, перекрывающих одно и то же лицо. Чтобы оставить только наилучшее предсказание, используется коэффициент пересечения (IoU): если степень перекрытия превышает заданный порог (например, 0.3), менее уверенное предсказание удаляется.

Кроме того, предусмотрен параметр **top\_k**, ограничивающий количество прямоугольников, которые будут обрабатываться на этапе NMS. Это позволяет ускорить работу системы, особенно при наличии большого числа лиц в кадре.

### MobileFaceNet

После успешной детекции лица обрабатываются моделью **MobileFaceNet**, которая выполняет векторизацию (feature embedding) — преобразование фрагмента изображения в набор признаков, описывающих лицо в многомерном пространстве. Эти признаки являются числовым представлением индивидуальности лица и позволяют сравнивать изображения между собой с помощью математических метрик.

Для сравнения используется **косинусное сходство** (cosine similarity) между вектором текущего лица и всеми сохраненными векторами из базы данных. Это безразмерная метрика, варьирующаяся от –1 до 1, где 1 соответствует полному совпадению. Если значение сходства превышает установленный порог (в нашей системе он установлен на уровне 0.55), лицо считается идентифицированным, и пользователю отображается имя. В противном случае оно обозначается как «Unknown».

Таким образом, качество работы алгоритма определяется сочетанием детерминированных параметров, таких как порог уверенности, порог подавления пересечений и порог сходства признаков, а также архитектурных особенностей применяемых моделей. Эти критерии позволяют достигать разумного компромисса между точностью и скоростью, что особенно важно при использовании системы в реальном времени.

Для задачи векторизации лиц (получения embedding-представления) и последующей идентификации по векторной близости была выбрана модель **MobileFaceNet**. Это облегчённая архитектура, оптимизированная для задач face verification на мобильных устройствах. Причины выбора MobileFaceNet включают:

* **Высокое качество в компактной модели:** MobileFaceNet достигает сопоставимой точности с более тяжёлыми моделями (например, ArcFace) при значительно меньшем размере модели (~1MB);
* **ONNX-совместимость:** Это позволяет легко интегрировать модель в C++-приложение на базе OpenCV;
* **Устойчивость к изменению условий освещения и поворота головы:** Благодаря обучению на больших датасетах с широким разнообразием примеров.

Архитектурно MobileFaceNet опирается на модифицированную MobileNetv2, с добавлением специализированных блоков Bottleneck, PReLU-активаций и глубокой нормализации для улучшения представления лиц в embedding-пространстве.

Во время проектирования также рассматривались альтернативные подходы:

* Классические методы (Haar Cascades, HOG + SVM): обладают низкой точностью, чувствительны к освещению и поворотам;
* MTCNN и RetinaFace: обладают высокой точностью, но имеют большую вычислительную сложность и не подходят для ARM-архитектур без GPU;
* FaceNet, ArcFace: точны, но требуют больше памяти и времени на инференс.

Выбор YuNet + MobileFaceNet оказался оптимальным с учётом требований по **весу, скорости и удобству интеграции в C++/OpenCV среду**.

## Блок-схема разработанного алгоритма

Схема описывает алгоритм запуска, настройки и основного цикла обработки видеопотока с применением моделей распознавания лиц YuNet и MobileFaceNet. Алгоритм реализован в главной функции программы (main), предназначен для идентификации или регистрации лица с камеры в реальном времени.

После запуска осуществляется парсинг аргументов командной строки, инициализируются модели (FaceDetectorYN, MobileFaceNet), загружается база векторов признаков.

Далее открывается поток с камеры, начинается цикл. Каждый кадр обрабатывается: выполняется детекция лиц, если лицо найдено — оно передаётся в модель для извлечения embedding-вектора. В зависимости от режима (register или identify) происходит либо сохранение вектора в базу, либо сравнение с существующими векторами. В конце каждого шага кадр отображается на экране. Цикл завершается по нажатию ESC.

Изображение выглядит как текст, снимок экрана, чек, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Блок-схема алгоритма визуализации

Схема иллюстрирует логику функции визуализации, отвечающей за отображение кадров с наложенными результатами распознавания, отрисовку рамок, меток и подписание имён.

Функция принимает на вход кадр, список лиц (матрицу координат), режим и путь к модели. Для каждого найденного лица рисуется рамка и landmarks, формируется область ROI. Далее ROI нормализуется и подаётся в MobileFaceNet. В режиме регистрации вызывается ввод имени с клавиатуры и сохранение результата. В режиме идентификации embedding сравнивается с базой, и при успешном распознавании имя отображается на кадре.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Назначение и структура классов

**Класс YuNet**

**Назначение**:  
Инкапсулирует работу с моделью детекции лиц cv::FaceDetectorYN из OpenCV. Предоставляет простой интерфейс для инициализации модели, задания размера входа и получения результатов инференса.

**Зоны ответственности**:

* Загрузка и конфигурация ONNX-модели YuNet;
* Настройка параметров: входной размер, порог уверенности, backend/target;
* Обработка изображения (infer()) и возврат результата — списка лиц и их координат.

**Класс Database**

**Назначение**:  
Реализует хранение и поиск признаков лиц (эмбеддингов), сохранённых в формате JSON.

**Зоны ответственности**:

* Сохранение эмбеддингов с привязкой к имени (saveEmbedding);
* Поиск наиболее похожего эмбеддинга (findMostSimilar);
* Использование косинусного сходства для сравнения признаков;
* Работа с файловой системой и сериализацией/десериализацией JSON.

Изображение выглядит как текст, снимок экрана, визитная карточка, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Тестирование программы

### Цель и задачи тестирования

Целью проведения испытаний является подтверждение соответствия реализованной программной системы требованиям, предъявляемым к функциональности и надёжности программного обеспечения, а также выявление возможных отклонений в логике работы отдельных компонентов.

К основным задачам тестирования относятся:

* Проверка корректности реализации ключевых алгоритмов — детекции лица и сравнения векторов признаков;
* Контроль за стабильностью функционирования программы при типовых сценариях использования;
* Обнаружение возможных пограничных состояний и исключительных ситуаций;
* Подготовка к интеграции системы в продуктивную среду с минимальным уровнем ошибок.

### Объект и виды испытаний

Объектом тестирования является программный комплекс, реализующий идентификацию личности по изображению с использованием моделей **YuNet** и **MobileFaceNet**, реализованный на языке C++ с использованием библиотеки OpenCV.

Для проверки качества работы системы были выбраны следующие виды испытаний:

1. **Модульное тестирование** — направлено на проверку отдельных логических компонентов:
   * загрузка и сохранение JSON-базы (модуль Database);
   * подсчёт косинусного расстояния между векторами признаков;
   * обработка строки команд запуска.
2. **Функциональное тестирование** — проводится в форме "чёрного ящика" с имитацией пользовательских действий:
   * регистрация нового лица;
   * последующая идентификация;
   * поведение при пустой базе или отсутствии видеопотока.

### Условия и средства проведения испытаний

Тестирование проводилось в следующих условиях:

* Аппаратное обеспечение: MacBook с процессором Apple Silicon M4;
* Программное окружение: macOS 15.5, OpenCV 4.13.0 (custom build), CMake 3.29, CLion 2024.1.3;
* Язык программирования: C++17;
* Видеопоток: камера встроенная, разрешение 1280×720;
* Данные: модель face\_detection\_yunet\_2023mar.onnx, векторизатор MobileFaceNet.onnx, база в формате JSON.

Модульные тесты реализованы в виде отдельных C++-функций и вызываются вручную. Функциональное тестирование проводилось вручную путём запуска приложения в разных конфигурациях.

## Вывод

В ходе выполнения практики была разработана и реализована система распознавания лиц в режиме реального времени на языке программирования C++ с использованием библиотеки OpenCV. Основной задачей проекта являлось создание легковесного и модульного приложения, способного выполнять как детекцию, так и идентификацию лиц с камеры, с возможностью регистрации новых пользователей в локальную базу.

В результате работы были изучены и применены современные методы компьютерного зрения. Для задачи обнаружения лица использовалась модель YuNet, а для извлечения признаков (векторизация лиц) — MobileFaceNet, преобразованная в формат ONNX. Была реализована возможность загрузки базы эмбеддингов, их сравнения по косинусному расстоянию и сохранения новых данных. Визуализация осуществлялась в реальном времени с наложением результатов на видеопоток.

Проект обладает чёткой модульной структурой, включающей отдельные компоненты для работы с базой данных, визуализации, обработки аргументов командной строки и взаимодействия с моделями. Был обеспечен минимум внешних зависимостей, а все сторонние библиотеки и модели подключались через механизм CMake и git submodule. Такой подход делает проект удобным для развёртывания и масштабирования.

Также было подготовлено подробное описание алгоритмов, реализованных в проекте, и разработана документация в соответствии с ГОСТ. В частности, построены блок-схемы основного алгоритма и функции визуализации, а также представлены описания структурных компонентов.

Таким образом, поставленные в начале практики цели были достигнуты в полном объёме, а полученный результат представляет собой полнофункциональное приложение для распознавания лиц с открытым кодом и прозрачной архитектурой.

## ****Список использованных источников****

* [1] Yu, Shiqi. "YuNet: A lightweight face detector." GitHub, 2023. <https://github.com/opencv/opencv_zoo/tree/main/models/face_detection_yunet>
* [2] Deng, Jiankang et al. "MobileFaceNets: Efficient CNNs for accurate real-time face verification on mobile devices." arXiv preprint arXiv:1804.07573 (2018).
* [3] OpenCV Documentation. <https://docs.opencv.org>

|  |  |  |
| --- | --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | / Григорьева Е.В. | /\_\_\_ \_\_\_\_\_\_\_\_ 20\_\_г. |
| *подпись обучающегося* | *расшифровка подписи* | *дата* |

## Приложение а. Структура проекта

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.

## Приложение б. Код проекта

### Main.cpp

#include "YuNet.cpp"  
#include "database.cpp"  
#include "face\_detection.h"  
  
int main(int argc, char \*\*argv) {  
 cv::CommandLineParser parser(  
 argc, argv,  
 "{help h | | Print this "  
 "message}"  
 "{database db | face\_db.json | Set path "  
 "to the database}"  
 "{model m | face\_detection\_yunet\_2023mar.onnx | Set path to "  
 "the model}"  
 "{v-model vm | MobileFaceNet.onnx | Set path to the "  
 "model for vectoring}"  
 "{mode | identify | Mode: register "  
 "or identify}"  
 "{backend b | opencv | Set DNN "  
 "backend}"  
 "{target t | cpu | Set DNN target}"  
 */\* model params below\*/* "{conf\_threshold | 0.9 | Set the "  
 "minimum confidence for "  
 "the model to identify a face. Filter out faces of conf < conf\_threshold}"  
 "{nms\_threshold | 0.3 | Set the "  
 "threshold to suppress "  
 "overlapped boxes. Suppress boxes if IoU(box1, box2) >= nms\_threshold, "  
 "the one of higher "  
 "score is kept.}"  
 "{top\_k | 5000 | Keep top\_k "  
 "bounding boxes before "  
 "NMS. Set a lower value may help speed up postprocessing.}");  
 if (parser.has("help")) {  
 parser.printMessage();  
 return 0;  
 }  
  
 Database db(parser.get<std::string>("database"));  
 std::string model\_path = parser.get<std::string>("model");  
 std::string vmodel\_path = parser.get<std::string>("v-model");  
 std::string backend = parser.get<std::string>("backend");  
 std::string target = parser.get<std::string>("target");  
 std::string mode = parser.get<std::string>("mode");  
 std::cout << "Mode:" << mode << std::endl;  
  
 *// model params* float conf\_threshold = parser.get<float>("conf\_threshold");  
 float nms\_threshold = parser.get<float>("nms\_threshold");  
 int top\_k = parser.get<int>("top\_k");  
 const int backend\_id = str2backend.at(backend);  
 const int target\_id = str2target.at(target);  
  
 *// Instantiate YuNet* YuNet model(model\_path, cv::Size(320, 320), conf\_threshold, nms\_threshold,  
 top\_k, backend\_id, target\_id);  
  
 int device\_id = 0;  
 auto cap = cv::VideoCapture(device\_id);  
 if (!cap.isOpened()) {  
 std::cerr << "Error: Cannot open camera with device\_id " << device\_id  
 << "\n";  
 return -1;  
 }  
 int w = static\_cast<int>(cap.get(cv::*CAP\_PROP\_FRAME\_WIDTH*));  
 int h = static\_cast<int>(cap.get(cv::*CAP\_PROP\_FRAME\_HEIGHT*));  
 model.setInputSize(cv::Size(w, h));  
  
 auto tick\_meter = cv::TickMeter();  
 cv::Mat frame;  
  
 while (cv::waitKey(1) < 0) {  
  
 bool has\_frame = cap.read(frame);  
  
 if (!has\_frame) {  
 std::cout << "No frames grabbed! Exiting ...\n";  
 break;  
 }  
  
 *// Inference* tick\_meter.start();  
 cv::Mat faces = model.infer(frame);  
 tick\_meter.stop();  
  
 *// Draw results on the input image* auto res\_image = visualize(vmodel\_path, frame, faces, mode, db,  
 (float)tick\_meter.getFPS());  
  
 *// Visualize in a new window* cv::imshow("Face detection for Avrora", res\_image);  
  
 tick\_meter.reset();  
 }  
  
 return 0;  
}

### Visualizer.cpp

#include "face\_detection.h"  
  
cv::Mat visualize(const std::string &vmodel, cv::Mat &image,  
 const cv::Mat &faces, const std::string &mode, Database &db,  
 float fps) {  
 static cv::Scalar box\_color{0, 255, 0};  
 static std::vector<cv::Scalar> landmark\_color**{** cv::Scalar(255, 0, 0), *// right eye* cv::Scalar(0, 0, 255), *// left eye* cv::Scalar(0, 255, 0), *// nose tip* cv::Scalar(255, 0, 255), *// right mouth corner* cv::Scalar(0, 255, 255) *// left mouth corner* **}**;  
 static cv::Scalar text\_color{0, 255, 0};  
  
 cv::Mat output\_image = image.clone();  
  
 if (fps >= 0) {  
 cv::putText(output\_image, cv::format("FPS: %.2f", fps), cv::Point(0, 15),  
 cv::*FONT\_HERSHEY\_SIMPLEX*, 0.5, text\_color, 2);  
 }  
  
 for (int i = 0; i < faces.rows; ++i) {  
 *// Draw bounding boxes* int x1 = static\_cast<int>(faces.at<float>(i, 0));  
 int y1 = static\_cast<int>(faces.at<float>(i, 1));  
 int w = static\_cast<int>(faces.at<float>(i, 2));  
 int h = static\_cast<int>(faces.at<float>(i, 3));  
 cv::rectangle(output\_image, cv::Rect(x1, y1, w, h), box\_color, 2);  
  
 *// Confidence as text* float conf = faces.at<float>(i, 14);  
 cv::putText(output\_image, cv::format("%.4f", conf), cv::Point(x1, y1 + 12),  
 cv::*FONT\_HERSHEY\_DUPLEX*, 0.5, text\_color);  
  
 *// Draw landmarks* for (int j = 0; j < landmark\_color.size(); ++j) {  
 int x = static\_cast<int>(faces.at<float>(i, 2 \* j + 4)),  
 y = static\_cast<int>(faces.at<float>(i, 2 \* j + 5));  
 cv::circle(output\_image, cv::Point(x, y), 2, landmark\_color[j], 2);  
 }  
  
 *// ROI with size: 112x112* cv::Rect rect(x1, y1, w, h);  
 rect &= cv::Rect(0, 0, output\_image.cols, output\_image.rows); *// ограничение* if (rect.width <= 0 || rect.height <= 0)  
 continue;  
  
 cv::Mat faceROI = output\_image(rect).clone();  
 cv::resize(faceROI, faceROI, cv::Size(112, 112));  
  
 *// Make blob for MobileFaceNet* cv::Mat blob =  
 cv::dnn::blobFromImage(faceROI, 1.0 / 128.0, cv::Size(112, 112),  
 cv::Scalar(127.5, 127.5, 127.5), true, false);  
  
 *// recognize from MobileFaceNet* static cv::dnn::Net recognizer;  
 static bool net\_loaded = false;  
 if (!net\_loaded) {  
 try {  
 recognizer = cv::dnn::readNet(vmodel);  
 net\_loaded = true;  
 } catch (const cv::Exception &e) {  
 std::cerr << "Warning: Could not load model: " << e.what() << std::endl;  
 *// Для теста можно вернуть image без изменений* return output\_image;  
 }  
 }  
 recognizer.setInput(blob);  
 cv::Mat embedding = recognizer.forward();  
  
 if (mode == "register") {  
 std::string name;  
 std::cout << "Введите имя: ";  
 std::cin >> name;  
 db.saveEmbedding(name, embedding);  
 } else if (mode == "identify") {  
 *// bool match = db.findMostSimilar(embedding);* if (db.findMostSimilar(embedding)) {  
 *// std::cout << "Access allowed\n";* cv::putText(output\_image, "Access allowed", cv::Point(x1, y1 - 5),  
 cv::*FONT\_HERSHEY\_SIMPLEX*, 0.6, text\_color, 2);  
 } else {  
 *// std::cout << "Access denied\n";* cv::putText(output\_image, "Access denied", cv::Point(x1, y1 - 5),  
 cv::*FONT\_HERSHEY\_SIMPLEX*, 0.6, text\_color, 2);  
 }  
 }  
 }  
 return output\_image;  
}

## Приложение с. Результаты тестов

Изображение выглядит как текст, снимок экрана, меню

Содержимое, созданное искусственным интеллектом, может быть неверным.