Face.SDK

Набор программных средств для разработки систем идентификации по изображению лица

Версия 5.x.x

Руководство разработчика

STC innovations

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Общие положения

В руководстве указана последовательность действий пользователя, обеспечивающих работу с диктофонной станций Гном-Сити и её составными частями, а также программой управления диктофонами.

Руководство предназначено для разработчиков ПО, использующих **Face.SDK** в повседневном процессе разработки.

Руководство предоставляет общий обзор возможностей системы. Для получения более подробной информации ознакомьтесь с документацией HTML, которая находится в папке doc дистрибутива SDK (**doc/html/index.html**). Страницы руководства для Linux также доступны в папке doc.

Настоящий документ не заменяет учебную, справочную литературу, руководства от производителей операционной системы, освещающие работу с их графическим пользовательским интерфейсом.

Руководство относится к **Face.SDK** версии 5.x.x.

Соглашения и обозначения

В руководстве приняты следующие типографские соглашения:

Формат	Значение
Обычный	Основной текст документа.
Курсив	Применяется для выделения первого появления *термина*, значение которого поясняется здесь же или даётся в приложении. Также применяется для привлечения *внимания* и оформления *примечаний*.
Полужирный	Применяется для написания наименований **программных продуктов**, **компонентов**, **управляющих** и **информационных элементов** интерфейса (**заголовки**, **кнопки** и т.п.).
Полужирный курсив	Применяется для написания ***имён файлов*** и ***путей доступа*** к ним.

Словосочетания «нажать, выбрать объект», «коснуться объекта» означают: «прикоснуться пальцем к экрану мобильного устройства в том месте, где расположен данный объект».

Выбор меню может быть показан при помощи стрелки **>**, например, текст **Файл > Выход**, должен пониматься так: выбрать меню **Файл**, затем команду **Выход** из меню **Файл**.

Ниже приведены примеры оформления материала руководства, указывающие на важность сведений.

Ссылки на другие документы в основном тексте.

Примечания; важные сведения; указания на действия, которые необходимо выполнить в обязательном порядке.

Требования, несоблюдение которых может привести к некорректной работе, повреждению или выходу из строя изделий или программного обеспечения.

ОБЩИЕ СВЕДЕНИЯ

Сведения о продукте и изготовителе

Наименование: Система разработки ПО для распознавания лиц **Face.SDK**

Изготовитель: Общество с ограниченной ответственностью «ЦРТ-инновации»

Адрес: 194044, г. Санкт-Петербург, ул. Гельсингфорсская, д. 3, корпус 11, литер Д, помещение 229

Телефон: (812) 325-88-48

Факс: (812) 327-92-97

Служба технической поддержки

Адрес службы сервисного обслуживания и технической поддержки в Интернете:

Электронная почта: support@speechpro.com

Веб-сайт: http://www.speechpro.ru/

При обращении в службу технической поддержки необходимо представить четкое описание возникшей проблемы.

Предварительно подготовьте следующую информацию:

– Номер версии и конфигурация системы;
– Конфигурация оборудования;
– наименование и версия используемой операционной системы Windows;
– видеозапись возникшей проблемы (при наличии);
– проблемный файл биометрического шаблона (при наличии);
– сообщение об исключении SDK.

Контакты

Россия, Санкт-Петербург
Адрес: 194044, ул. Гельсингфорсская, д. 3, корпус 11, литер Д, помещение 229
Телефон: (812) 325-88-48
Факс: (812) 327-92-97
Электронная почта: support@speechpro.com

НАЗНАЧЕНИЕ И ВОЗМОЖНОСТИ

Назначение

Система разработки ПО для распознавания лиц **Face.SDK** предназначена для создания биометрических шаблонов на основе биометрических характеристик и последующего биометрического распознавания с помощью созданных шаблонов.

Face.SDK распространяется в виде библиотеки динамической компоновки с минимумом дополнительных зависимостей, которая содержит все функции, необходимые для интеграции сервисов биометрической идентификации и проверки в другие программные продукты.

Возможности

Face.SDK содержит несколько модулей с разным функционалом:

• Основные понятия - базовая информация об алгоритмах, структурах и контейнерах **Face.SDK** (см. Раздел 6).

• Обнаружение - ориентировано на обнаружение лиц человека в анфас (см. Раздел 7).

• Отслеживание - позволяет отслеживать лица в видеопотоке (см. Раздел 8).

• Определение ключевых точек - определение 68 ключевых точек лица (см. Раздел 9).

• Распознавание - технология биометрического распознавания, основанная на методах глубокого обучения (см. Раздел 11).

• Портретные характеристики - некоторые популярные характеристики лица, такие как резкость, освещение, геометрические свойства и другие (см. Раздел 11).

• Верификация живого человека - электронный инструмент или метод, который проверяет, принадлежит ли взятый биометрический образец живому человеку. Такая технология помогает предотвратить т.н. «спуфинг», например использование ранее сделанных фотоизображений, голосовых записей или видеофайлов для проверки (см. Раздел 12).

ТРЕБОВАНИЯ К ОБЕСПЕЧЕНИЮ

Требования к программным средствам

Face.SDK поддерживает следующие операционные системы (только 64-разрядные):

- Windows 10, Server 2016;
- CentOS 7;
- Ubuntu 18.04.

Обратите внимание, что мы поставляем 64-разрядные сборки **Face.SDK**, поэтому их следует использовать только в 64-битных приложениях.

Требования к ОС Windows:

- Microsoft Visual Studio 2013 SP1 ч64 или более новые версии;
- Microsoft Visual C++ Redistributable for Visual Studio 2017 x64.

Обратите внимание, что для максимальной производительности **Face.SDK** в Windows должен быть установлен режим **Высокая производительность**. Дополнительные сведения см. в официальной документации Microsoft Windows: https://docs.microsoft.com/en-us/windows/desktop/power/power-policy-settings

Требования к ОС Linux:

- Набор компиляторов GCC 4.8 (или более новая версия);
- Для CentOS: пакеты glibc.i686, initscripts, libgomp;
- Для Ubuntu: пакеты libc6-i386, ffmpeg, libgtk2.0.

Требования к техническим средствам

Для установки **Face.SDK**, ваше оборудование должно соответствовать следующим параметрам:

Процессор: Intel® Core™ x64/Intel® Xeon® x64 с поддержкой набора команд AVX2;

- Оперативная память: не менее 8 GB;
- Жесткий диск: не менее 10 GB;
- USB порт (для локального ключа HASP);
- Ethernet соединение (если используется ключ Net HASP).

Face.SDK тестировался на следующих процессорах:

– Intel® Core™ i7-9700;

– Intel® Xeon® Gold 5215.

Если у вас возникли трудности с запуском **Face.SDK** на других 64-разрядных системах Intel® Core ™/Xeon®, обратитесь в службу технической поддержки.

Требования к системной памяти различаются в зависимости от сценария использования. Мы рекомендуем выделять не менее 8 ГБ на каждый экземпляр **Face.SDK** для типичных сценариев.

Требования для GPU-ускорения

Поддержка **Face.SDK**:

• Для сборок с поддержкой CUDA 10.1 поддерживаются системы графических процессоров Windows и Linux с 3,0, 3,5, 5,0, 6,0, 6,1, 7,0, 7,5 (Kepler, Maxwell, Pascal, Volta, Turing). Библиотеки и двоичные файлы расположены в каталогах с суффиксом «cuda», например: CentOS7_x86_64_gcc_cuda10.1. Должны быть установлены драйверы, совместимые с CUDA 10.1.

• Сборки без поддержки CUDA (если имеются) доступны для Windows и Linux. Библиотеки и двоичные файлы расположены в каталогах с суффиксом «nocuda», например: CentOS7_x86_64_gcc_nocuda.

Требования к входным данным

Кодировка строк

Face.SDK не поддерживает символы Unicode. Для уверенности в правильном результате используйте только символы ASCII в любых строках, которые вы передаете **Face.SDK**.

Формат изображений

Поддерживаются следующие форматы файлов изображений:

- Форматы Windows — ***.bmp***;
- файлы JPEG — ***.jpeg***, ***.jpg***;
- Portable Network Graphics — ***.png***;
- форматы ***.pbm***, ***.pgm***, ***.ppm***;
- форматы SUN— ***.sr***, ***.ras***;
- файлы TIFF — ***.tiff***, ***.tif***.

Формат видео

Поддерживаются следующие форматы видеофайлов:

- CentOS — ***.MJPEG**;
- Ubuntu — все форматы, поддерживаемые ***ffmpeg***, установленные в ОС (см. Требования к программным средствам);
- Windows — все форматы, поддерживаемые FFmpeg 3.x;
- файлы QuickTime— ***.mov**.

Изображение лица

Обнаружение

Минимальное разрешение лица на изображении составляет примерно 35 пикселей на ширину лица. Для лучшего качества обнаружения рекомендуется не менее 60 пикселей. Хотя рекомендовано изображение лица в анфас, допустимы и некоторые отклонения. Отклонения под любым углом (из вертикального обзора или по крену, вверх/вниз или по тангажу, по горизонтали или по рысканию) от

позиции в анфас должно быть не более 70 градусов. Различная маскировка лица может значительно повлиять на качество обнаружения.

Распознавание

Минимальное разрешение лица на изображении для распознавания лица составляет минимум 60 пискелей на ширину лица. В случае если условия освещения контролируемы, соблюдайте следующие трубования:

- избегайте теней в области лица;
- избегайте бликов на очках или блеска кожи;
- избегайте солнечного света;
- условия освещения должны быть аналогичны условиям, при которых проводились этапы регистрации и проверки.

ОПИСАНИЕ ДИСТРИБУЦИИ SDK

Дистрибутив **Face.SDK** содерит следующие подпапки:

- bin — бинарные сборки и тестовые утилиты;
- include — файлы SDK C++ API;
- doc — документация SDK;
- lib — бинарные библиотеки, которые можно использовать для ссылок на SDK;
- java — JNI для SDK;
- SDK_data — данные для инициализации алгоритмов.

УСТАНОВКА И КОМПОНОВКА

Установка

Для установки **Face.SDK** распакуйте архив в любую папку и следуйте указанным ниже инструкциям.

ОС Linux

Для установки **Face.SDK** на Linux, выполните следующие шаги:

- Установите драйвер HASP с сайта http://www.safenet-inc.com/.

ОС Windows

Для установки **Face.SDK** на Windows, выполните следующие шаги:

- Установите драйвер HASP с сайта http://www.safenet-inc.com/.
- Скачайте и установите Microsoft Visual C++ 2013 Redistributable Package с сайта Microsoft.

Компоновка

C++ API

Для компиляции и компоновки приложений при помощи **Face.SDK** C ++ API необходимо выполнить следующие действия:

- добавить подкаталог include каталога установки **Face.SDK** в путь поиска include компилятора;
- необходимо добавить подкаталог lib в путь поиска компоновщика установочного каталога **Face.SDK**;
- ввести подходящую библиотеку в компоновщик.

Java API

Для компиляции и компоновки приложений при помощи **Face.SDK** Java API, необходимо выполнить следующие действия:

- добавить файл Face.SDK-jni-<version>.jar или подпапку в соответутсвующую библиотку вашего проекта;
- добавить файлы Face.SDK-jni-<version>-javadoc.jar и Face.SDK-jni-<version>-sources.jar в проект или ИСР чтобы сделать доступными JavaDocs и Sources;
- добавить путь к подкаталогу lib, содержащему библиотеку Face.SDK-jni, к пути к библиотеке Java. Это означает добавление Djava.library.path = "path_to_directory_with_Face.SDK-jni_library" в параметры JVM.

ОСНОВНЫЕ ПОНЯТИЯ

Базовые конструкции

Face.SDK содержит несколько полезных конструкций для обработки и передачи данных, таких как:

Таблица 2: Базовые конструкции **Face.SDK**

C++	Java	Описание
FSDK::core_types::PointF	**com.speechpro.fsdk.coretypes.PointF**	(x,y) точка
FSDK::core_types::Area	**com.speechpro.fsdk.coretypes.Area**	(x,y,w,h) область
FSDK::containers::ByteArray	byte[]	массив байтов
FSDK::containers::Image	**com.speechpro.fsdk.containers.Image**	контейнер изображений
FSDK::containers::FVector	массивы Java	свертка векторов
FSDK::containers::FIR	**com.speechpro.fsdk.containers.FIR**	контейнер FIR
FSDK::containers::Decision	**com.speechpro.fsdk.containers.Decision**	контейнер для хранения решений алгоритмов
FSDK::containers::IdentificationResult	**com.speechpro.fsdk.containers.IdentificationResult**	контейнер результатов идентификации
FSDK::containers::EyesPosition	**com.speechpro.fsdk.containers.EyesPosition**	положение глаз на лице
FSDK::containers::DetectionResult	**com.speechpro.fsdk.containers.DetectionResult**	положение лица на изображении
FSDK::containers::FaceInfo	**com.speechpro.fsdk.containers.FaceInfo**	результат отслеживания лица
FSDK::containers::VideoSource	**com.speechpro.fsdk.containers.VideoSource**	кадры видеопотока

Значения всех основанных на координатах структур **Face.SDK** приведены в декартовой системе координат Oxy с началом O = (0, 0) в верхнем левом углу изображения и осями Ox и Oy, направленными от O к верхнему правому и ограниченному левому углам изображения соответственно. Единицы измерения Ox и Oy - это ширина и высота пикселя изображения соответственно.

Класс **FSDK::container::VideoSource** является экспериментальным, поэтому не используйте его в производственной среде (ни в Python, ни в Java API).

Одна из наиболее важных структур - это **FSDK::configuration::FSDKProfile**. Эта структура служит для инициализации всех остальных компонентов SDK. **FSDKProfile** основанный на профилях, находится в **SDK_data/profiles**. Каждый профиль наследует свои параметры от профиля по умолчанию. Если один параметр определен внутри дочернего профиля, этот параметр перезаписывается.

Типичная схема работы SDK

Типичная схема работы SDK приведена на рис. 1. Она состоит из:

- этапа нахождения лица,
- этапа нахождения ключевых точек лица ,
- этап проверки или идентификации,
- выборочно: этап проверки лица живого человека,
- выборочно: определение портретных характеристик;
- этапа оценки. Примечания к обозначениям на изображении ниже:
- блока обозначающего данные (источники или результаты),
- оранжевых блоков, обозначающих внешние данные,
- зеленых блоков, обозначающих значимые для пользователя результаты,
- эллипсов, обозначающих алгоритмы,
- если от узла есть несколько входящих стрелок, это означает, что есть несколько способов получить этот результат, и входы для каждого из них имеют одинаковый цвет (зеленый/оранжевый/синий /…).

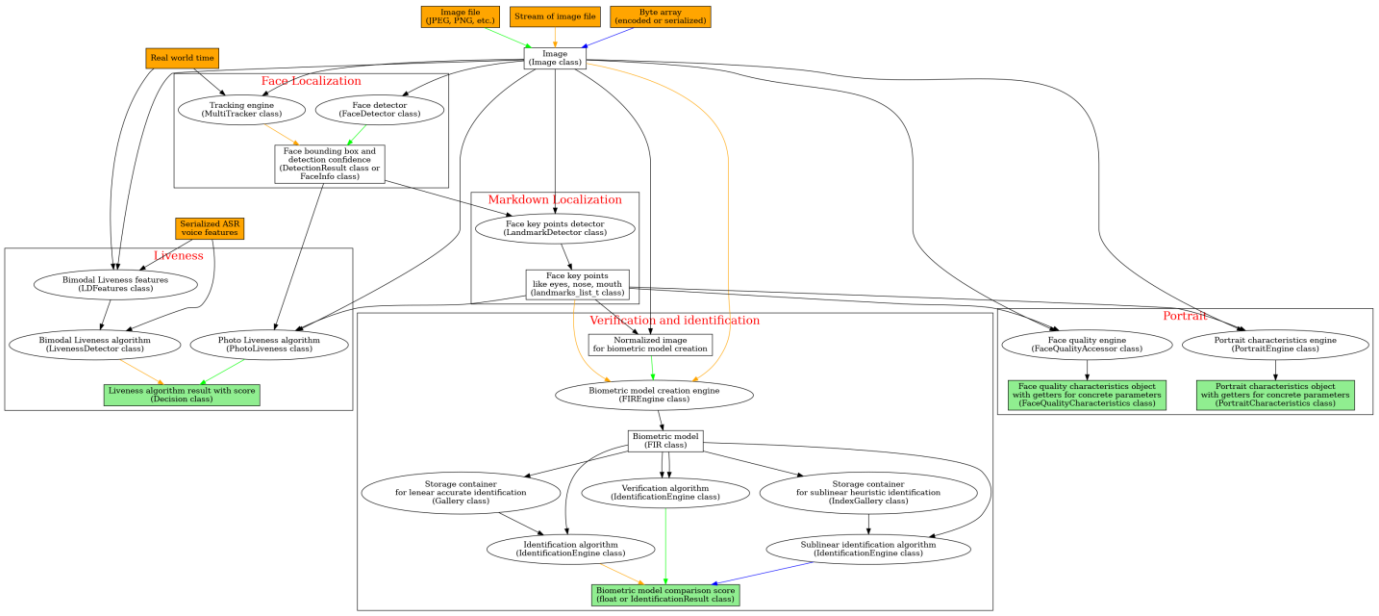

Рисунок 1: Типичная схема использования SDK

Движки для нейронных сетей

Многие алгоритмы **Face.SDK** основаны на нейронных сетях. Существуют три движка для подобных алгоритмов:

- caffe;
- opencv;
- pt.

Движки Caffe и opencv взаимозаменяемы, это означает, что вы можете менять их в профилях в соответствии с вашими требованиями к производительности (см. раздел 13). Движок pt несовместим с сетями для движков caffe и opencv.

Сети IR_SE_38_256 и face_pixels_segmentator нельзя использовать на Android x86, вероятно, из-за проблем с опреативной памятью.

Примеры использования

Инициализация SDK

Инициализация SDK выполняется вызовом **FSDK::configuration::init_fsdk**. Вы должны выполнить инициализацию до начала работы с **Face.SDK**.

```
bool is_initialized = FSDK::configuration::init_fsdk();
```

Список 1: C++: инициализация Face.SDK

```
boolean isInitialized = Initialization.init_fsdk();
```

Список 2: Java: инициализация Face.SDK

Инициализация профиля

```
const FSDK::configuration::FSDKProfile profile("path_to_SDK_data_folder",
    "profile_name(default_by_default)");
```

Список 3: C++: инициализация профиля

```
try (FSDKProfile profile = FSDKProfile.createFSDKProfile("path_to_SDK_data_folder",
    "profile_name")) {
}
```

Список 4: Java: инициализация профиля

Загрузка изображений

Изображение может быть загружено по имени файла, из **FSDK::container::ByteArray**, из закодированных данных изображения или из необработанных данных char*.

Обратите внимание, что com.speechpro.fsdk.containers.Image реализует java.lang.AutoCloseable интерфейс, поэтому мы рекомендуем использовать оператор try-with-resources.

```
const FSDK::containers::Image img1("path_to_image_file");                // считать
    изображение с пути

const FSDK::core_types::image_pixel_t* data1_ = img1.get_image_data(); // нет копии,
    укажите данные изображения

FSDK::core_types::image_pixel_t* data2_ =
    new FSDK::core_types::image_pixel_t[img1.height() * img1.width()]{0}; // пиксельные
        данные
const FSDK::containers::Image img2(data2_, img1.height(),
                                   img1.width());                // по умолчанию новое
                                       изображение создает копию data2_
const FSDK::containers::ByteArray bytearray = img2.serialize(); // теперь изображение
    можно сериализовать
FSDK::containers::Image img3;
img3.load(bytearray); // и создать новое изображение из сериализованных
данных
size_t length;
const char* data3_ = img3.get_stream_data(length); // получить jpg-кодированные данные
из изображения
FSDK::containers::Image img4;
img4.load_from_stream_encoded_data(data3_, length); // и создать изображение
    из jpg-кодированных данных
```

Список 5: C++: использование контейнеров

```
byte[] data1_;
int height, width;
try (Image img1 = Image.createImage("path_to_image_file", keepColorImage)) { //
    считать изображение с пути
    data1_ = img1.getImageData(); // получить данные, этот метод копирует
    данные
    height = img1.height();
    width = img1.width();
}
byte[] serialized_img2;
try (Image img2 = Image
        .createImage(data1_, height, width, ColorType.GRAY, false, false)) { // этот
            метод скопирует данные
    serialized_img2 = img2.serialize(); // изображение можно сереализовать в формат Face.SDK
}
try (Image img3 = Image.createImage()) {
    img3.load(serialized_img2); // и создать новое изображение из сериализованных данных
}
try {
    byte[] jpegEncodedData = Files.readAllBytes(
            FileSystems.getDefault().getPath("image.jpg")); // можно считать
                закодированное изображение в байты
    try (Image img4 = Image.createImage()) {
        img4.loadFromEncodedData(jpegEncodedData); // и загрузить изображение из этих
        байтов
    }
```

Список 6: Java: использование контейнеров

Обработка ошибок

Face.SDK использует **FSDK::exceptions::FException** для передачи сообщений об ошибках. Используйте блок try-catch для их обработки:

```
catch (FSDK::exceptions::FException& e) {
    std::cerr << e.what() << std::endl;
}
```

Listing 7: C++: блок catch

```
} catch (FSDKNativeException e) {
    e.printStackTrace();
}
```

Список 8: Java: блок catch

Завершение работы SDK

Завершение работы SDK производится с помощью вызова **FSDK::configuration::terminate_fsdk**:

```
bool is_terminated = FSDK::configuration::terminate_fsdk();
```

Listing 9: C++: завершение работы

```
boolean isTerminated = Initialization.terminate_fsdk();
```

Listing 10: Java: завершение работы

Не забудьте очистить память, удалив все используемые объекты, перед завершением работы SDK.

ОБНАРУЖЕНИЕ

Краткий обзор

Модуль, предназначенный для обнаружения лица на изображениях. Поддерживаемые типы детекторов:

- каскадные;
- основанные на CNN.

Алгоритм может обнаруживать лица со следующими параметрами:

- отклонение под любым углом от позиции в анфас должно быть не менее 70 градусов;
- минимальная ширина лица – не менее 35 пикселей.

Для детектора на основе CNN минимальная ширина лица 35 пикселей предназначена для изображения размером 512×512 или 256×256 (в зависимости от алгоритма детектора). Далее описывается процесс преобразования исходного изображения в размер 512×512 (256×256). Итак, сначала, если *соотношение сторон* изображения больше, чем определено в профиле в разделе детектора (параметр **add_border_max_ratio**), к изображению добавляются границы, чтобы получилось вышеуказанное соотношение сторон. Затем оно масштабируется до размера 512×512 (256×256). Таким образом можно понять, как рассчитывается «реальная» минимальная ширина лица на входном изображении для детектора на основе CNN.

Для улучшения качества обнаружения рекомендуется на менее 60 пикселей. Рекомендуется вид лица в анфас, но некоторые отклонения допустимы. Различное маскирование лица может значительно повлиять на качество обнаружения.

Каскадные алгоритмы детекторов основаны на:

- особенностях типа Haar и LBP;
- каскадном классификаторе:
- могут быть созданы меньшие и, таким образом, более эффективные классификаторы, которые отклонят множество негативных подокон, в то время как почти все позитивные экземпляры будут обнаружены;
- упрощенные классификаторы используются для отклонения большинства подокон;
- более сложные классификаторы используются, чтобы достичь низкого уровня ложных срабатываний;
- повышении качества выбора функций.

Алгоритмы CNN основаны на подходе одноразового обнаружения. Так же как в алгоритмах yolo или ssd, только один проход необходим для верного срабатывания обнаружения. Мы используем каскадные сверточные нейронные сети для обнаружения лиц.

Существует максимальное разрешение изображения, с которым может справиться детектор. Оно зависит от типа детектора:

- для каскадного детектора - это 4K (3840 × 2160). Использовать изображения с более высоким разрешением допустимо, но может привести к фатальным ошибкам;
- для детектора основанного на CNN - это 8K (7680×4320). Проводились тесты и для изображений большего размера, так что есть потенциал использования изображений до 16K без потери производительности.

Алгоритмы распознавания лиц могут обнаружить изображения лиц на целом изображении. Термин «изображение лица» может относиться к настоящему лицу, заснятому на камеру, так же как и к портретам, распечаткам и другим изображениям лица (некоторые животные, имеющие сходные черты лица могут быть обнаружены детекторами как изображения лица). Чтобы определить принадлежит ли изображение лица реальному человеку, необходимо использовать алгоритмы проверки лица живого человека.

Существует проблема с детектором Cascade на некоторых процессорах Intel, связанная со сломанным драйвером OpenCL. Если вы столкнулись с данной проблемой, замените метку детектора в разделе facedetector вашего профиля следующим XML-кодом:

```
<detector>
<type>simd</type>
<id>cascade_m25_25_var_light_35x42</id>
<cascadename>MqhyD.fsdk</cascadename>
<scale>1.16</scale>
<neighbors>6</neighbors>
<minwindowsize>40</minwindowsize>
<maxwindowsize>0.99</maxwindowsize>
<isextended>false</isextended>
<nthreads>2</nthreads>
</detector>
```

Примеры использования

Если какие-либо параметры **FSDK::detection::FaceDetector::detect()** используются по умолчанию, то они загружаются из профиля.

```
FSDK::detection::FaceDetector face_detector(profile, device_idx);
FSDK::containers::detection_list_t faces = face_detector.detect(img1, -1, -1,
    FSDK::core_types::Area());
```

Список 11: C++: детектор лиц

```
try (FaceDetector faceDetector = FaceDetector.createFaceDetector(profile)) {
    DetectionResult[] faces = faceDetector.detect(image, -1, -1, new Area());
}
```

Список 12: Java: детектор лиц

Точность

Результаты представлены для следующих бэкендов:

• для Cascade и SSD расчеты производятся на бэкэнде Caffe;

• Расчеты для SSD-Retina, SSD-Retina-quantized, SSD-Retina-quantized-mobile и SSD-Retina-ShuffleNet выполняются на бэкэнде pt.

и в этих наборах данных:

• WiderFace

• FDDB

• Подвыборка FDDB, которая представляет собой набор данных FDDB, закрытый для контролируемого случая. Он содержит 3508 лиц на 2446 фотографиях. Все лица имеют размер больше 10% размера фотографии и малые углы поворота.

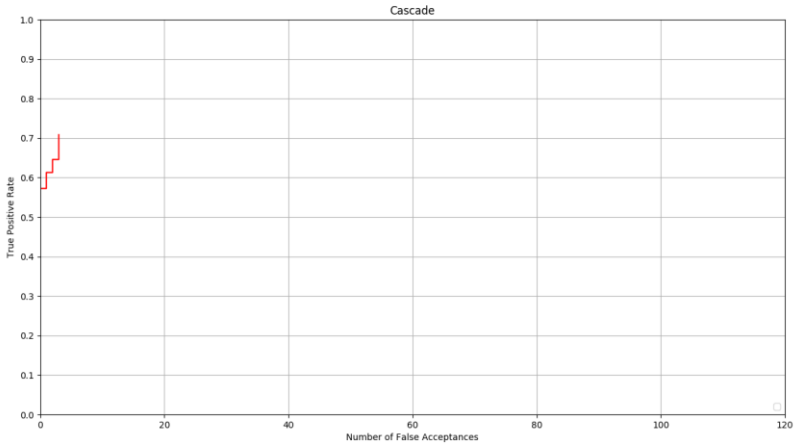

Рисунок 2: *Дискретный тест FDDB для детектора Cascade*

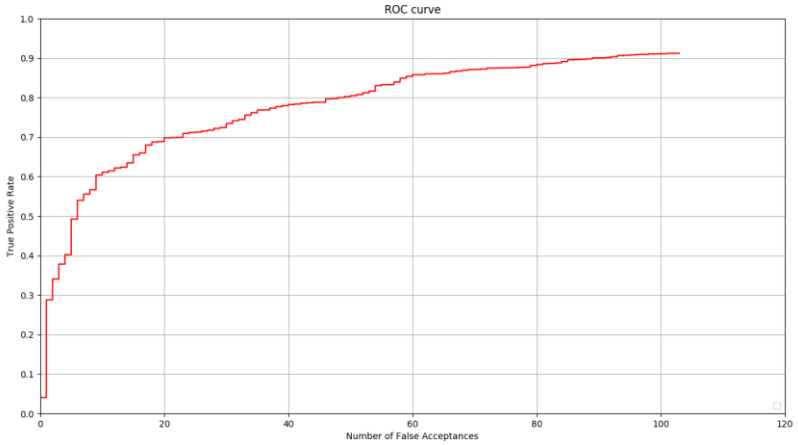

Рисунок 3: *Дискретный тест FDDB для детектора SSD*

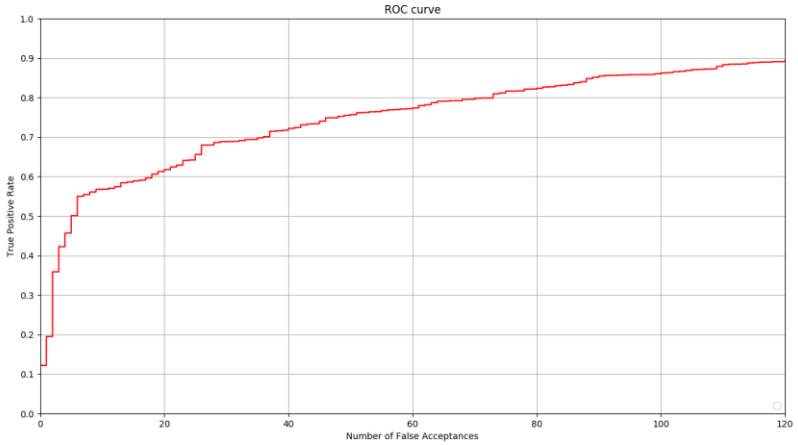

Рисунок 4: *Дискретный тест FDDB для детектора SSD-Retina*

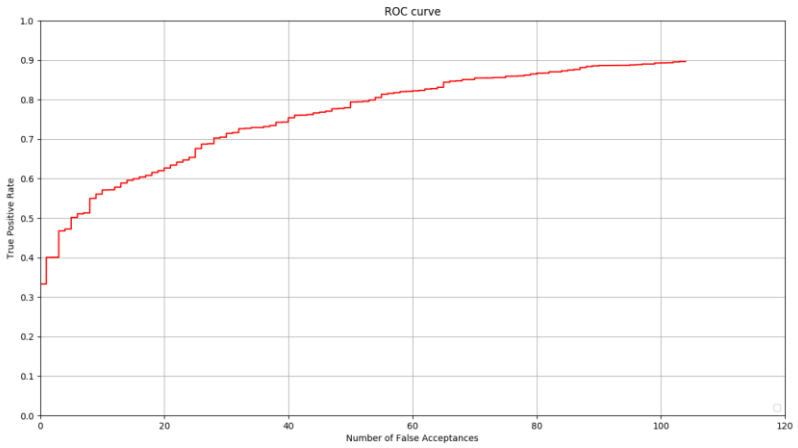

Рисунок 5: *Дискретный тест FDDB для детектора SSD-Retina-quantized*

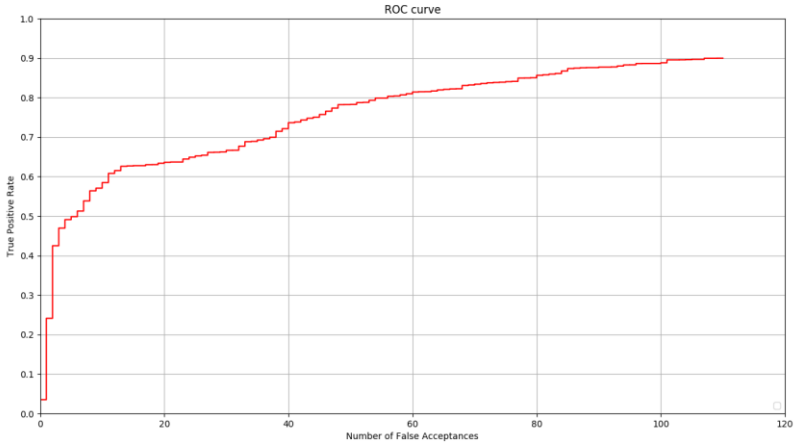

Рисунок 6: FDDB discrete test for CNN based detector **SSD-Retina-quantized-mobile**

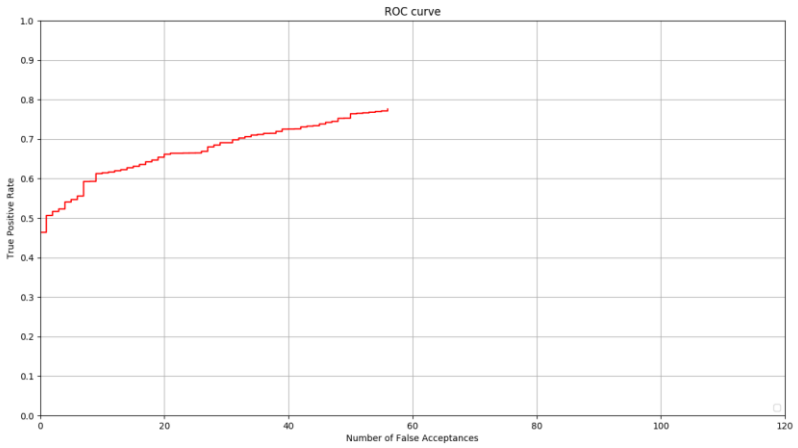

Рисунок 7: FDDB discrete test for CNN based detector **SSD-Retina-ShuffleNet**

Точность на более обширном контрольном тесте:

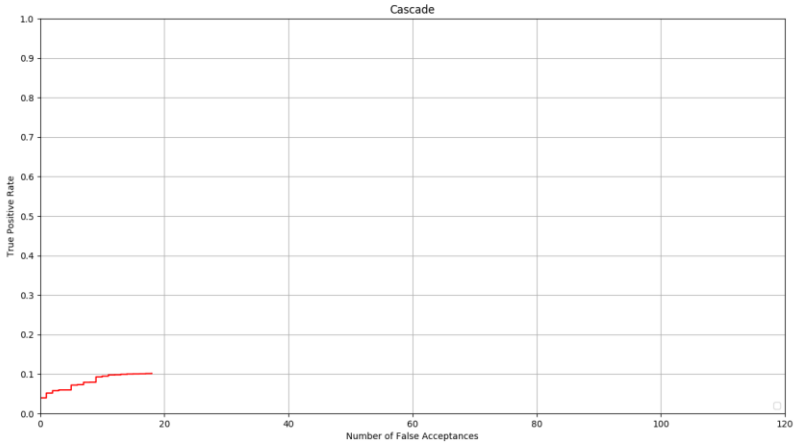

Рисунок 8: Обширный тест для детектора Cascade

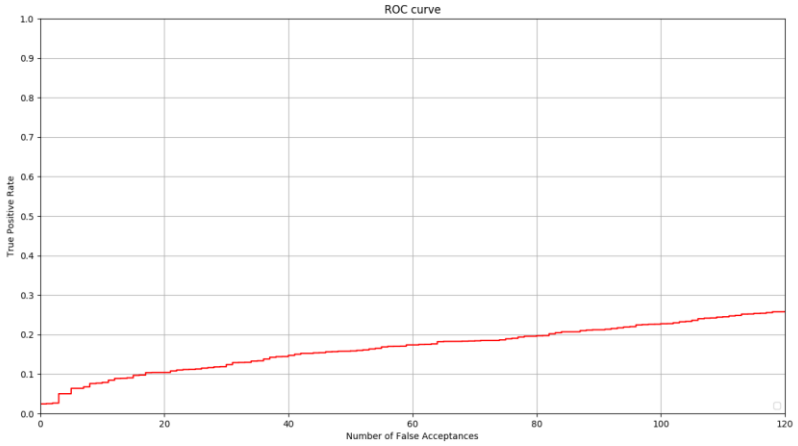

Рисунок 9: Обширный тест для детектора SSD

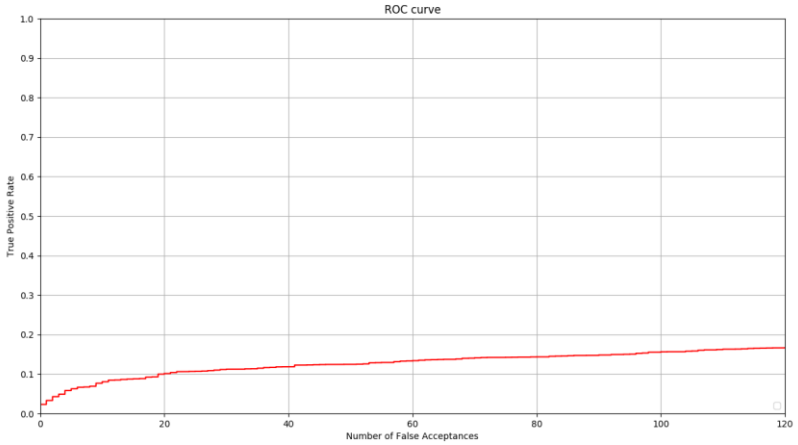

Рисунок 10: Обширный тест для детектора SSD-Retina

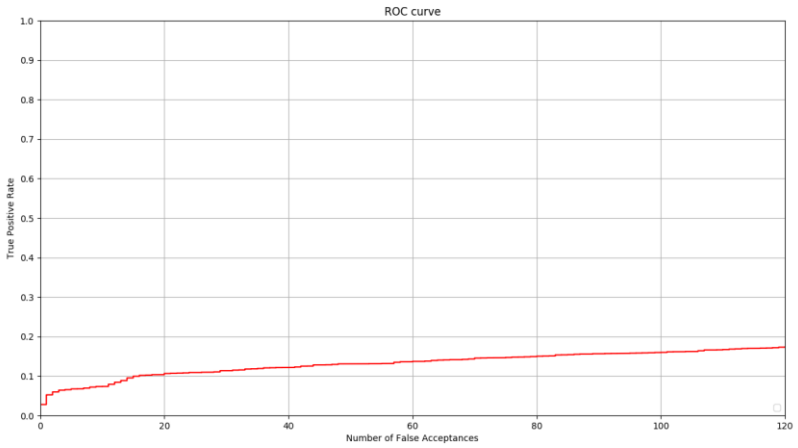

Рисунок 11: Обширный тест для детектора SSD-Retina-quantized

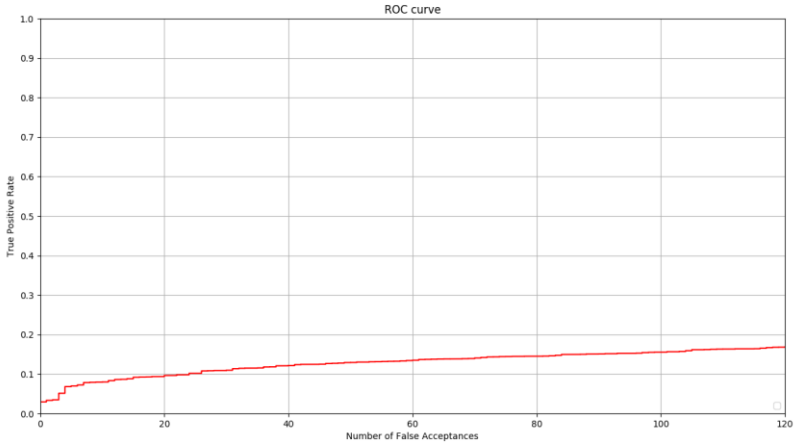

Рисунок 12: Обширный тест для детектора SSD-Retina-quantized-mobile

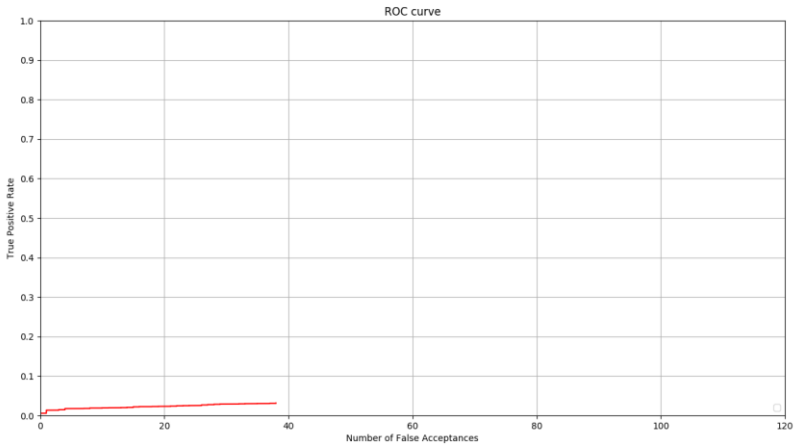

Рисунок 13: Обширный тест для детектора SSD-Retina-ShuffleNet

Точность контрольного теста подвыборки, контролируемого FDDB:

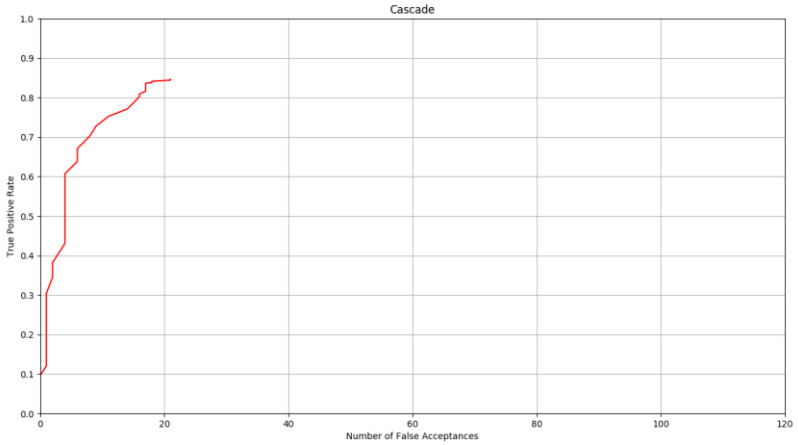

Рисунок 14: Управляемый FDDB дискретный тест для детектора Cascade

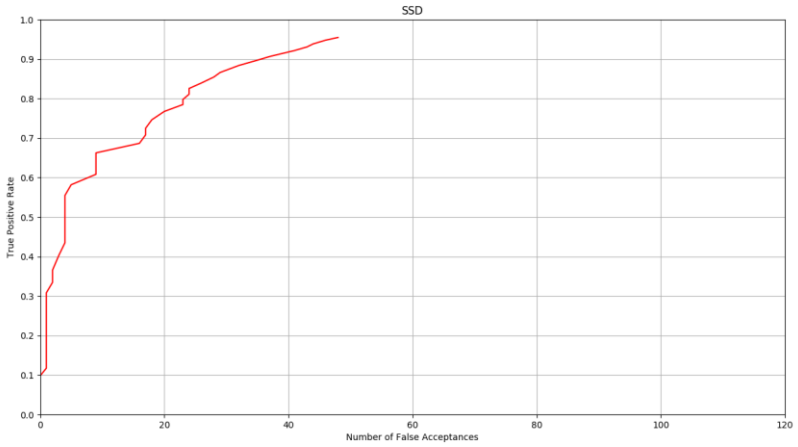

Рисунок 15: Управляемый FDDB дискретный тест для детектора **SSD**

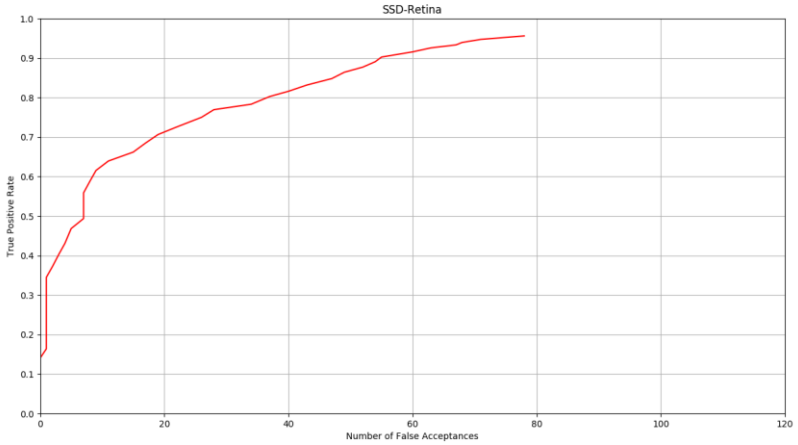

Рисунок 16: Управляемый FDDB дискретный тест для детектора **SSD-Retina**

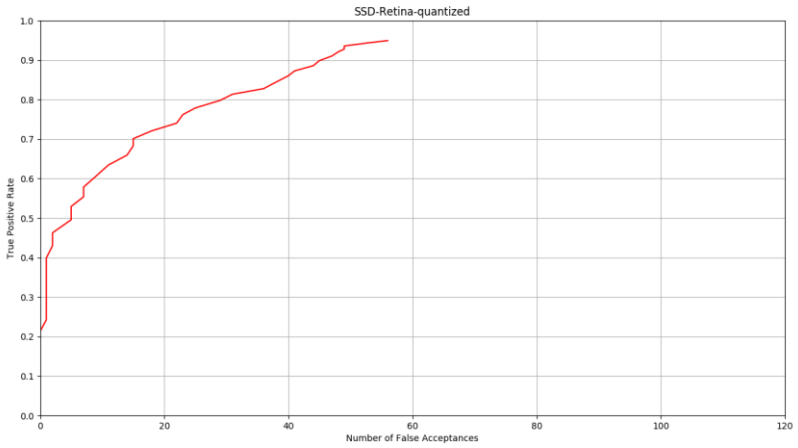

Рисунок 17: Управляемый FDDB дискретный тест для детектора **SSD-Retina-quantized**

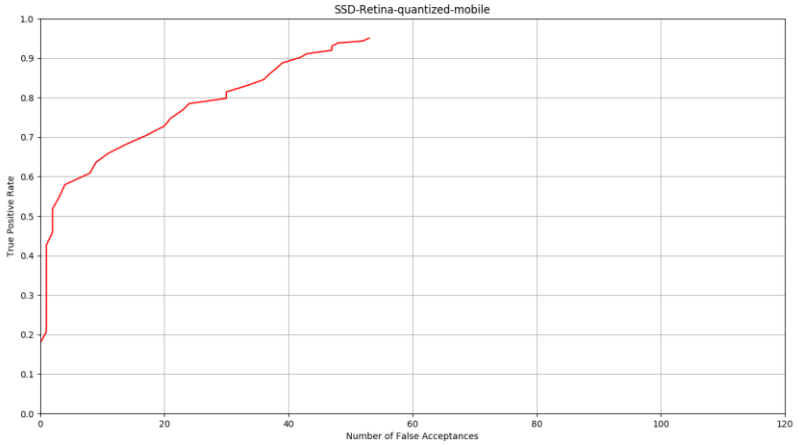

Рисунок 18: Управляемый FDDB дискретный тест для детектора **SSD-Retina-quantized-mobile**

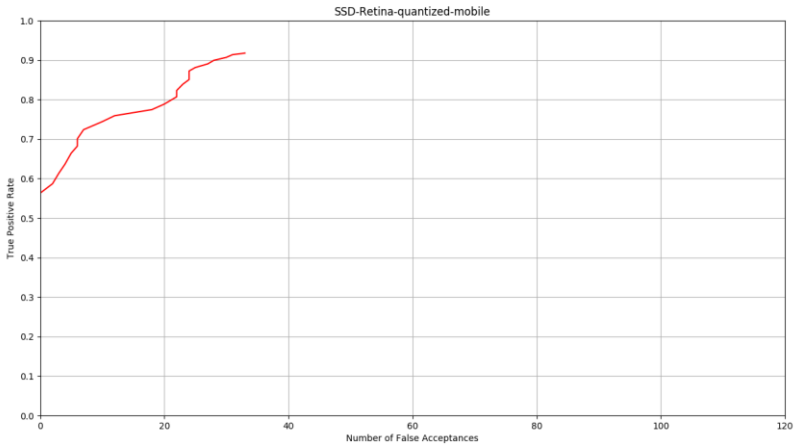

Рисунок 19: Управляемый FDDB дискретный тест для детектора **SSD-Retina-ShuffleNet**

ОТСЛЕЖИВАНИЕ

Краткий обзор

Модуль для отслеживания лиц в видеопотоке в **Face.SDK** представлен классом **FSDK::tracking::MultiTracker.** Он работает совместно с детектором лиц (можно использовать любой детектор лиц, реализованный в **Face.SDK**).

Обратите внимание, что для защиты от неточностей детектора **FSDK::tracking::MultiTracker** добавляет небольшую ошибку в координаты ограничивающих рамок лиц, локализованных детектором. Поэтому результаты локализации детектора на некоторых кадрах могут незначительно отличаться от результатов **FSDK::tracking::MultiTracker**, который использует тот же детектор для локализации лица. По умолчанию эта дополнительная ошибка вычисляется как случайная для каждой ограничивающей лицо рамки, поэтому результаты отслеживания не воспроизводятся с абсолютной точностью из-за этой неопределенности. Чтобы сделать расчет ошибок детерминированным, при создании объекта **FSDK::tracking::MultiTracker** установите для аргумента *is_deterministic* (*isDeterministic* в Java API) значение «true». В этом случае результаты отслеживания будут детерминированными и воспроизводимыми, но менее устойчивыми к ошибкам обнаружения лиц.

Для каждого лица на каждом кадре модуль предоставляет следующее:

- положение (например, ограничивающую рамку);
- уникальный ID;
- оценка уверенности (качество изображения лица).

Набор положений с одинаковым ID называется треком. Обратите внимание, что несколько треков могут соответствовать одному человеку. Окончательная оценка достоверности вычисляется с помощью средства проверки лиц.

Модуль имеет три базовые функции:

- обработка существующих треков;
- удаление устаревших треков;
- создание и коррекция треков (с помощью детектора лиц).

Удаление устаревших треков основано на внутренней уверенности, которая недоступна через интерфейс отслеживания. Некоторые параметры можно использовать для корректировки этого показателя достоверности. В зависимости от приложения детектор может активироваться для каждого кадра или нет. В последнем случае можно выбрать период работы (в кадрах). Обычно это 3-5 кадров.

Алгоритм отслеживания основан на локальных функциях, которые называются двоичными дескрипторами. Эти черты вычисляются по лицевым ориентирам.

Рекомендации по использованию

Ниже приведен минимальный размер лица для отслеживания при различных сценариях:

- Стадионы: кадр 180px/2Mpx
- Пропускные пункты: кадр 300px/2Mpx
- Наблюдение: кадр 120px/2Mpx

Примеры использования

```
FSDK::tracking::MultiTracker tracker(profile, device_idx);                // Запуск
    трекера лиц
int fps = 15;                                                             // Кадров в
    секунду
while (true) {                                                            // до конца
    видео
    FSDK::containers::Image current_frame = get_next_image_from_video(); // получение
        нового кадра
    if (!current_frame.is_initialized()) {
        break;
    }
    FSDK::containers::FVector<FSDK::containers::FaceInfo> current_tracks =
        tracker.process_next_frame(current_frame, static_cast<int>(1000. / fps));
}
```

Список 13: C++: трекер лиц

```
try (MultiTracker tracker = MultiTracker.createMultiTracker(profile)) {
    int fps = 15;
    while (true) { // до конца видео
        try (Image currentFrame = getNextImageFromVideo()) { // получение нового
            изображения из видео
            if (!currentFrame.isInitialized()) {
                break;
            }
            FaceInfo[] currentTracks = tracker.processNextFrame(currentFrame, (int)
                (1000. / fps));
        }
    }
}
```

Список 14: Java: трекер лиц

Производительность

Качество отслеживания во многом зависит от детектора лица. Скорость сильно зависит от параметров и количества отслеживаемых лиц. При указанных выше значениях параметров типичное время обработки составляет 5-7 мс на лицо в кадре (без учета времени работы детектора лиц). Обратите внимание, что это значение почти не зависит от разрешения входного кадра.

ОПРЕДЕЛЕНИЕ КЛЮЧЕВЫХ ТОЧЕК

Краткий обзор

Face.SDK может обнаруживать лицевые ориентиры на изображении. Он использует стандартную схему разметки ориентиров:

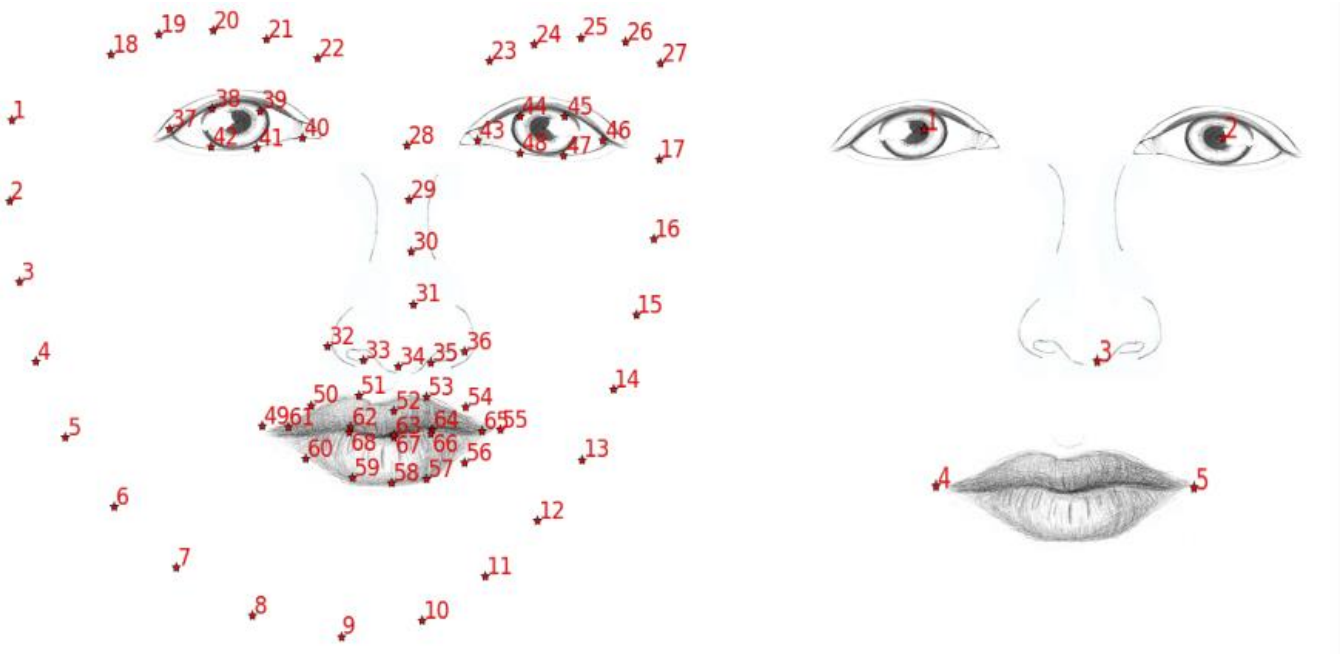

Рисунок 20: 5 и 68-точечная разметка ориентиров

Существует два типа определния ключевых точек. **FSDK::detection::LandmarkDetector** определяет точки на одном изображении, тогда как **FSDK::detection::LandmarkTracker** подходит только для работы с видео. Второй тип использует информацию с предыдущего кадра и отмечает точки на текущем кадре. Если вы не уверены какой тип детектора подходит для ваших задач, используйте **FSDK::detection::LandmarkDetector.** Алгоритм определения ключевых точек основан на подходе с ограниченной локальной моделью. Constrained Local Model (CLM) – это класс методов нахождения наборов точек (ограниченных статистической моделью формы) на целевом изображении. Общий подход включает следующие шаги:

• выборка области изображения вокруг текущей оценки;

• для каждой точки создание «ответного изображения» с указанием стоимости размещения точки в каждом пикселе;

• поиск комбинации точек, которая оптимизирует общую стоимость, манипулируя параметрами модели формы.

Другой тип реализованного здесь алгоритма - это подход на основе ансамбля деревьев регрессии.

Примеры использования

Для **FSDK::detection::LandmarkDetector**:

```
FSDK::detection::LandmarkDetector lm_detector(profile);
bool correct_rotation_det = false;
FSDK::containers::landmarks_list_t landmarks1 =
    lm_detector.detect(img1, faces[0].area, correct_rotation_det); // снимите первое
        лицо с детектора
FSDK::core_types::PointF eye_right = FSDK::detection::eye_location_right(landmarks1);
FSDK::core_types::PointF eye_left  = FSDK::detection::eye_location_left(landmarks1);
FSDK::containers::EyesPosition eyesPosition(eye_right, eye_left);
FSDK::containers::landmarks_list_t landmarks2 = lm_detector.detect(img1, eyesPosition);
    // или используйте координаты глаз
```

Список 15: C++: детектор лицевых точек

```
try (LandmarkDetector lmDetector = LandmarkDetector.createLandmarkDetector(profile)) {
    boolean correctRotation = false;
    PointF[] landmarksByArea =
            lmDetector.detect(image, faces[0].area, correctRotation); // снимите
                первое лицо с детектора
    PointF rightEye = LandmarksFunctions.eyeLocationRight(landmarksByArea);
    PointF leftEye = LandmarksFunctions.eyeLocationLeft(landmarksByArea);
    EyesPosition eyesPosition = new EyesPosition(rightEye, leftEye);
    PointF[] landmarksByEyes = lmDetector.detect(image, eyesPosition); // или
        используйте координаты глаз
}
```

Список 16: Java: детектор лицевых точек

For **FSDK::detection::LandmarkTracker**:

```
FSDK::detection::LandmarkTracker lm_tracker1(profile);
FSDK::detection::LandmarkTracker lm_tracker2(profile);
bool correct_rotation_track = false;
FSDK::containers::landmarks_list_t landmarks =
    lm_tracker1.track(img1, faces[0].area, correct_rotation_track); // снимите первое
        лицо среди остальных лиц
FSDK::core_types::PointF eye_right2 = FSDK::detection::eye_location_right(landmarks);
FSDK::core_types::PointF eye_left2  = FSDK::detection::eye_location_left(landmarks);
FSDK::containers::EyesPosition eyesPosition2(eye_right2, eye_left2);
FSDK::containers::landmarks_list_t landmarks3 = lm_tracker2.track(img1, eyesPosition2);
    // или используйте координаты глаз
```

Список 17: C++: трекер лицевых точек

```
try (LandmarkTracker lmTracker1 = LandmarkTracker.createLandmarkTracker(profile);
     LandmarkTracker lmTracker2 = LandmarkTracker.createLandmarkTracker(profile)) {
    boolean correctRotation = false;
    PointF[] landmarksByArea = lmTracker1.track(image, faces[0].area, correctRotation);
        // снимите первое лицо
    PointF rightEye = LandmarksFunctions.eyeLocationRight(landmarksByArea);
    PointF leftEye = LandmarksFunctions.eyeLocationLeft(landmarksByArea);
    EyesPosition eyesPosition = new EyesPosition(rightEye, leftEye);
    PointF[] landmarksByEyes = lmTracker2.track(image, eyesPosition); // или
        используйте координаты глаз
}
```

Список 18: Java: трекер лицевых точек

Точность

Все расчеты произведены на бэкенде Caffe. Точность на наборах данных Helen и LFPW.

Таблица 3: Точность детектора лицевых точек

Ошибки на наборах данных Helen/LFPW	68_error	51_error	5_error
ERT	5.75/5.76	4.52/4.73	3.87/4.15
CNN	—	—	7.81/8.55

где:

68_error = (среднее отклонение 68 точек от истины, где среднее отклонение нормируется расстоянием между глазами) * 100%;

51_error = (среднее отклонение 51 точек (кроме точек с 1 по 17) от истины, где среднее отклонение отклонение нормируется расстоянием между глазами) * 100%.

5_error = (среднее отклонение 5 точек (глаза, нос и уголки рта) от истины, где среднее отклонение нормируется расстоянием между глазами) * 100%.

РАСПОЗНАВАНИЕ

Краткий обзор

Модуль биометрического распознавания создан для идентификации или верификации лица человека по изображению лица. В Face.SDK распознавание выполняется путем сравнения биометрических характеристик, полученных из изображений. Эти функции называются ИЗЛ (идентификационная запись лица). Вы можете создать ИЗЛ, обновить существующий ИЗЛ, сравнить пару ИЗЛ и идентифицировать ИЗЛ с набором (галереей), содержащим ранее созданные ИЗЛ.

Для процедуры распознавания лиц рекомендуется минимум 150 пикселей на ширину лица. В случае, если условия освещения контролируемы, необходимо соблюдать следующие требования:

- избегать теней в области лица;
- избегайте бликов на очках или на блестящей коже;
- защищать от естественного солнечного света;
- условия освещения должны быть аналогичны условиям, при которых выполняются этапы регистрации и проверки.

Идентификационная запись лица

Face.SDK создает ИЗЛ используя **Face.SDK::identification::FIREngine** – интерфейса для набора реализованных в **Face.SDK** алгоритмов на основе глубокой сверточной нейронной сети для извлечения ИЗЛ из изображений лиц и точек (см. Раздел 10 о ориентирах).

Длина созданного вектора ИЗЛ зависит от алгоритма, используемого для его извлечения. Важно учитывать это при выборе алгоритма, поскольку длина ИЗЛ влияет на производительность распознавания **Face.SDK** и требования к памяти (см. Разделы 14.5 и 14.6).

В таблице 4 представлены оценки размера ИЗЛ для всех используемых в **Face.SDK** алгоритмов извлечения ИЗЛ:

Таблица 4: *Длина и размер ИЗЛ для алгоритмов извлечения ИЗЛ*

Алгоритм	Длина ИЗЛ	Размер ИЗЛ
IR_SE_38_256	256	557
SlenderFAN-D3	512	1061
tpaminet3-ao256a	256	552
FastNet22F	256	555
FastNet22FMA	256	562
vfn_b	256	550
vfn_bm	256	551

где:

- длина ИЗЛ – это количество элементов в векторе ИЗЛ;
- Размер ИЗЛ — это размер сериализованного ИЗЛ в байтах. Он состоит из размера вектора признаков и размера служебной информации. ИЗЛ может также содержать изображения, которые увеличат его размер.

Примеры использования

```
FSDK::identification::FIREngine fir_engine(profile);
FSDK::containers::FVector<FSDK::containers::landmarks_list_t> faces_landmarks;
faces_landmarks.push_back(landmarks);
FSDK::containers::FVector<FSDK::containers::Image> faces_images;
faces_images.push_back(img1);
FSDK::containers::FIROptional p_fir = fir_engine.create_fir(faces_landmarks,
    faces_images);
if (p_fir) {
    const FSDK::containers::FIR& fir = p_fir.get();
    FSDK::containers::ByteArray sfir = fir.serialize();
    std::ofstream fir_stream("название_файла_изл", std::ios::out |
    std::ios::binary); fir_stream.write(sfir.get_data(), sfir.length());
    fir_stream.close();
}
```

Список 19: C++: создание ИЗЛ

```
try (FIREngine firEngine = FIREngine.createFIREngine(profile, −1,
    CaffeNetEngineType.DEFAULT, 0)) {
    PointF[][] facesLandmarks = new PointF[][]{landmarks};
    Image[] facesImages = new Image[]{image};
    FIR fir = firEngine.createFIR(facesLandmarks, facesImages, false);
    if (fir != null) {
        byte[] sfir = fir.serialize();
        Files.write(FileSystems.getDefault().getPath("изл.модель"), sfir);
        fir.close();
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

Список 20: Java: создание ИЗЛ

Обратите внимание, что функция create_fir (создать ИЗЛ) возвращает пустое значение, если извлеченный ИЗЛ не прошел проверку валидатором Идентификационной Записи Лица **Face.SDK**. Валидатор **Face.SDK** ИЗЛ — это внутренний алгоритм для оценки уверенности в том, что ИЗЛ был извлечен из изображения, которое действительно содержит лицо, и из точно локализованных лицевых точек. Чтобы определить, верен ли ИЗЛ, эта оценка достоверности сравнивается с порогом проверки. Чтобы уменьшить количество отклоненных ИЗЛ, вы можете снизить порог проверки или отключить валидатор в конфигурации **Face.SDK**. Графики точности валидации для всех алгоритмов извлечения ИЗЛ **Face.SDK** представлены в п. 11.8.2.

Для оценки сходства ИЗЛ **Face.SDK** использует **FSDK::identity::IdentificationEngine** - интерфейс для методов, основанных на сравнении ИЗЛ (сравнение ИЗЛ и идентификация ИЗЛ).

```
FSDK::identification::IdentificationEngine identification_engine(profile);

FSDK::containers::FIR first_fir("первое_имя_изл");
FSDK::containers::FIR second_fir("второе_имя_изл");

float res = identification_engine.compare(first_fir, second_fir);
```

Список 21: C++: сравнение ИЗЛ

```
try (IdentificationEngine identificationEngine = IdentificationEngine
        .createIdentificationEngine(profile);
     FIR firstFir = FIR.createFIR("первоеИмяФайлаИзл");
     FIR secondFir = FIR.createFIR("второеИмяФайлаИзл ")) {
    float res = identificationEngine.compare(firstFir, secondFir);
}
```

Список 22: Java: сравнение ИЗЛ

Функция **compare** возвращает оценку схожести двух ИЗЛ (значение от 0,0 до 1,0).

Асинхронный интерфейс

Также существует асинхронный интерфейс для создания ИЗЛ.

Он предназначен для использования в многопоточных системах с множеством задач по созданию ИЗЛ. По сравнению с описанным выше **FSDK::identity::FIREngine**, он предоставляет только один метод, поэтому предполагается, что обнаружение/отслеживание и нормализация выполняются вручную (возможно, в другом потоке или даже на другом устройстве).

Эта функциональность обеспечивается классом FSDK::identity::FIRMachine и несколькими вспомогательными классами.

Принимая во внимание способность FIRMachine использовать больше ресурсов для минимизации временного интервала от отправки новой задачи до ее завершения, предполагается, что в текущей работающей системе будет не более одного экземпляра FIRMachine.

Поскольку существует возможность передавать конфигурацию, которая использует произвольное количество движков с произвольными размерами пакетов на графическом процессоре, пользователь несет ответственность за то, чтобы на каждом графическом процессоре было достаточно памяти, чтобы соответствовать всем необходимым движкам, которые ему назначены. Хотя существует механизм, который не позволит вам совершать грубые ошибки, этот механизм не идеален, поэтому проверьте свои конфигурации дважды.

Работа автоконфигурации FIRMachine будет пытаться использовать максимальное количество ресурсов центрального и графического процессоров, поэтому в момент инициализации FIRMachine все ресурсы, необходимые для других алгоритмов или самой системы, должны быть инициализированы.

Автоконфигурация FIRMachine не предназначена для использования в производственных системах, поскольку несмотря на то, что она делает некоторые предположения и вычисления для проверки доступных ресурсов, но эти вычисления могут быть несовершенными. Таким образом, хотя автоконфигурация является простейшим способом опробовать FIRMachine в действии, она не обеспечит вам наилучшую производительность.

Примеры использования

Процесс создания ИЗЛ отличается при использовании FIRMachine и представлен ниже. Процесс сравнения ИЗЛ аналогичен описанному в 11.2.1.

```
class Callback : public FSDK::containers::FIRFutureCallback
{
public:
    Callback(const std::string& dir, const std::string& log_file) :
        save_directory(dir), output(log_file) {
    }
    ~Callback() final = default;

    void ready(const char* label, const FSDK::containers::FIROptional& fir) const final
        {
         const std::string save_path = save_directory + '/' + label + ".fir";
         if (fir) {
             fir.get().save(save_path.c_str());
         }

         std::unique_lock<std::mutex> lock(write_mutex);
         output << "Model created for task with label " << label << " and saved by path
             " << save_path << std::endl;
    }
    void exception(const char* label, const FSDK::exceptions::FException& e) const
        final {
         FIRFutureCallback::exception(label, e);
         std::unique_lock<std::mutex> lock(write_mutex);
         output << "Error while executing task with label " << label << ": " << e.what()
             << std::endl;
    }

private:
    const std::string save_directory;
    mutable std::ofstream output;
    mutable std::mutex write_mutex;
};
```

Список 23: C++: реализация обратного вызова для асинхронного создания ИЗЛ

```
FSDK::containers::FIRMachineCPUConfig cpu_config(4, 2,
    FSDK::core_types::CaffeNetEngineType::OPENCV);
// or FSDK::containers::FIRMachineCPUConfig cpu_config =
    FSDK::containers::FIRMachineConfig::AutoCPUConfig();
// or FSDK::containers::FIRMachineCPUConfig cpu_config =
    FSDK::containers::FIRMachineConfig::NoCPUConfig();
FSDK::containers::FIRMachineGPUConfig gpu_config({{0, 10, 1}, {1, 1, 20}});
// or FSDK::containers::FIRMachineGPUConfig gpu_config =
    FSDK::containers::FIRMachineConfig::AutoGPUConfig();
// or FSDK::containers::FIRMachineGPUConfig gpu_config =
    FSDK::containers::FIRMachineConfig::NoGPUConfig();
FSDK::containers::FIRMachineConfig fir_machine_config(cpu_config, gpu_config);
const Callback callback("/path/to/models/dir", "/path/to/log/file");
FSDK::identification::FIRMachine fir_machine(profile, fir_machine_config, &callback, 5);

const FSDK::containers::Image crop =
    FSDK::detection::normalize_by_landmarks(profile, faces_landmarks[0],
        faces_images[0]);
const FSDK::containers::LabeledImage labeled_image1(crop), labeled_image2(crop),
    labeled_image3(crop);
const FSDK::containers::FVector<FSDK::containers::LabeledImage> tasks =
    {labeled_image1, labeled_image2, labeled_image3};
FSDK::containers::FVector<FSDK::containers::FIRFuture> futures        =
    fir_machine.create_firs(tasks);
for (size_t fidx = 0; fidx < futures.size(); ++fidx) {
    auto& future            = futures[fidx];
    const std::string label = tasks[fidx].label();
    try {
        const FSDK::containers::FIROptional firopt = future.get();
        if (firopt) {
            std::cout << "Got model for task with label " << label << std::endl;
        }
    } catch (const FSDK::exceptions::FException&) {
        std::cerr << "Got exception for task with label " << label << std::endl;
    }
}
```

Список 24: C++: асинхронное создание ИЗЛ

```
static class Callback extends FIRFutureCallback implements AutoCloseable {
    public Callback(File dir, File logFile) throws FileNotFoundException {
        this.saveDirectory = dir;
        this.logFile = logFile;
        this.printWriter = new PrintWriter(new BufferedOutputStream(new
            FileOutputStream(this.logFile)));
    }

    @Override
    public void ready(String label, FIR fir) {
        File savePath = new File(saveDirectory, label + ".fir");
        if (fir != null) {
            try {
                fir.save(savePath.getAbsolutePath());
            } catch (FSDKNativeException e) {
                e.printStackTrace();
            }
        }

        printWriter.print("Model created for task with label" + label + " and saved by
            path " + savePath
                .getAbsolutePath());
    }

    @Override
    public void exception(String label, FSDKNativeException e) {
        super.exception(label, e);
        printWriter.print("Error while executing task with label " + label + ": " +
            e.getMessage());
    }

    @Override
    public void close() throws Exception {
        printWriter.flush();
        printWriter.close();
    }

    private File saveDirectory;
    private File logFile;
    private PrintWriter printWriter;
}
```

Список 25: Java: реализация обратного вызова для асинхронного создания ИЗЛ

```
try (Callback callback = new Callback(new File("/path/to/models/dir"), new
    File("/path/to/log/file"));
     Image crop = LandmarksFunctions.normalizeByLandmarks(profile, landmarks, image)) {
    FIRMachineCPUConfig cpuConfig = new FIRMachineCPUConfig(4, 2,
        CaffeNetEngineType.OPENCV);
    // or FIRMachineCPUConfig cpuConfig = FIRMachineConfig.getAutoCPUConfig();
    // or FIRMachineCPUConfig cpuConfig = FIRMachineConfig.getNoCPUConfig();
    FIRMachineGPUConfig gpuConfig = new FIRMachineGPUConfig(
            new FIRMachineGPUSingleConfig[]{new FIRMachineGPUSingleConfig(0, 10, 1),
                    new FIRMachineGPUSingleConfig(1, 1, 20)});
    // or FIRMachineGPUConfig gpuConfig = FIRMachineConfig.getAutoGPUConfig();
    // or FIRMachineGPUConfig gpuConfig = FIRMachineConfig.getNoGPUConfig();
    FIRMachineConfig firMachineConfig = new FIRMachineConfig(cpuConfig, gpuConfig);
    FIRMachine firMachine = FIRMachine.createFIRMachine(profile, firMachineConfig,
        callback, 5);
    LabeledImage labeledImage1 = LabeledImage.createLabeledImage(crop);
    LabeledImage labeledImage2 = LabeledImage.createLabeledImage(crop);
    LabeledImage labeledImage3 = LabeledImage.createLabeledImage(crop);
    LabeledImage[] tasks = new LabeledImage[]{labeledImage1, labeledImage2,
        labeledImage3};
    FIRFuture[] futures = firMachine.createFIRs(tasks);
    for (int fidx = 0; fidx < futures.length; ++fidx) {
        FIRFuture future = futures[fidx];
        String label = tasks[fidx].getLabel();
        try {
            FIR firopt = future.get();
            if (firopt != null) {
                System.out.println("Got model for task with label " + label);
            }
        } catch (FSDKNativeException e) {
            System.out.println("Got exception for task with label " + label);
        }
    }
    firMachine.close();
    for (LabeledImage li : tasks) {
        li.close();
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

Список 26: Java: асинхронное создание ИЗЛ

Идентификация

Для идентификации лица по эталонному набору лиц **Face.SDK** содержит функцию сравнения ИЗЛ со всеми эталонными ИЗЛ и получения идентификаторов лиц из ближайших ИЗЛ.

Чтобы идентифицировать человека по FIR, вы можете использовать контейнеры **FSDK::identification::Gallery** или **FSDK::identification::IndexGallery** - для хранения списка ИЗЛ для отсылок и быстрого поиска ближайших соседей по сохраненным элементам.

Описание **FSDK::identification::Gallery** и **FSDK::identification::IndexGallery** и рекомендации по выбору между этими модулями приведены ниже.

Идентификация при помощи **FSDK::identification::Gallery**:

```
FSDK::identification::IdentificationEngine identification_engine(profile);

int gpu_device_id = -1;
FSDK::identification::Gallery gallery("path_to_galery_file", gpu_device_id);

FSDK::containers::FIR fir_for_identification("fir_filename");

const size_t max_count_of_closest_candidates = 1;
const float comparison_score_threshold       = 0.5f;

FSDK::containers::FVector<FSDK::containers::IdentificationResult> closest_ids_list =
    identification_engine.identify(gallery,
                                   fir_for_identification,
                                   max_count_of_closest_candidates,
                                   comparison_score_threshold);
```

Список 27: C++: идентификация ИЗЛ с помощью Gallery

```
int gpuDeviceId = -1;
try (IdentificationEngine identificationEngine = IdentificationEngine.
            createIdentificationEngine(profile);
     Gallery gallery = Gallery.createGallery("path_to_galery_file", gpuDeviceId);
     FIR firForIdentification = FIR.createFIR("firFilename")) {
    int macCountOfClothestCandidates = 1;
    float comparisonScoreThreshold = 0.5f;

    IdentificationResult[] clothestIdsList =
            identificationEngine.identify(gallery,
                                          firForIdentification,
                                          macCountOfClothestCandidates,
                                          comparisonScoreThreshold);
}
```

Список 28: Java: идентификация ИЗЛ с помощью Gallery

Identification with **FSDK::identification::IndexGallery**:

```
FSDK::identification::IdentificationEngine identification_engine(profile);

FSDK::identification::IndexGallery index_gallery("path_to_index_galery_file");

FSDK::containers::FIR fir_for_identification("fir_filename");

const size_t max_count_of_closest_candidates = 1;
const float comparison_score_threshold       = 0.5f;

FSDK::containers::FVector<FSDK::containers::IdentificationResult> closest_ids_list =
    identification_engine.identify(index_gallery,
                                   fir_for_identification,
                                   max_count_of_closest_candidates,
                                   comparison_score_threshold);
```

Список 29: C++: идентификация ИЗЛ с помощью IndexGallery

```
try (IdentificationEngine identificationEngine = IdentificationEngine.
            createIdentificationEngine(profile);
     IndexGallery indexGallery =
         IndexGallery.createIndexGallery("path_to_index_galery_file");
     FIR firForIdentification = FIR.createFIR("firFilename")) {
    int macCountOfClothestCandidates = 1;
    float comparisonScoreThreshold = 0.5f;

    IdentificationResult[] clothestIdsList =
            identificationEngine.identify(indexGallery,
                                          firForIdentification,
                                          macCountOfClothestCandidates,
                                          comparisonScoreThreshold);
}
```

Список 30: Java: идентификация ИЗЛ с помощью IndexGallery

Gallery

FSDK::identification::Gallery – это простой, основанный на векторах контейнер для ИЗЛ. Задача поиска ближайшего соседа выполняется при помощи сравнения введеного ИЗЛ со всеми элементами в контейнере и возвращения элементов-соседей.

FSDK::identification::Gallery может обрабатываться на центральном и графическом процессорах.

```
// создать Gallery
int gpu_device_id = -1;
FSDK::identification::Gallery gallery("path_to_galery_file", gpu_device_id);

// добавить ИЗЛ в Gallery
FSDK::containers::FIR fir_for_gallery("reference_fir_filename");
char fir_for_gallery_id[] = "reference fir id";
gallery.append(fir for gallery id, fir for gallery);

// удалить ИЗЛ из Галереи
char                                = "reference fir id";
bool fir_was_successfully_removed = gallery.remove(fir_to_remove_id);

// исправить Gallery
bool shrink_to_fit = false;
if (gallery.is_rectify_needed()) {
    gallery.rectify(shrink_to_fit);
}
```

Список 31: C++: примеры использования Gallery

```
// создать Gallery
int gpuDeviceId = -1;
try (Gallery gallery = Gallery.createGallery("path_to_galery_file", gpuDeviceId);
     FIR firForGallery = FIR.createFIR("referenceFirFilename")) {
    // добавить ИЗЛ в Gallery
    String firForGalleryId = "reference fir id";
    gallery.append(firForGalleryId, firForGallery);

    // удалить ИЗЛ из Gallery
    String firToRemoveId = "reference fir id";
    boolean firWasSuccessfullyRemoved = gallery.remove(firToRemoveId);

    // исправить Gallery
    boolean shrinkToFit = false;
    if (gallery.isRectifyNeeded()) {
        gallery.rectify(shrinkToFit);
    }
}
```

Список 32: Java: примеры использования Gallery

Метод исправления перераспределяет внутреннее хранилище **FSDK::identification::Gallery** для оптимизации дальнейших операций. Мы рекомендуем вызывать его, когда метод is_rectify_needed (isRectifyNeeded) вернул true.

IndexGallery

FSDK::identification::IndexGallery основана на иерархической структуре графа поиска и ускоренно выполняет задачу поиска ближайшего соседа. **FSDK::identification::IndexGallery** реализован только на центральном процессоре.

```
// создать IndexGallery
FSDK::identification::IndexGallery index_gallery("path_to_index_galery_file");

// добавить ИЗЛ в IndexGallery
FSDK::containers::FIR fir_for_gallery("reference_fir_filename");
char fir_for_gallery_id[] = "reference fir id";
index_gallery.append(fir_for_gallery_id, fir_for_gallery);

// удалить ИЗЛ из IndexGallery
char fir_to_remove_id[]           = "reference fir id";
bool fir_was_successfully_removed = index_gallery.remove(fir_to_remove_id);

// переиндексировать IndexGallery
if (index_gallery.is_reindex_needed()) {
    index_gallery.reindex();
}
```

Listing 33: C++: IndexGallery Usage Examples

```
// создать IndexGallery
try (IndexGallery indexGallery =
    IndexGallery.createIndexGallery("path_to_index_galery_file");
     FIR firForGallery = FIR.createFIR("referenceFirFilename")) {
    // добавить ИЗЛ в IndexGallery
    String firForGalleryId = "reference fir id";
    indexGallery.append(firForGalleryId, firForGallery);

    // удалить ИЗЛ из IndexGallery
    String firToRemoveId = "reference fir id";
    boolean firWasSuccessfullyRemoved = indexGallery.remove(firToRemoveId);

    // переиндексировать IndexGallery
    if (indexGallery.isReindexNeeded()) {
        indexGallery.reindex();
    }
}
```

Список 34: Java: примеры использования IndexGallery

Обратите внимание, что команда remove в действительности не удаляет запись из графа поиска **FSDK::identification::IndexGallery,** а только помечает ее на удаление, чтобы в дальнейшем исключить из результатов идентификации. Это приводит к быстрой процедуре удаления, но ухудшает производительность идентификации, если количество «удаленных» записей велико. В таком случае мы

рекомендуем переиндексацию, которая перестраивает граф поиска, действительно удаляя все помеченные на удаление записи.

Рекомендуем использовать команду **reindex** если метод is_reindex_needed (isReindexNeeded) вернул true.

Выбор между Gallery и IndexGallery

При выборе между **FSDK::identification::Gallery** и **FSDK::identification::IndexGallery** учитывайте следующие условия:

• В отличие от **FSDK::identification::Gallery**, который выполняет более точный поиск ближайшего соседа методом перебора, **FSDK::identification::IndexGallery** обеспечивает быстрый приблизительный поиск, поэтому результаты могут быть неточными.

• **FSDK::identification::IndexGallery** обеспечивает более быстрый поиск, но более медленное создание и модификацию, чем **FSDK::identification::Gallery**, поэтому мы рекомендуем выбирать его, если вы планируете галерею большого размера, частый поиск по галерее и редкую модификацию галереи. Сравнение производительности идентификации **FSDK::identification::Gallery** и **FSDK::identification::IndexGallery** на центральном и графическом процессорах представлено в 14.6.

• **FSDK::identification::Gallery** может обрабатываться на CPU и GPU, **FSDK::identification::IndexGallery** реализован только на CPU.

• **FSDK::identification::IndexGallery** требует примерно в два раза больше оперативной памяти, чем **FSDK::identification::Gallery** на центральном процессоре для хранения того же количества ИЗЛ.

Тест-кейсы

Чтобы показать производительность в различных тест-кейсах, мы собрали наборы тестов для каждого из них. Список кейсов с кратким описанием:

• Сценарий шестнадцатеричной системы - картинки из социальных сетей;

• набор данных DFW - набор данных из конкурса Disguised Faces in the Wild;

• IJB-C - набор тестов NIST IJB-C 1-1;

• ID азиатов крупным планом - азиатские лица в сценарии селфи-ID, селфи близко к лицу;

• ID азитов снятое с вытянутой руки - азиатские лица в сценарии селфи-ID.

Описание шестнадцатеричной системы

Изображения снимаются камерой выше и ниже уровня глаз, одной или двумя руками. Углы тангажа различаются больше, чем углы рыскания, которые являются фронтальными. Видно некоторое искажение перспективы. Все изображения сделаны в реальном времени. Все изображения принадлежат взрослым людям. Изображения представлены в оттенках серого.

Пример изображений:

Список 21: набор тестов шестнадцатеричной системы

Описание DFW

Набор данных DFW содержит 11157 изображений лиц 1000 личностей, в основном индийского или европейского происхождения. На каждом объекте есть одно обычное изображение лица, которое соответствует не замаскированному фронтальному изображению лица. 903 субъекта имеют одно проверочное изображение лица, которое соответствует не замаскированному фронтальному изображению лица. Это изображение можно использовать для создания незамаскированной пары внутри объекта. У всей 1000 субъектов есть изображения их замаскированных лиц в наборе данных DFW. Для данного субъекта замаскированные лица соответствуют изображениям лиц того же субъекта, замаскированных намеренно или непреднамеренно. У каждого субъекта есть от 1 до 12 замаскированных изображений лиц. У 874 испытуемых есть изображения, принадлежащие их имитаторам. Под имитатором субъекта подразумевается изображение любого другого человека намеренно или непреднамеренно выдающего себя за личность субъекта. Из 874 субъектов каждый имеет от 1 до 21 изображений их имитатора.

Пример изображений:

Рисунок 22: примеры набора тестов DFW

Описание IJB-C

Набор данных IARPA Janus Benchmark–C (IJB-C) содержит лицензированные Creative Commons изображения и видео с лицами 3531 субъектов, дополнительно к 1661 субъекта к IJB-B. Все субъекты в наборе данных обязательно появляются как минимум в двух изображениях и одном видео. Субъекты изображены в различных позах. Субъекты - представители различных профессий, что позволяет избежать ловушки, связанной с использованием СМИ «только для знаменитостей», поскольку люди с профессиями, сильно связанными с внешностью, например актеры и исполнители, могут быть менее репрезентативными для мирового населения.

Пример изображений:

Рисунок 23: примеры набора тестов IJB-C

Фото ID азиатов крупным планом

Селфи лиц азиатов в сравнении с фото с фото из их документов, удостоверяющих личность (id). Селфи сняты крупным планом.

Пример изображений:

Рисунок 24: пример набора тестов ID азиатов крпным планом

Фото ID азиатов с расстояния вытянутой руки

Селфи лиц азиатов в сравнении с фото с фото из их документов, удостоверяющих личность. Селфи сняты с расстояния вытянутой руки.

Пример изображений:

Рисунок 25: пример набора тестов ID азиатов снятое с расстояния вытянутой руки

Метрики

Описание метрик для верификации и идентификации. Основные метрики - FRR, FAR для проверки и FPIR, FNIR для идентификации.

Верификация

Учитывая вектор из N подлинных оценок, u, коэффициент ложного отказа (FRR) вычисляется как доля ниже некоторого порогового значения, T:

$$\mathrm{FRR}(T) = 1 - \frac{1}{N}\sum_{i=1}^{N} H(u_i - T),$$

где $H(x)$ – этофункция единичного шага, а H(0) принимается за равную 1.

Точно так же, учитывая вектор из N оценок самозванца, v, коэффициент ложного принятия (FAR) вычисляется как доля выше T:

$$\mathrm{FAR}(T) = \frac{1}{N}\sum_{i=1}^{N} H(v_i - T).$$

Порог T может принимать любое значение из интервала [0, 1]. Компромиссные характеристики ошибок - это графики зависимости FRR(T) от FAR(T), которые называются графиком DET (обнаружение компромисса с ошибками).

Идентификация

Обычно поиск проводится в зарегистрированной совокупности из N идентификаторов, а алгоритм конфигурируется так, чтобы возвращать L ближайших кандидатов-идентификаторов. Эти кандидаты ранжируются по количеству очков в порядке убывания. Человек-аналитик может исследовать либо всех L кандидатов, либо только лучшие R ≤ L идентичности, либо только тех, у которых оценка превышает пороговое значение T.

Коэффициент ложноположительной идентификации — это доля поисков, которые приводят к неблагоприятному результату:

$$\mathrm{FPIR}(N, T) = \frac{\text{Num. поиски без совпадения, когда один или несколько зачисленных кандидатов возвращаются на уровне или выше порогового значения T}}{\text{Num. предпринята попытка обыска без партнера.}}$$

В соответствии с этим определением FPIR может быть вычислен из лучшего кандидата без совпадений, полученного в результате поиска - нет необходимости рассматривать кандидатов с рангом 2 и выше.

Если в ходе поиска возвращено L кандидатов, можно подготовить более короткий список кандидатов, взяв R ≤ L лучших кандидатов, для которых оценка превышает некоторый порог, T. Сокращение списка кандидатов происходит потому, что могут применяться пороговые значения и рассматриваются только короткие списки (например, в соответствии с правилами или наличием рабочей силы). В таком случае полезно указывать точность в терминах R и T, поэтому мы определяем «коэффициент промахов» с общим названием «частота ложных отрицательных идентификаций» (FNIR) следующим образом:

$$\mathrm{FNIR}(N, R, T) = \frac{\text{Num. поиск совпадения с зарегистрированным кандидатом, который не имеет верхних рангов R или имеет оценку ниже порогового значения, T}}{\text{Num. попытка поиска совпадений.}}$$

Эта формулировка проста для оценки, поскольку не различает причины промахов. Таким образом, совпадение, не указанное в списке кандидатов, рассматривается так же, как промах, возникший из-за сбоя в обнаружении лица, алгоритмов низкого качества или сбоев программного обеспечения. Таким образом, если алгоритм не может создать список кандидатов либо из-за неудачного поиска, либо из-за того, что поисковый шаблон не был создан, результат считается неудачным и добавляется к FNIR.

В то время как FNIR указывает «коэффициент неудач» как то, насколько часто правильный кандидат оказывается не выше порогового значения или не имеет хорошего ранга, многие предпочитают

говорить о «коэффициенте попадания». Это коэффициент положительной идентификации (TPIR), который является дополнением к FNIR и дает представление о том, как часто поиски бывают успешными:

$$\mathrm{TPIR}(N, R, T) = 1 - \mathrm{FNIR}(\mathrm{N,R,T})$$

Важным частным случаем является совокупная характеристика соответствия (CMC), которая суммирует точность только поисков с совпадениями. Он игнорирует оценки сходства, ослабляя пороговое требование, и просто сообщает о доле совмещенных поисков, вернувших совпадение с рангом R или выше.

$$\mathrm{CMC}(N, R) = 1 - \mathrm{FNIR}(N, R, 0).$$

Выбор рабочей точки

Ложные срабатывания: ошибки типа I возникают, когда данные поиска по человека, которого никогда раньше не видели, неверно связаны с данными одного или нескольких зарегистрированных лиц.

Пропуски: ошибки типа II возникают, когда поиск биометрических данных зарегистрированного лица не дает правильных данных.

Выбор рабочей точки (или порога) позволяет балансировать между ошибками I и II типа. В нашем случае порог может варьироваться от 0% до 100%.

Для правильного выбора порога выполните следующие шаги:

• определите условия труда;

• найдите наиболее подходящий набор тестов для ваших условий;

• выберите сценарий работы (идентификация или верификация, размер галереи для верификации);

• определите приемлемый уровень FAR (верификация) или FPIR (идентификация). Например: для схемы идентификации с 1000 лицами в галерее максимально допустимая ложная тревога - одна из ста. Это означает, что мы можем выбрать FPIR равным 1%.

• найти в папке SDK html-график FRFA с результатами тестовых данных, наиболее подходящий к выбранным условиям и сценарию работы;

• найти выбранный FAR или FPIR и определить соответствующий порог;

• определить FRR или FNIR.

Калибровка рабочей точки

Начиная с версии 4.5.0 **Face.SDK** имеет возможность настраивать рабочую точку путем преобразования значения старой рабочей точки в новое значение.

При калибровки рабочей точки качество вашей системы в новой рабочей точке будет таким же, как и качество в старой рабочей точке. В настоящее время оба порога могут принимать значения от 0,001 до 0,999 с точностью 0,001.

Для использования этой функции необходимо отредактировать раздел профиля параметров identify.score_calibration:

• установить параметр threshold_before для определения старого значения рабочей точки,

• установить параметр threshold_after для определения нового значения для рабочей точки,

• установить включенный параметр, чтобы разрешить калибровку рабочей точки.

Точность

Using of methods

- **FSDK::identification::FIREngine::create_fir_by_normalized,**
- **FSDK::identification::FIREngine::create_multiple_firs_by_normalized** or
- **FSDK::identification::FIREngine::extend_fir_by_normalized**

with network IR_SE_38_256 may lead to a slight verification/identification quality degradation. To prevent quality loss use

- **FSDK::identification::FIREngine::create_fir,**
- **FSDK::identification::FIREngine::create_multiple_firs** or
- **FSDK::identification::FIREngine::extend_fir**

methods.

Точность верификации

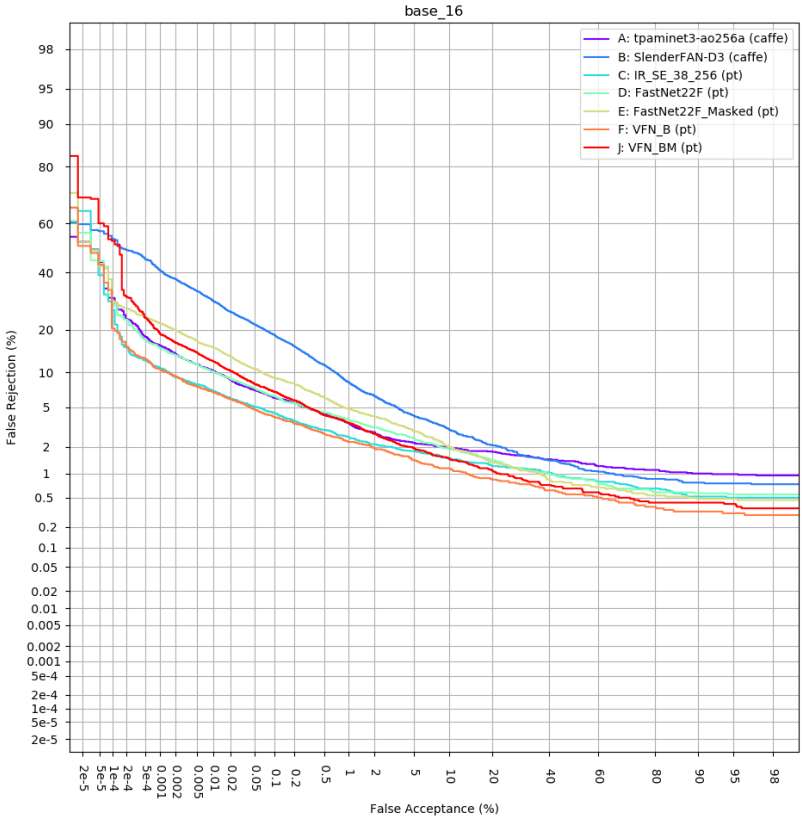

Рисунок 26: шестнадцатеричная система DET

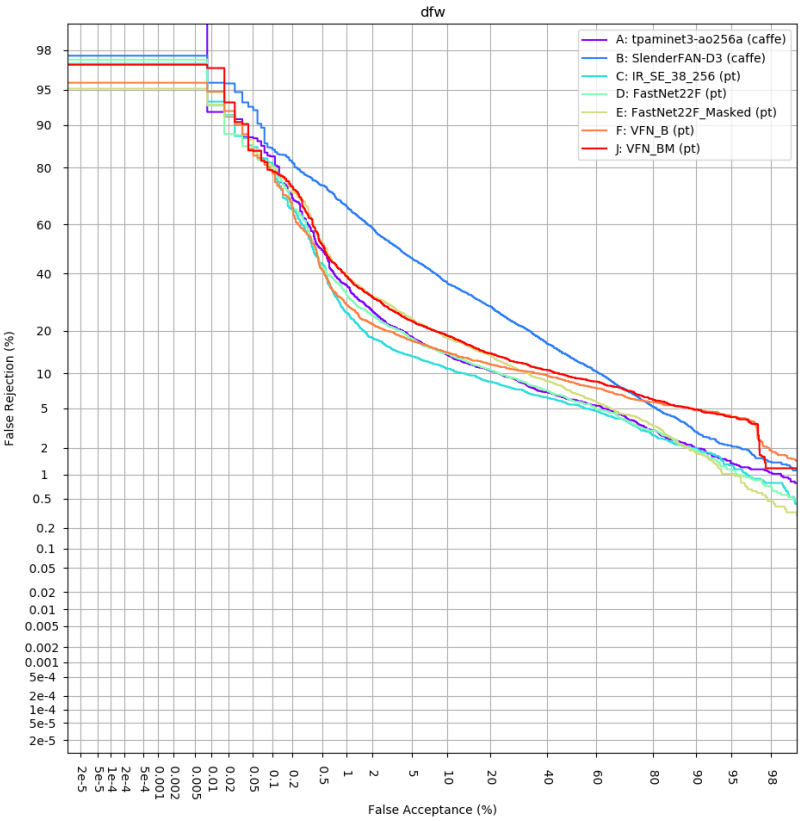

Рисунок 27: DFW DET

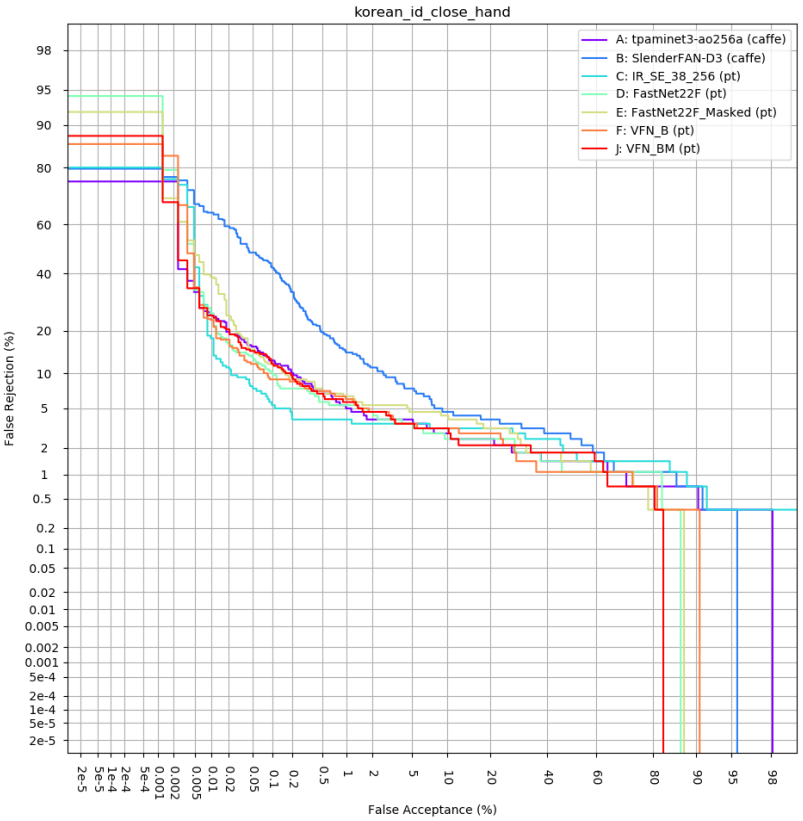

Рисунок 28: DET ID азиатских лиц

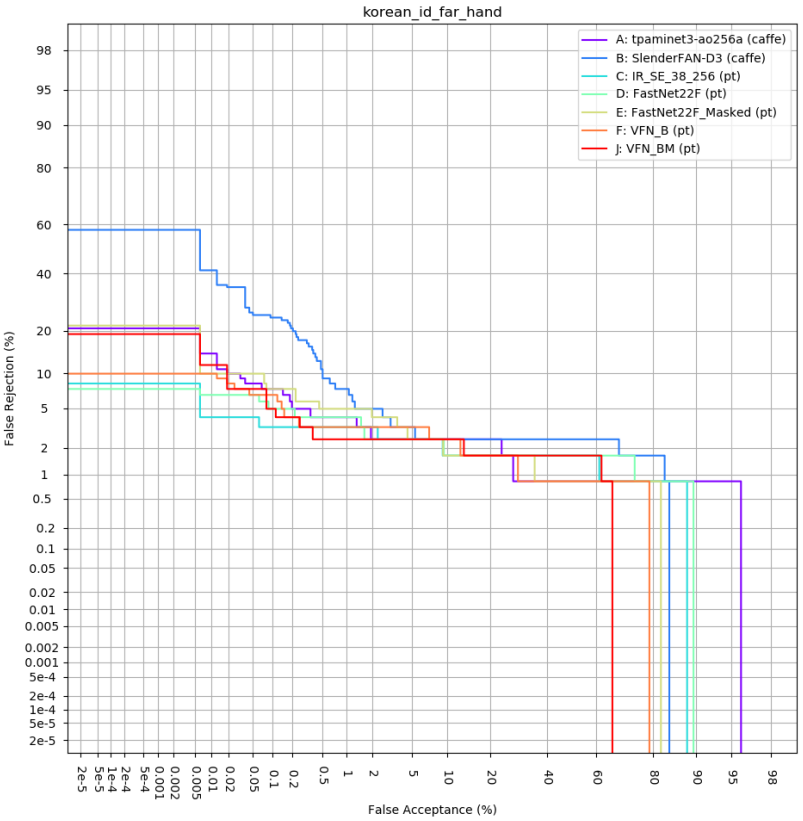

Рисунок 29: DET ID азиатских лиц с расстояния вытянутой руки

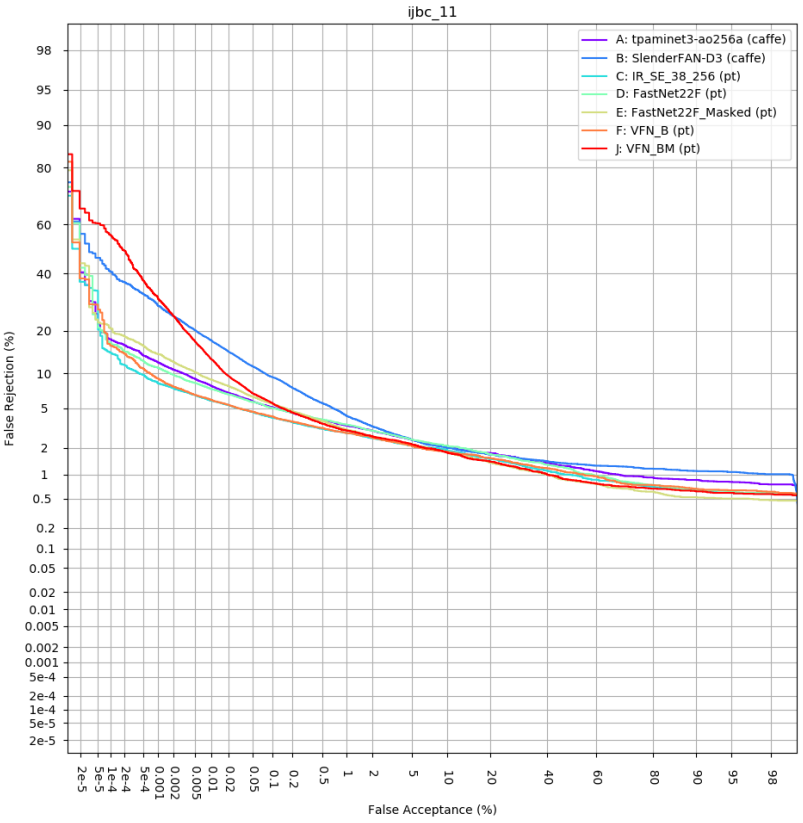

Рисунок 30: IJB-C 1-1 DET

Точность валидации

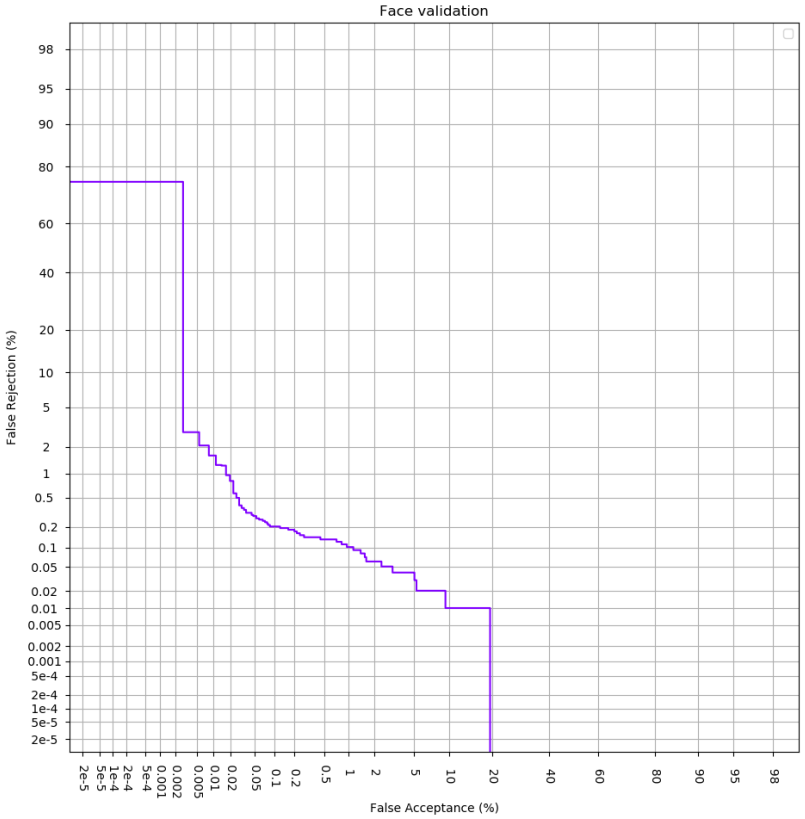

Рисунок 31: SlenderFAN-D3 DET-валидатор

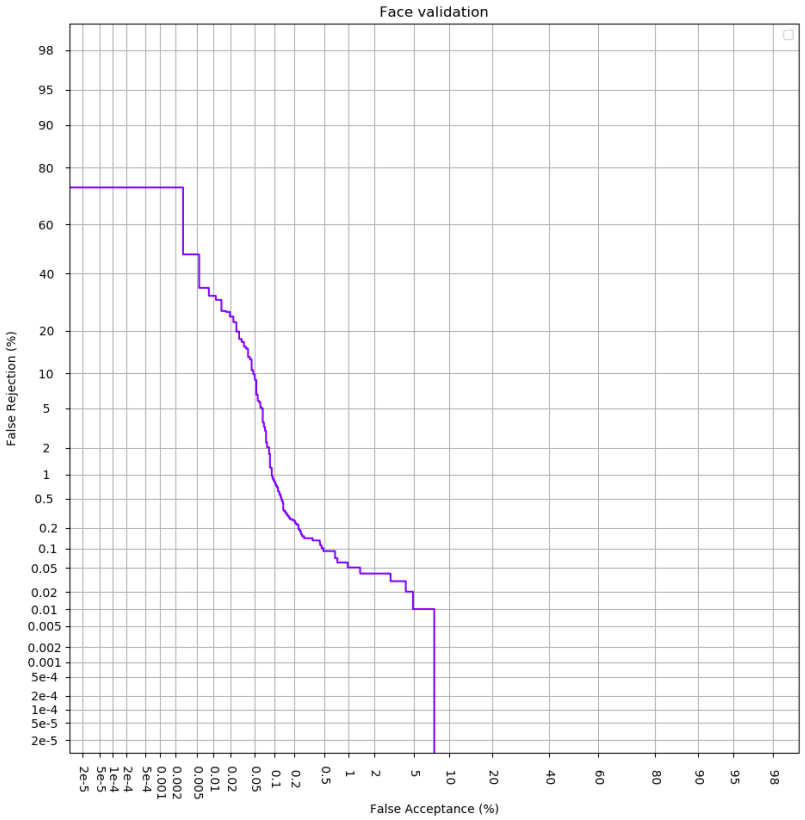

Рисунок 32: tpaminet3-ao256a DET-валидатор

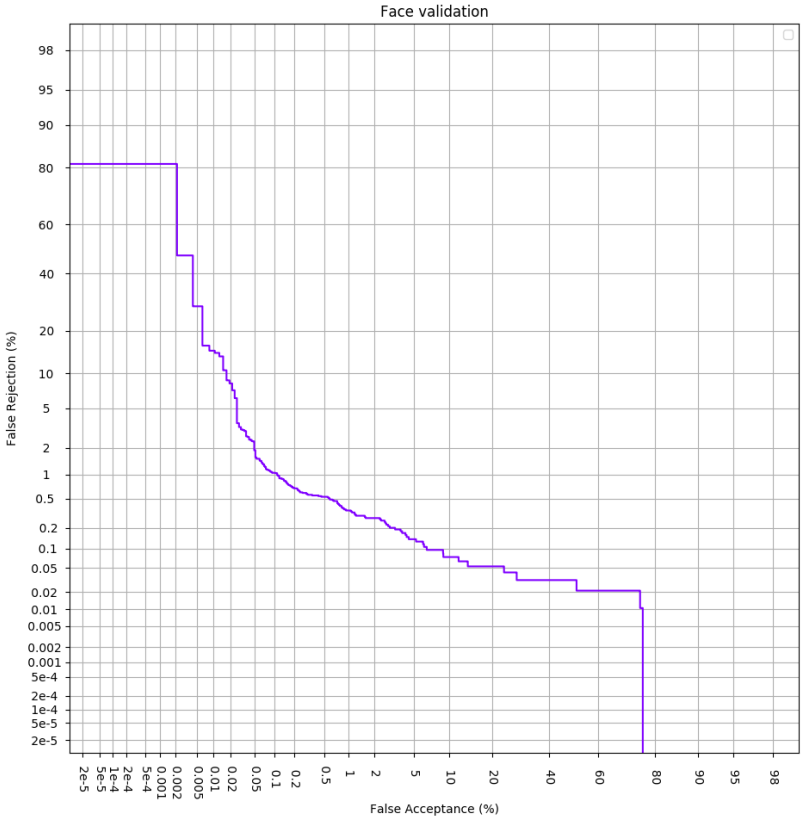

Рисунок 33: IR_SE_38_256 DET-валидатор

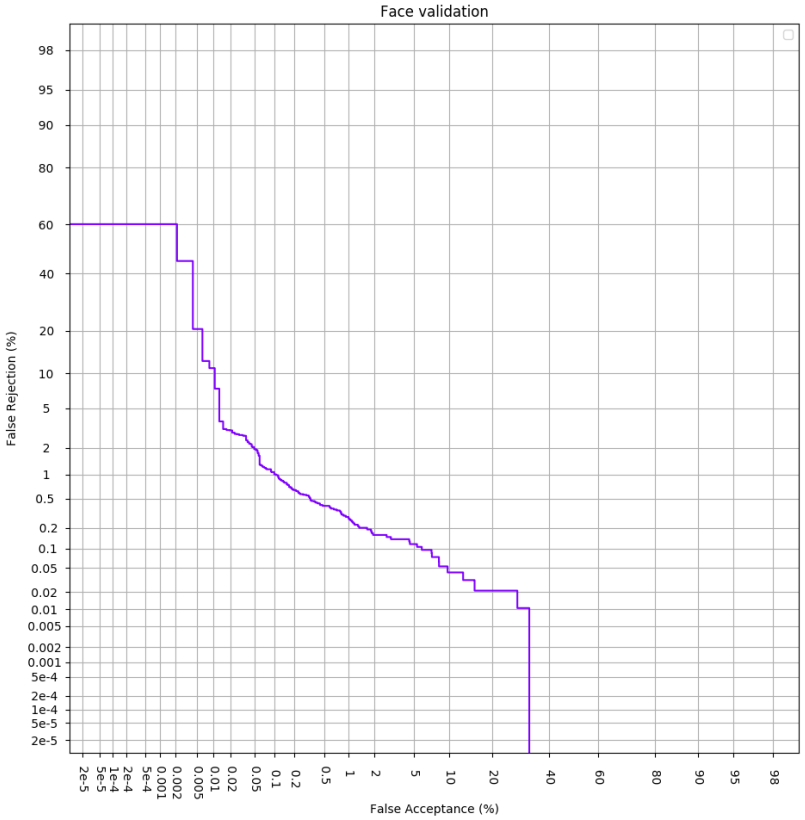

Рисунок 34: FastNet22F DET-валидатор

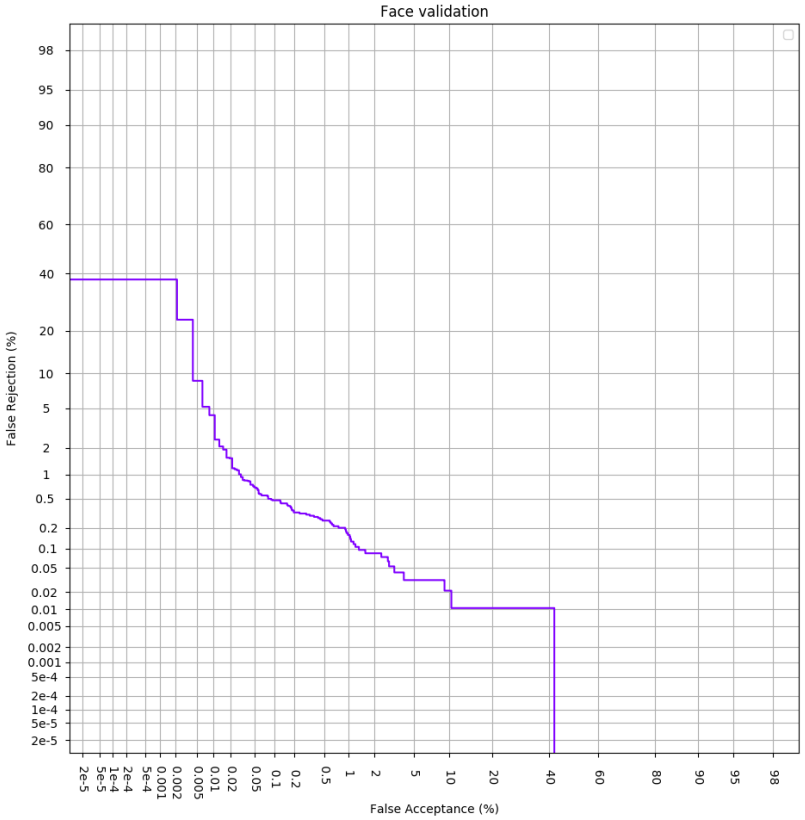

Рисунок 35: FastNet22FMA DET-валидатор

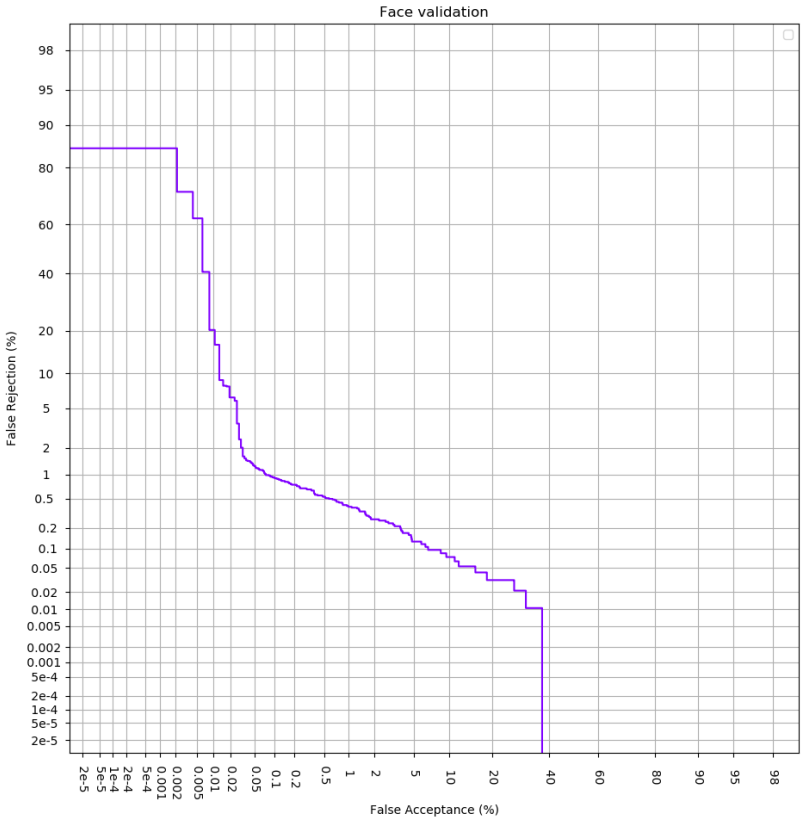

Рисунок 36: vfn_b DET-валидатор

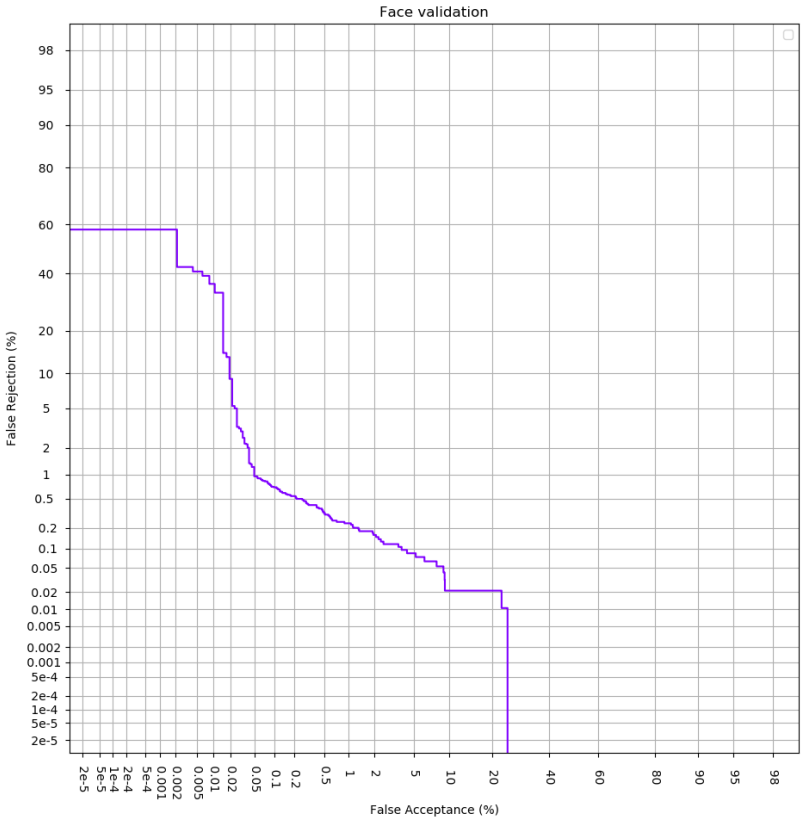

Рисунок 37: vfn_bm DET-валидатор

ПОРТРЕТНЫЕ ХАРАКТЕРИСТИКИ

Данные изображения лица, или сокращенно ISO/IEC 19794-5, являются пятой из 8 частей стандарта ISO/IEC 19794, опубликованного в 2005 году, в котором описаны форматы обмена для нескольких типов биометрических данных. ISO/IEC 19794-5 определяет, в частности, стандартную схему кодификации данных, описывающих человеческие лица, в структуре данных, совместимой с CBEFF, для использования в системах распознавания лиц. Современные фотографии биометрических паспортов должны соответствовать данному стандарту. Многие организации уже начали обеспечивать соблюдение его директив, и было создано несколько программных приложений для автоматической проверки соответствия спецификациям. Стандарт предназначен для обеспечения компьютерного анализа изображений лиц во время автоматической идентификации (поиск по принципу «один-ко-многим») и аутентификации (сравнение один-к-одному), а также для идентификации человека по его отличительным чертам, таким как родинки и шрамы, которые могут быть использованы для проверки личности и проверки результатов компьютерной идентификации человека. Для того чтобы приложения могли работать на различных устройствах, включая устройства с ограниченными ресурсами (например, встроенные системы), и повысить точность распознавания лиц, в спецификации описывается не только формат данных, но и дополнительные требования, а именно: ограничения сцены (освещение, поза, выражение и т. д.); фотографические свойства (позиционирование, фокус камеры и т. д.); и атрибуты цифрового изображения (разрешение изображения, размер изображения и т. д.)

Изображение лица должно соответствовать следующим требованиям:

фронтальная поза анфас. Вращение головы должно быть не более +/- 10 градусов спереди во всех направлениях: крен, тангаж и рыскание;

фон должен быть однотонным или однотонным с постепенным изменением яркости от светлого к темному в одном направлении;

на изображении должно быть только одно лицо;

разрешение должно быть больше 60 пикселей между глазами.

Портретные характеристики

Краткий обзор

Мы реализуем несколько функций для оценки автоматических атрибутов лица. **Face.SDK** может оценить следующие параметры

(см. FSDK::portrait::PortraitCharacteristics **и** FSDK::portrait::PortraitEngine**):**

- deviation_from_uniform_lighting — Равномерность освещения лица в диапазоне от 0 до 1;
- exposure — Среднее значение серого в области лица в диапазоне от 0 до 255;
- grayscale_density — Количество различных значений серого в области лица в диапазоне от 0 до 255;
- sharpness — Резкость изображения лица в диапазоне от 0 до 255;
- no_glasses — Уверенность в том, что человек не носит очки, в диапазоне от 0 до 1;
- no_dark_glasses — Уверенность в том, что человек не носит темные очки, в диапазоне от 0 до 1;

- no_beard — Уверенность в том, что человек не носит бороду, в диапазоне от 0 до 1;

- mouth_closed — Уверенность в том, что рот закрыт, в диапазоне от 0 до 1;
- male — Уверенность в том, что человек относится к мужскому полу, в диапазоне от 0 до 1;
- female — Уверенность в том, что человек относится к женскому полу, в диапазоне от 0 до 1;
- no_smile — Уверенность в том, что человек не улыбается, в диапазоне от 0 до 1;
- right_eye_open — Уверенность в том, что правый глаз открыт, в диапазоне от 0 до 1;
- left_eye_open — Уверенность в том, что левый глаз открыт, в диапазоне от 0 до 1;
- frontal_pose_deviation — Отклонение от фронтальной позы в диапазоне от 0 до 90 градусов;
- hot_spots — Уверенность в наличии бликов на лице в диапазоне от 0 до 1;
- angles — Отклонение от анфас (фронтальное положение) в трех углах: крен, тангаж, рыскание в диапазоне от -99 до 99 градусов.

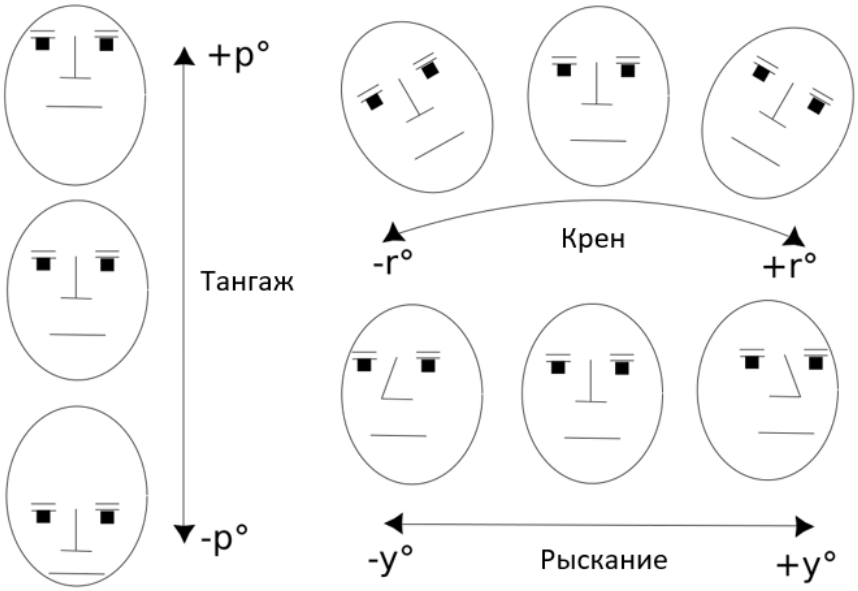

Рисунок 38: Визуализация углов крена, тангажа и рыскания

Пороговые рекомендации по некоторым характеристикам для получения изображений лиц хорошего качества для биометрического распознавания:

- deviation_from_uniform_lighting — 0.3;
- exposure — 100;
- hot_spots — 0.7;
- sharpness — 18.

Доказано, что метод **angles** работает корректно, если лицо отклоняется от фронтального положения не более чем на 20 градусов в любую сторону.

Метод **angles** точнее, но медленнее, чем функция estimation_deviation из модуля обнаружения.

Примеры использования

Для получения более подробной информации обратитесь к документации по коду.

```
FSDK::portrait::PortraitEngine portrait_engine(profile, device_idx);
FSDK::portrait::PortraitCharacteristics portrait_res =
    portrait_engine.compute_characteristics(img1, landmarks, faces[0].area);
```

Список 35: C++: Портретные характеристики

```
try (PortraitEngine portraitEngine = PortraitEngine.createPortraitEngine(profile);
     PortraitCharacteristics portraitCharacteristics =
            portraitEngine.computeCharacteristics(image, landmarks, faces[0].area)) {
    Boolean noGlassesDecision = portraitCharacteristics.noGlasses().decision;
}
```

Список 36: Java: Портретные характеристики

Оценка качества изображения лица

Краткий обзор

Face.SDK предоставляет функциональные возможности для оценки качества изображения лица, т. е. видимости ключевых областей лица, которые наиболее важны для распознавания лиц. Для оценки видимости областей лица **Face.SDK** предлагает интерфейс **FSDK::portrait::FaceQualityAssessor**, который предоставляет следующие оценки (все значения в диапазоне от 0 до 1, см. **FSDK::portrait::FaceQualityCharacteristics**):

face_visible — Уверенность в том, что данное изображение является изображением лица;

left_eyebrow_visible — Уверенность в том, что левая бровь видна на изображении лица;

right_eyebrow_visible — Уверенность в том, что правая бровь видна на изображении лица;

left_eye_visible — Уверенность в том, что левый глаз виден на изображении лица;

right_eye_visible — Уверенность в том, что правый глаз виден на изображении лица;

left_ear_visible — Уверенность в том, что левое ухо видно на изображении лица;

right_ear_visible — Уверенность в том, что правое ухо видно на изображении лица;

nose_visible — Уверенность в том, что нос виден на изображении лица;

teeth_visible — Уверенность в том, что зубы видны на изображении лица (т. е. человек на изображении улыбается или его рот открыт);

mouth_visible — Уверенность в том, что рот виден на изображении лица;

hat_visible — Уверенность в том, что человека на изображении носит головной убор.

Примеры использования

Для получения более подробной информации обратитесь к документации по коду.

```
FSDK::portrait::FaceQualityAssessor face_quality_assessor(profile, device_idx);
FSDK::portrait::FaceQualityCharacteristics face_quality_res =
    face_quality_assessor.estimate_face_areas_visibility(img1, landmarks);
```

Список 37: C++: Оценка качества изображения лица

```
try (FaceQualityAssessor faceQualityAssessor =
    FaceQualityAssessor.createFaceQualityAssessor(profile);
     FaceQualityCharacteristics faceQualityCharacteristics =
            faceQualityAssessor.estimateFaceAreasVisibility(image, landmarks)) {
    Boolean noseVisible = faceQualityCharacteristics.noseVisible().decision;
}
```

Список 38: Оценка качества изображения лица

Точность

Портретные характеристики

Все расчеты производятся на бэкэнде Caffe.

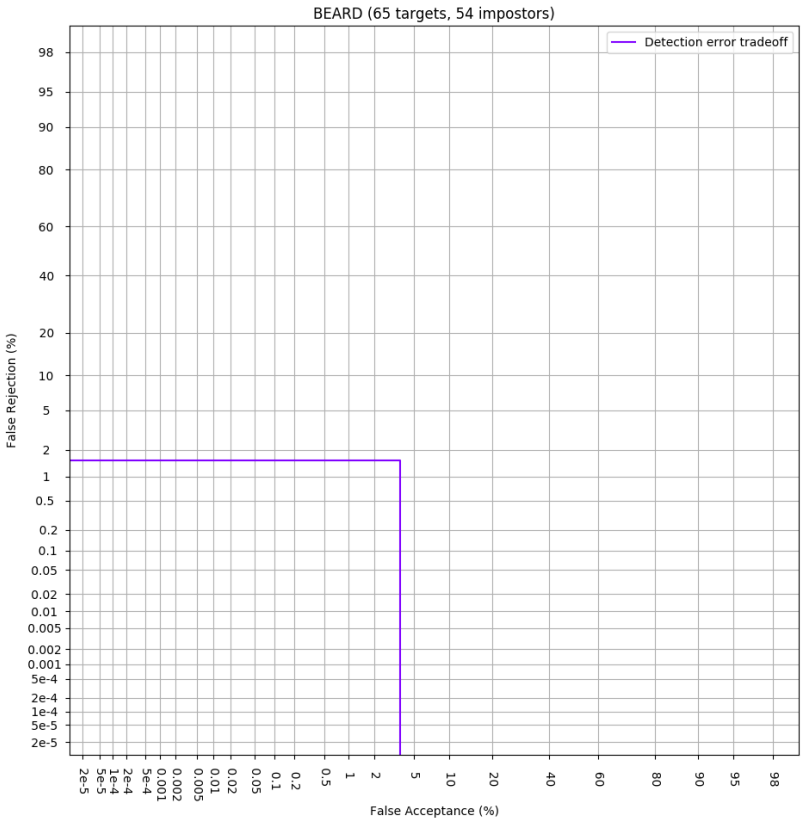

Рисунок 39: DET классификатора бороды

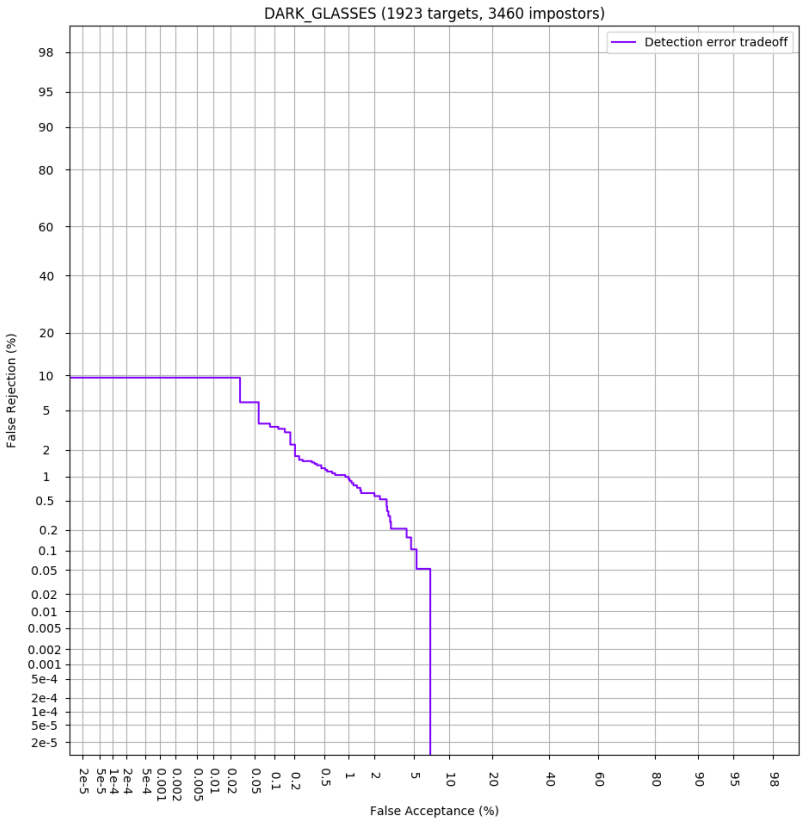

Рисунок 40: DET классификатора тёмных очков

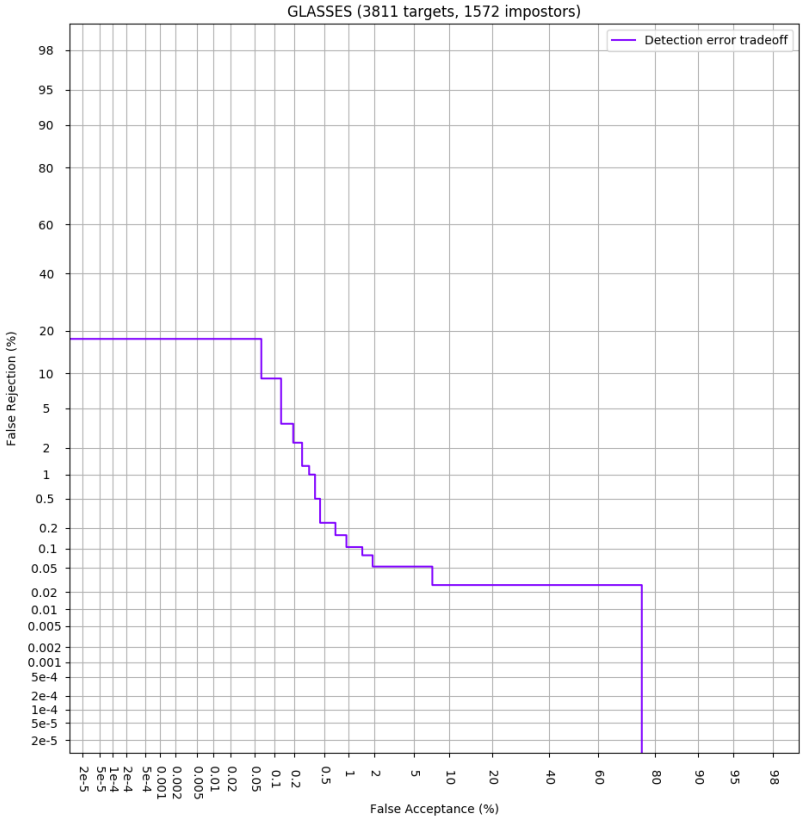

Рисунок 41: DET классификатора очков

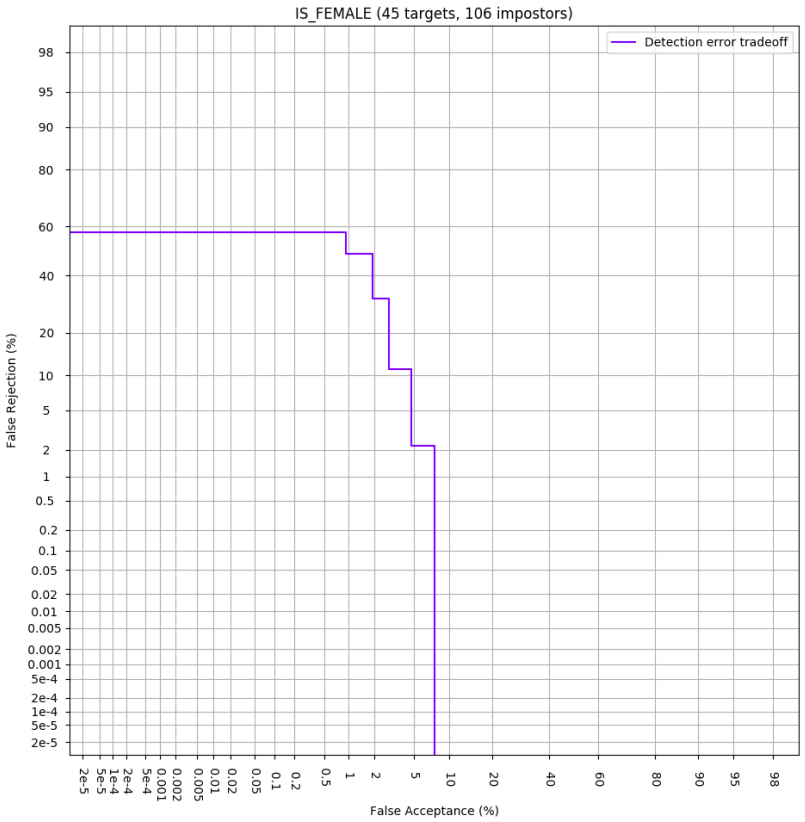

Рисунок 42: DET классификатора женского пола

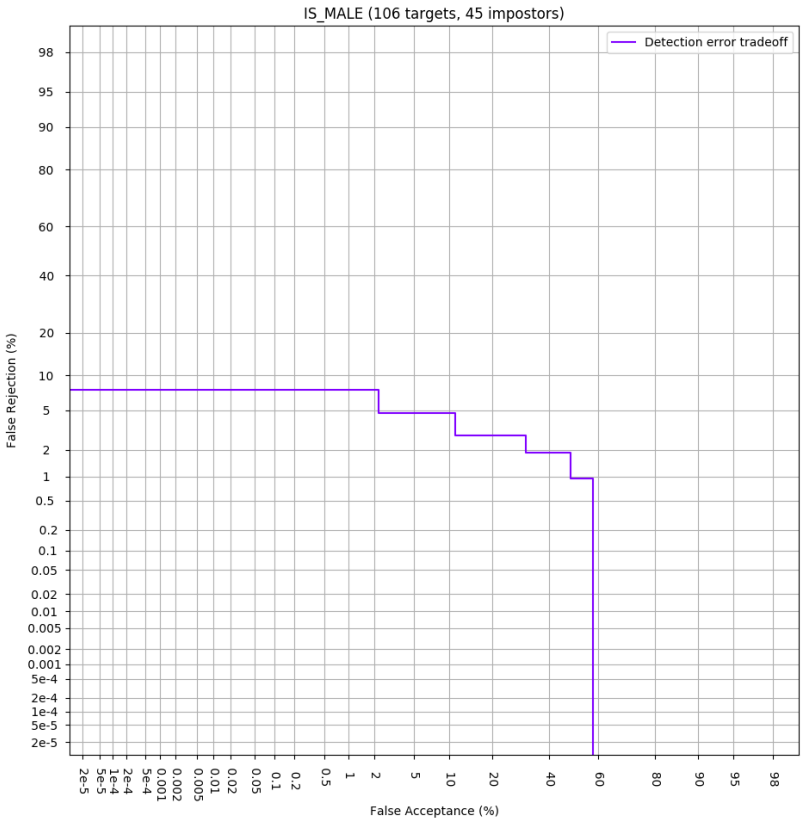

Рисунок 43: DET классификатора мужского пола

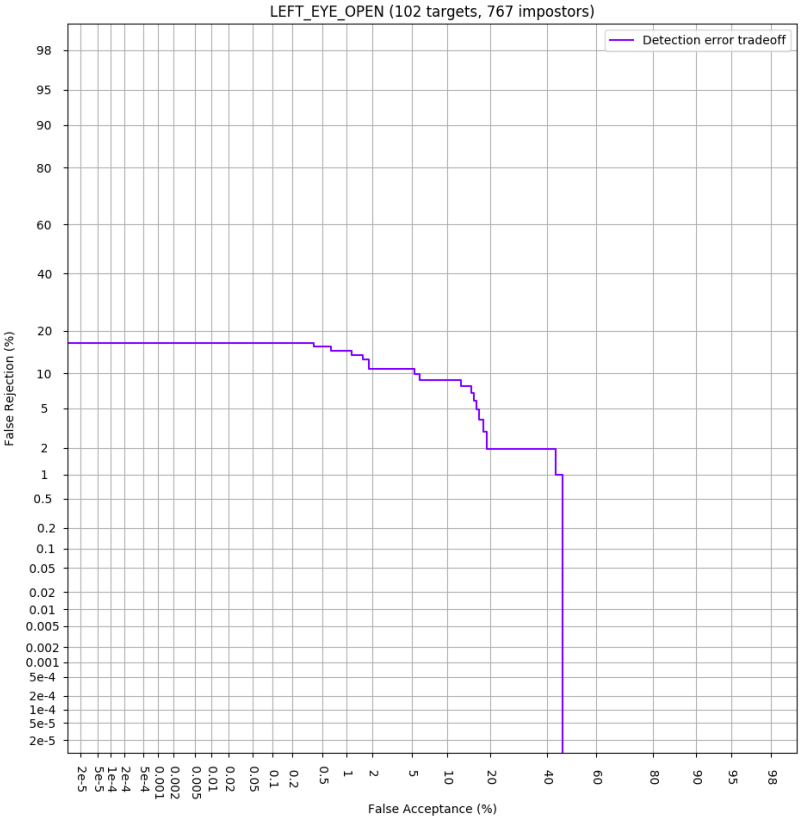

Рисунок 44: DET классификатора открытого левого глаза

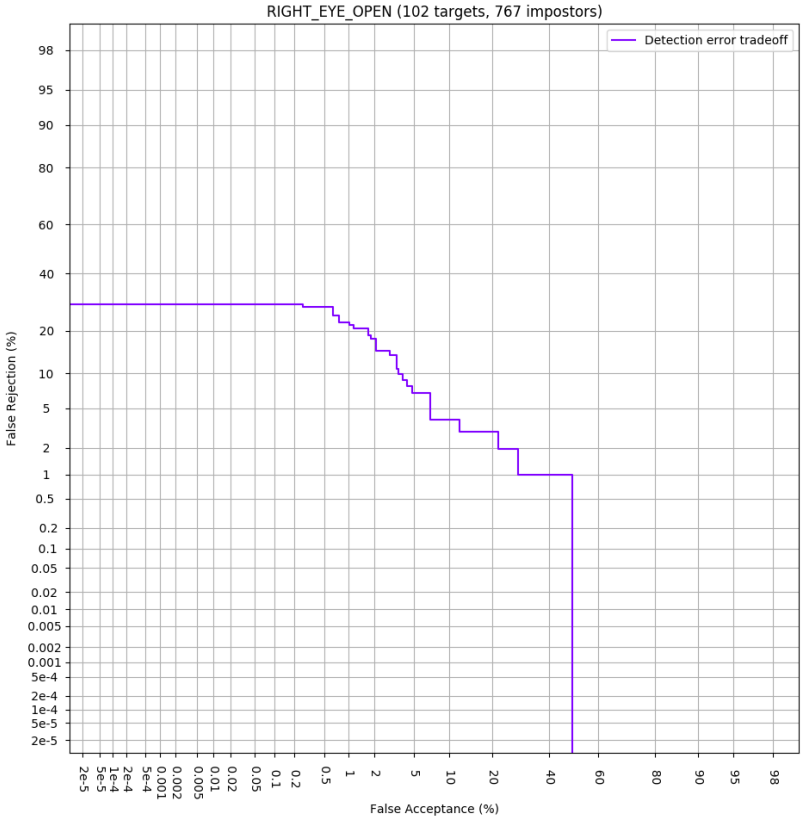

Рисунок 45: DET классификатора открытого правого глаза

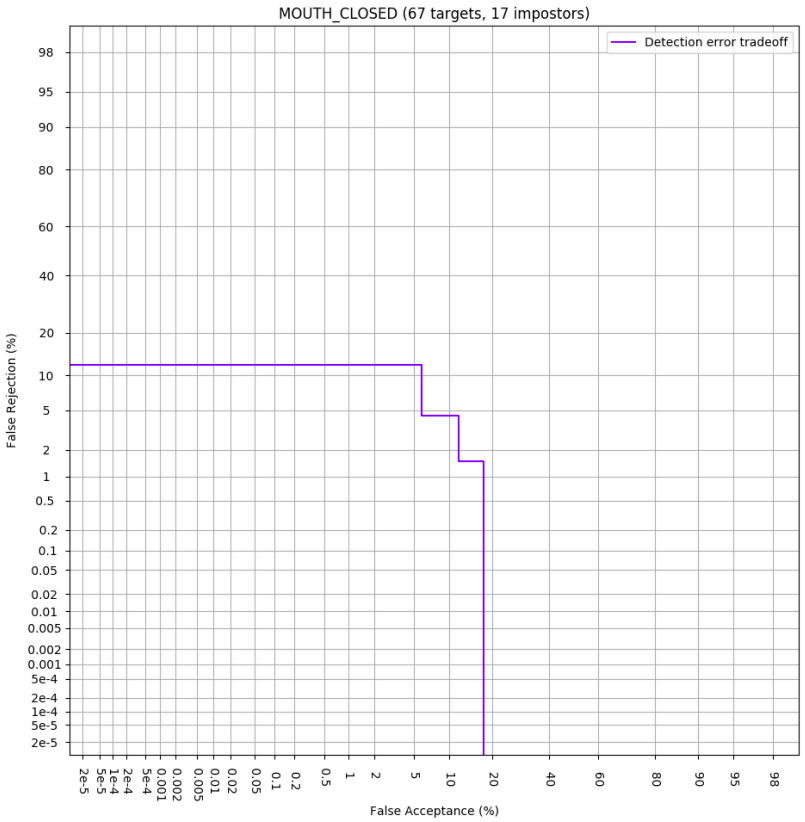

Рисунок 46: DET классификатора закрытого рта

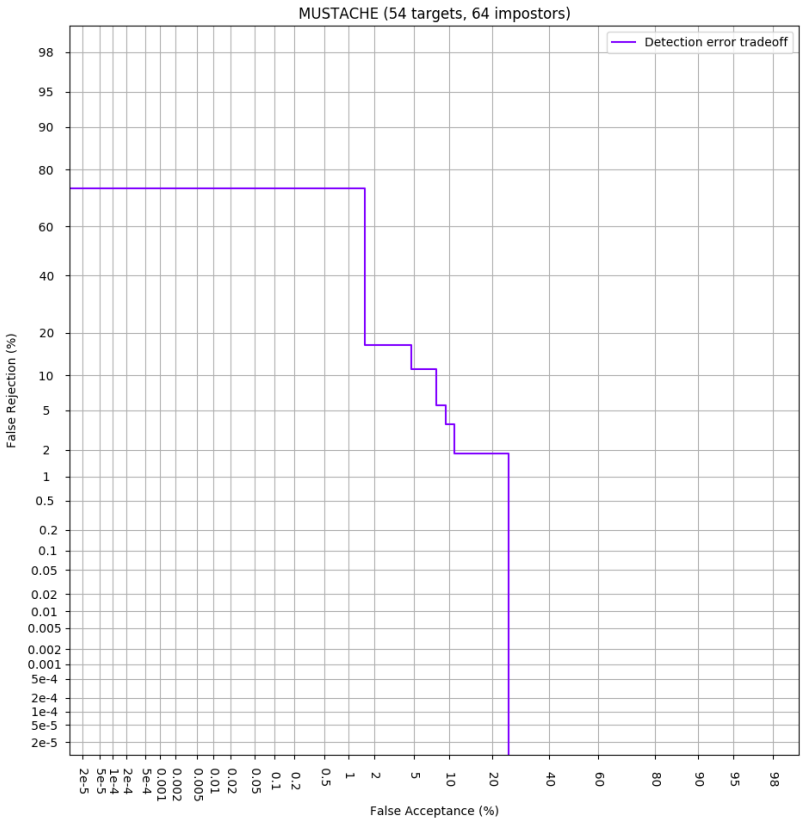

Рисунок 47: DET классификатора усов

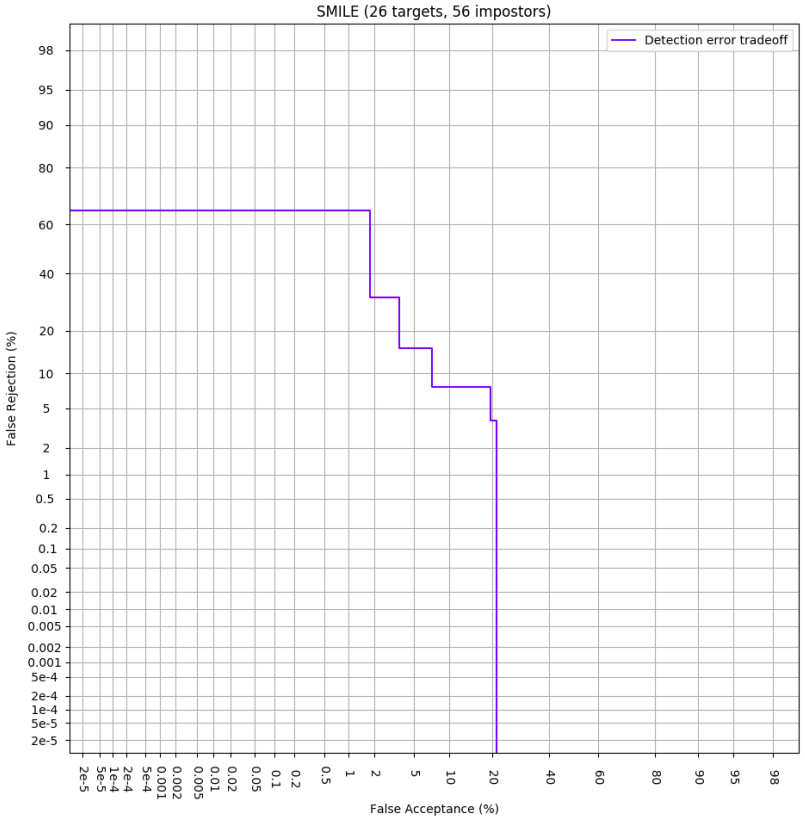

Рисунок 48: DET классификатора улыбки

Ниже вы можете увидеть графики, которые описывают значения ошибок для каждого угла отклонения лица от анфас на разных типах углов. Для построения каждого графика использовались только изображения с отклонением от положения анфас менее чем на 20 градусов по двум другим типам углов.

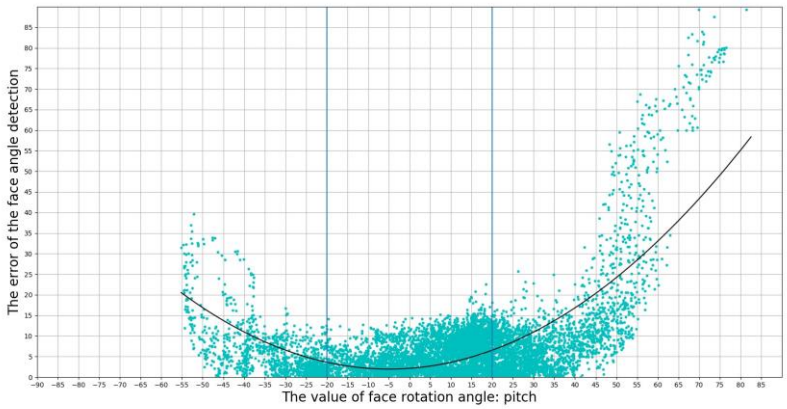

Рисунок 49: Распределение ошибок определения угла тангажа лица

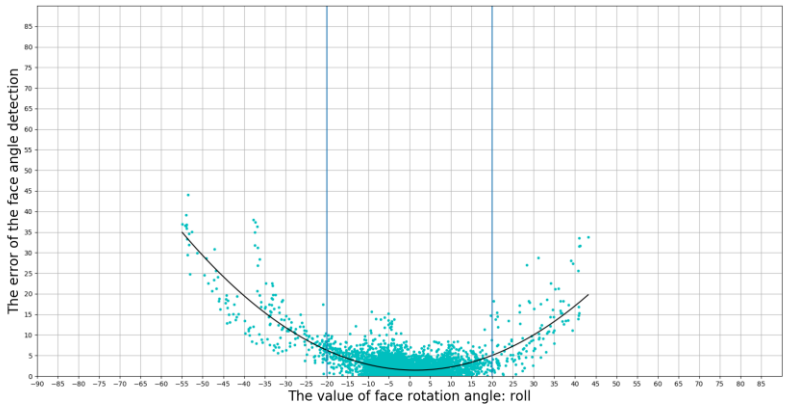

Рисунок 50: Распределение ошибок определения угла крена лица

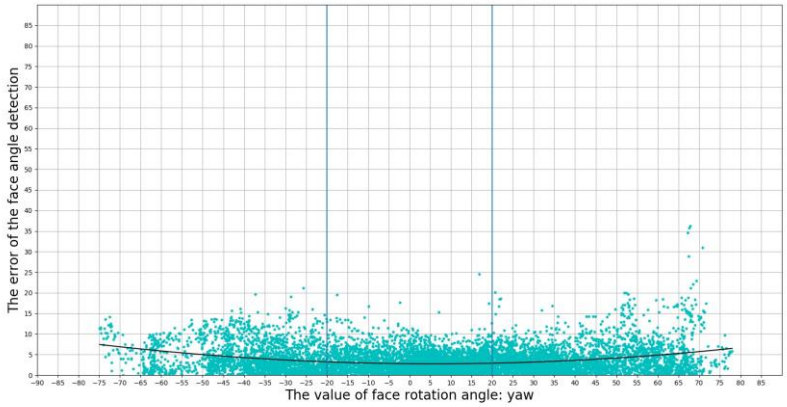

Рисунок 51: Распределение ошибок определения угла рыскания лица

Оценка качества изображения лица

Для оценки точности используется подмножество **LFW Face Database** с синтезированными помехами в области лица.

Примеры изображений:

Рисунок 52: Примеры тестового набора LFW с синтезированными помехами

В таблице 5 представлено качество оценки того, что входное изображение является изображением лица с качеством, подходящим для распознавания, то есть содержит вид лица спереди и видны следующие области:

- рот,
- нос,
- левый и правый глаза,
- левая и правая брови.

Следующие пороговые значения использовались для оценки **FSDK::portrait::FaceQualityCharacteristics**:

- no_face_pixels_rate_threshold = 0.35;
- no_mouth_pixels_rate_threshold = 0.011;
- no_nose_pixels_rate_threshold = 0.027;
- no_eye_pixels_rate_threshold = 0.0014;
- no_eye_brow_pixels_rate_threshold = 0.0015.

Таблица 5: Качество оценки качества лица

качество	FAR	FRR
0.912	0.04625	0.255

ПРОВЕРКА ЛИЦА ЖИВОГО ЧЕЛОВЕКА

Краткий обзор

В настоящее время мультимодальные системы биометрической верификации стремительно развиваются и демонстрируют впечатляющую производительность. Однако такие системы нуждаются в дополнительной защите от спуфинг-атак. Мы представляем метод анти-спуфинга в двух разных вариантах:

- бимодальный аудиовизуальный;
- по фотографии (один кадр).

Бимодальное обнаружение живости

Этот метод позволяет оценить параметры качества последовательности изображений лиц в процессе проверки. На основе этих параметров решается, подходят ли данные для обработки стандартным методом (аудиовизуальное обнаружение живости на основе реперных точек, FALD) или нет. Обнаружение живости доступно на 4 языках:

- русский;
- английский;
- португальский;
- корейский.

На вашем компьютере должен быть установлен **VoiceKey SDK** для корректного использования метода.

Обнаружение живости на фотографии

Этот метод основан на современном DNN-подходе и требует только одного кадра (возможно, содержащего реальное лицо) для расчета живости.

Обратите внимание, что данный алгоритм находится только на стадии прототипа, поэтому он недостаточно протестирован и в некоторых случаях может работать некорректно.

Существует утилита **photo_antispoofing_quality**, использующая алгоритм живости фотографий, основная цель которого – это обработка предоставленной базы, построение графиков FR/FA и некоторой дополнительной информации, такой как среднее время выполнения алгоритма.

Утилита **util_photo_antispoofing_quality.py** написана на Python и находится в каталоге util_photo_antispoofing_quality внутри каталога bin для вашей платформы. Для ее запуска должна быть установлена 64-битная версия интерпретатора Python 3.6, а также зависимости утилиты из файла requirements.txt, расположенного в том же каталоге, что и сама утилита. Зависимости могут быть установлены с помощью **pip**, который обычно входит в стандартный дистрибутив Python. Для получения дополнительной информации смотри руководство пользователя **pip**: https://pip.pypa.io/en/stable/user_guide/.

Описание параметров и выходных данных смотри в руководстве утилиты, доступном с параметром **-h.**

Рекомендуется использовать профили «progressive_4_7_quantized» и «progressive_4_7» для ЦП и ГП соответственно для достижения максимальной производительности канала обнаружения живости фотографии.

Примеры использования

Бимодальное обнаружение живости

```
FSDK::core_types::LivenessAlgoType type =
    FSDK::core_types::LivenessAlgoType::TEXT_DEPENDENT;
FSDK::core_types::Language language      = FSDK::core_types::Language::ENG;
FSDK::liveness::LivenessDetector bimodalLD(profile, type, language);
FSDK::liveness::LDFeatures ldFeatures(profile);
FSDK::containers::ByteArray voice_array;
voice_array = FSDK::containers::ByteArray(); // загрузите голосовые характеристики

int video_fps = 15; // Кадров в секунду
int frame_num = 0;
while (true) { // до конца видео
    const FSDK::containers::Image current_frame = get_next_image_from_video();
    if (!current_frame.is_initialized()) {
        break;
    }
    uint32_t timestamp                     = uint32_t(frame_num * (1000. / video_fps));
    FSDK::portrait::LMDetectionIssues res = ldFeatures.append_image(img1, timestamp);
    if ((res & FSDK::portrait::LMDetectionIssues::BadFacePosition) ==
        FSDK::portrait::LMDetectionIssues::BadFacePosition) {
        // отрегулируйте, если лицо расположено неверно
    }
    // TODO: проверьте на наличие других
    проблем
    frame_num++;
}
ldFeatures.finalize(voice_array);
const FSDK::containers::ByteArray serialized_ldfeatures = ldFeatures.serialize();
FSDK::liveness::LDFeaturesData ldfeatures_data(serialized_ldfeatures);
bool compensate                    = true; // нужно ли активировать компенсацию
FSDK::containers::Decision result = bimodalLD.calc_liveness(ldfeatures_data,
    voice_array, compensate);
```

Список 39: C++: бимодальное обнаружение живости

```
LivenessAlgoType livenessType = LivenessAlgoType.TEXT_DEPENDENT;
Language language = Language.ENG;
try (LivenessDetector livenessDetector = LivenessDetector
        .createLivenessDetector(profile, livenessType, language);
     LDFeatures ldFeatures = LDFeatures.createLDFeatures(profile)) {
    byte[] voiceFeatures =
       Files.readAllBytes(FileSystems.getDefault().getPath("vk.features"));
    int frameNum = 0, fps = 15;
    while (true) { // до конца видео
        try (Image currentFrame = getNextImageFromVideo()) {
            if (!currentFrame.isInitialized()) {
                break;
            }
            int timestamp = (int) (frameNum * (1000. / fps));
            EnumSet<LMDetectionIssues> issues = ldFeatures.appendImage(currentFrame,
                timestamp);
            // TODO: проверьте на наличие
            проблем
            frameNum++;
        }
    }
    ldFeatures.finalize(voiceFeatures);
    byte[] serializedLdFeatures = ldFeatures.serialize();
    Decision result;
    try (LDFeaturesData ldFeaturesData =
        LDFeaturesData.createLDFeaturesData(serializedLdFeatures)) {
        boolean compensate = true;
        result = livenessDetector.calcLiveness(ldFeaturesData, voiceFeatures,
            compensate);
    }
} catch (IOException e) {
    e.printStackTrace();
```

Список 40: Java: бимодальное обнаружение живости

Обнаружение живости фотографии

```
FSDK::liveness::PhotoLivenessDetector photo_liveness_detector(profile, device_idx);

if (!faces.empty()) {
    FSDK::detection::LandmarkDetector lm_detector(profile);
    FSDK::containers::landmarks_list_t face_landmarks =
        lm_detector.detect(img1, faces[0].area); // возьмите первое лицо из
    детектора
        FSDK::containers::Decision photo_liveness_result =
        photo_liveness_detector.calc_liveness(img1, face_landmarks, faces[0].area);
```

Список 41: C++: обнаружение живости фотографии

```
DetectionResult[] faces = faceDetector.detect(image, −1, −1, new Area());

int gpuDeviceId = −1;
int batchSize = 1;
try (PhotoLivenessDetector photoLivenessDetector =
    PhotoLivenessDetector.createPhotoLivenessDetector(profile, gpuDeviceId, batchSize)) {
    if (faces.length > 0) {
        PointF[] landmarks = lmDetector.detect(image, faces[0].area); // возьмите
            первое лицо из детектора
        Decision photoLivenessResult = photoLivenessDetector.calcLiveness(image,
            landmarks, faces[0].area);
    }
}
```

Список 42: Java: обнаружение живости фотографии

Точность

Точность бимодального обнаружения живости

Все расчеты производятся на бэкэнде Caffe.

Таблица 6: Качество бимодального обнаружения живости

Язык	EER	Компенсация	Порог
рус	9.4%	да	0.68
рус	5%	нет	0.5
англ	8.4%	да	0.6
англ	3.4%	нет	0.51
кор	11.36%	да	0.67
кор	7.56%	нет	0.54
порт	9,61%	да	0.6
порт	5.61%	нет	0.51

Точность обнаружения живости на фотографии

Все расчеты производятся на бэкэнде PT.

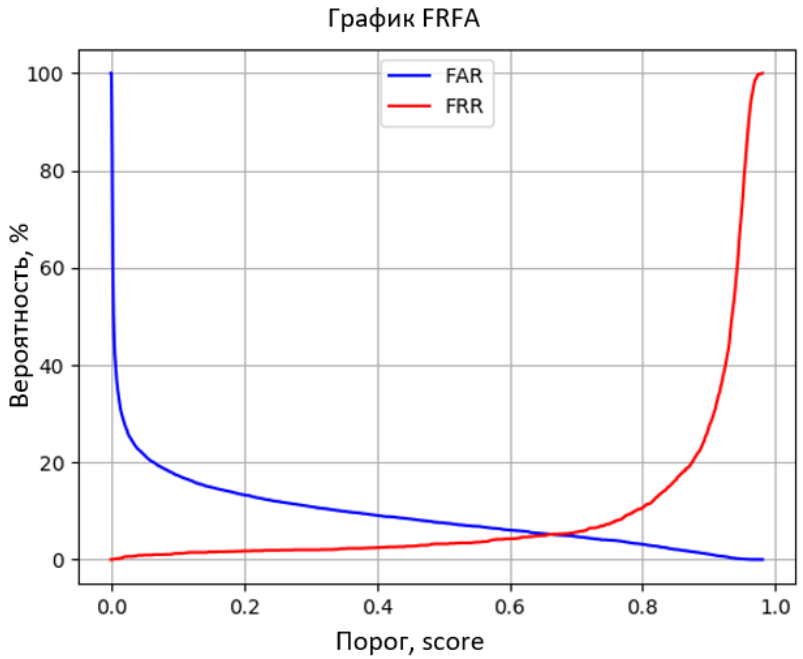

Рисунок 53: Живость фотографии FRFA

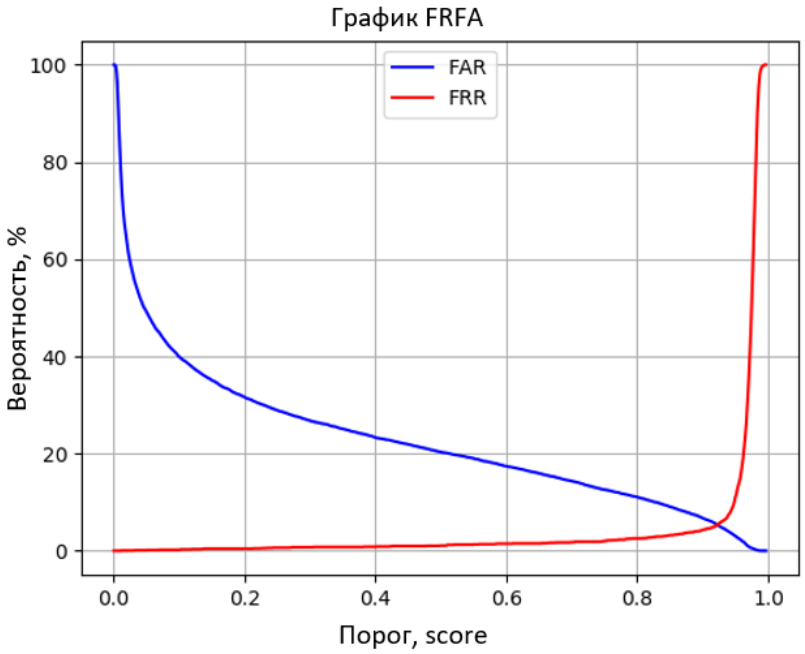

Рисунок 54: Живость фотографии (с классификацией атак) FRFA

Аппаратное обеспечение

Таблица 7: Производительность бимодального обнаружения живости

Аппаратное обеспечение	RT
ЦП i5-2500 @ 3.30ГГц	5
Nexus 4	1

ПРИЛОЖЕНИЕ 1: ПРОИЗВОДИТЕЛЬНОСТЬ

Обнаружение

Статистика производительности для метода обнаружения **FaceDetector** приведена в таблицах ниже. Для тестов использовались изображения с разрешением 1920x1080. Минимальный размер лица = 40 пикселей.

Таблица 8: FaceDetector, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Лица		Cascade	SSD				
			caffe	opencv			
		потоки					
		1	1	1	2	4	8
0	время, мс	122.885	150.252	87.046	49.005	43.384	41.256
1		82.533	146.134	75.936	54.356	40.659	41.152
2		122.153	146.171	74.557	53.101	40.854	47.339
3		91.977	145.980	86.612	54.426	41.177	42.988
4		108.648	148.613	76.840	49.181	41.362	52.629
6		148.945	151.109	75.890	53.542	42.530	40.967
10		121.107	151.581	75.338	49.527	42.688	44.122
0	ЦП, %	83	7	5	9	17	28
1		79	7	5	10	17	28
2		88	7	5	10	17	29
3		86	7	5	10	16	28
4		84	7	6	9	17	30
6		85	7	5	10	16	28
10		83	7	6	9	16	28
0	ОЗУ, КБ	0.000	1195.000	3568.000	3539.000	3544.000	3363.000
1		0.000	1073.000	3307.000	3373.000	3314.000	3346.000
2		9.000	1205.000	3307.000	3373.000	3297.000	3297.000
3		10.000	1129.000	3502.000	3495.000	3505.000	3314.000
4		7.000	1139.000	3505.000	3349.000	3505.000	3324.000
6		210.000	1139.000	3505.000	3495.000	3539.000	3495.000
10		139.000	1139.000	3304.000	3505.000	3324.000	3495.000

Таблица 9: FaceDetector, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	88.349	73.673	63.254	60.563
1		89.936	72.107	62.999	69.423
2		85.507	78.072	62.761	62.351
3		88.410	69.026	65.667	68.940
4		87.656	69.525	62.482	69.868
6		86.182	73.113	63.224	69.427
10		86.693	71.265	67.085	65.072
0	ЦП, %	10	15	25	44
1		10	15	25	43
2		10	15	25	43
3		10	16	25	43
4		10	15	25	43
6		10	15	25	42
10		10	15	24	42
0	ОЗУ, КБ	636.000	793.000	219.000	2129.000
1		569.000	5.000	3012.000	667.000
2		570.000	723.000	124.000	2746.000
3		1081.000	3035.000	192.000	500.000
4		629.000	3101.000	2751.000	518.000
6		2079.000	9.000	3084.000	633.000
10		2079.000	356.000	540.000	3232.000

Таблица 10: FaceDetector, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	35.743	33.290	32.926	37.375
1		36.367	32.700	32.295	38.662
2		35.975	32.600	33.201	36.966
3		35.867	32.218	29.197	39.532
4		35.270	31.788	32.395	38.771
6		34.109	31.151	31.498	38.648
10		38.530	33.988	39.312	41.311
0	ЦП, %	6	10	19	40
1		6	10	19	38
2		6	10	20	40
3		6	10	20	38
4		6	10	20	39
6		5	10	20	39
10		5	10	18	36
0	ОЗУ, КБ	1913.000	1573.000	1767.000	1479.000
1		1848.000	1503.000	1421.000	1880.000
2		1848.000	1504.000	1424.000	1877.000
3		1452.000	1503.000	1420.000	1881.000
4		1452.000	1835.000	1414.000	1473.000
6		1854.000	1905.000	1418.000	1491.000
10		1854.000	1899.000	1693.000	1869.000

Таблица 11: FaceDetector, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Лица		SSD-Retina-quantized			
		pt			
		потоки			
		1	2	4	8
0	время, мс	65.219	59.991	45.712	51.669
1		67.252	54.823	50.187	51.566
2		65.875	56.545	48.105	50.846
3		66.734	56.554	50.082	51.320
4		64.816	59.487	48.456	52.107
6		68.959	62.000	51.294	52.629
10		66.764	63.204	52.380	52.827
0	ЦП, %	11	16	27	44
1		11	17	26	43
2		11	17	27	44
3		11	17	26	43
4		11	16	26	43
6		11	16	26	43
10		11	16	25	42
0	ОЗУ, КБ	3711.000	1972.000	2814.000	2787.000
1		3784.000	2929.000	2730.000	2712.000
2		3718.000	2668.000	3059.000	2752.000
3		3651.000	2612.000	2574.000	2912.000
4		3189.000	3401.000	3811.000	2356.000
6		3711.000	3010.000	2742.000	2616.000
10		3579.000	3416.000	3663.000	3247.000

Таблица 12: FaceDetector, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

Лица		Cascade	SSD				
			caffe	opencv			
		потоки					
		1	1	1	2	4	8
0	время, мс	38.100	28.816	79.529	58.918	50.709	42.641
1		33.136	24.390	79.622	57.765	48.915	52.017
2		39.080	27.393	82.709	57.946	48.802	42.441
3		34.800	30.818	87.371	57.659	50.573	43.783
4		37.559	25.732	81.873	57.404	48.120	43.228
6		41.147	25.472	80.496	58.066	49.143	44.919
10		42.313	29.847	80.592	58.967	50.631	44.163
0	ЦП, %	32	13	6	10	17	28
1		35	11	6	10	17	31
2		31	12	6	10	16	28
3		33	13	6	10	17	28
4		31	11	6	10	17	28
6		30	11	6	10	16	28
10		29	12	6	10	16	27
0	ОЗУ, КБ	1024.000	198.000	3231.000	3192.000	3192.000	3182.000
1		461.000	0.000	3202.000	3192.000	3182.000	3182.000
2		461.000	0.000	3182.000	3192.000	3165.000	3182.000
3		461.000	66.000	3190.000	3165.000	3182.000	3182.000
4		461.000	40.000	3192.000	3182.000	3165.000	3182.000
6		461.000	66.000	3192.000	3192.000	3182.000	3182.000
10		461.000	154.000	3192.000	3192.000	3182.000	3182.000
0	ГП, %	59	53	0	0	0	0
1		51	58	0	0	0	0
2		60	54	0	0	0	0
3		52	49	0	0	0	0
4		55	57	0	0	0	0
6		58	57	0	0	0	0
10		53	51	0	0	0	0

0	Память ГП, КБ	2048.000	0.000	0.000	0.000	0.000	0.000
1		2048.000	0.000	0.000	0.000	0.000	0.000
2		2048.000	0.000	0.000	0.000	0.000	0.000
3		2048.000	0.000	0.000	0.000	0.000	0.000
4		2048.000	0.000	0.000	0.000	0.000	0.000
6		2048.000	0.000	0.000	0.000	0.000	0.000
10		2048.000	0.000	0.000	0.000	0.000	0.000

Таблица 13: FaceDetector, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	24.098	20.301	19.753	19.025
1		23.653	22.137	19.817	18.465
2		23.996	21.252	20.516	19.651
3		24.081	19.749	21.274	19.392
4		24.982	20.826	20.313	20.077
6		25.513	20.662	20.202	19.308
10		27.187	22.808	22.587	19.937
0	ЦП, %	14	18	22	31
1		14	18	23	31
2		14	17	22	30
3		14	18	22	30
4		14	17	22	31
6		14	18	22	31
10		15	17	22	30
0	ОЗУ, КБ	3061.000	2359.000	2293.000	2761.000
1		2227.000	3019.000	4483.000	3684.000
2		3068.000	2273.000	3936.000	2872.000
3		3842.000	4288.000	3121.000	3058.000
4		3397.000	2997.000	2911.000	5289.000
6		5277.000	5265.000	5275.000	3048.000
10		4551.000	777.000	2889.000	3092.000
0	ГП, %	25	29	30	30
1		25	26	30	32
2		25	28	29	29
3		25	30	28	30
4		24	28	28	29
6		24	27	28	30
10		22	27	27	30

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Таблица 14: FaceDetector, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	18.063	16.139	15.157	14.599
1		15.269	14.217	15.012	14.895
2		15.487	15.349	14.375	14.441
3		15.726	15.931	14.995	14.818
4		15.041	14.431	14.406	14.030
6		17.455	13.996	14.471	14.606
10		18.117	16.963	17.256	17.191
0	ЦП, %	6	8	13	24
1		6	8	13	23
2		6	8	13	24
3		6	8	14	24
4		6	8	13	23
6		6	8	13	25
10		6	8	12	21
0	ОЗУ, КБ	71.000	65.000	65.000	1104.000
1		330.000	2726.000	0.000	2720.000
2		0.000	2275.000	1621.000	1434.000
3		1351.000	1483.000	3033.000	1491.000
4		0.000	2910.000	1285.000	66.000
6		0.000	3050.000	1221.000	1228.000
10		0.000	2391.000	66.000	2456.000
0	ГП, %	23	26	27	27
1		26	28	27	28
2		25	27	27	27
3		27	26	27	27
4		27	27	28	28
6		24	27	27	27
10		23	24	23	23

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Таблица 15: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Лица			Cascade	SSD				
				caffe	opencv			
			потоки					
			1	1	1	2	4	8
0	время, мс		173.259	185.317	91.339	94.597	70.732	59.989
1			124.840	179.893	90.981	93.667	70.238	61.067
2			197.679	181.632	91.447	94.216	70.641	58.608
3			149.357	179.947	93.187	94.469	71.170	59.588
4			171.207	182.282	93.200	94.277	70.831	58.716
6			206.589	184.579	93.105	94.739	71.510	59.384
10			182.606	184.113	93.192	95.807	76.559	59.633
0	ЦП, %		67	6	5	8	13	23
1			62	6	5	8	12	22
2			67	6	5	8	13	22
3			65	6	5	8	13	22
4			65	6	6	8	12	21
6			67	6	5	8	13	22
10			67	6	5	8	12	21
0	ОЗУ, КБ		1774.000	421.000	147.000	111.000	280.000	444.000
1			1682.000	330.000	16.000	106.000	0.000	573.000
2			1585.000	355.000	66.000	117.000	97.000	380.000
3			1401.000	282.000	89.000	19.000	245.000	369.000
4			1474.000	355.000	63.000	100.000	265.000	294.000
6			1579.000	228.000	22.000	77.000	329.000	367.000
10			1651.000	883.000	93.000	68.000	201.000	107.000

Таблица 16: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	190.621	139.034	125.609	137.128
1		182.899	132.910	125.778	135.255
2		181.056	139.727	128.646	136.752
3		181.007	139.020	127.855	124.077
4		184.016	138.237	128.771	137.671
6		185.885	140.401	129.273	127.914
10		185.351	139.206	129.993	131.008
0	ЦП, %	6	11	21	41
1		6	11	21	41
2		6	11	21	41
3		6	11	21	41
4		6	11	21	41
6		6	11	21	41
10		6	11	21	41
0	ОЗУ, КБ	519.000	646.000	514.000	511.000
1		457.000	597.000	465.000	454.000
2		458.000	466.000	465.000	461.000
3		466.000	466.000	462.000	456.000
4		461.000	462.000	467.000	599.000
6		464.000	589.000	466.000	584.000
10		590.000	467.000	462.000	462.000

Таблица 17: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	58.534	52.766	52.419	57.102
1		57.344	51.330	52.455	57.278
2		58.158	52.306	51.412	56.153
3		57.360	51.336	51.286	56.351
4		57.787	51.786	51.547	56.246
6		58.259	51.445	51.083	55.962
10		62.719	56.206	55.856	60.705
0	ЦП, %	5	10	21	41
1		5	10	21	40
2		5	11	21	41
3		6	11	20	41
4		6	11	21	40
6		6	11	21	41
10		5	11	21	40
0	ОЗУ, КБ	603.000	737.000	741.000	611.000
1		534.000	553.000	661.000	650.000
2		533.000	539.000	577.000	537.000
3		527.000	522.000	525.000	551.000
4		531.000	534.000	545.000	523.000
6		650.000	534.000	534.000	664.000
10		537.000	661.000	557.000	529.000

Таблица 18: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Лица		SSD-Retina-quantized			
		pt			
		потоки			
		1	2	4	8
0	время, мс	107.902	85.461	77.871	82.237
1		107.615	84.129	79.765	83.708
2		107.836	84.636	78.975	84.248
3		107.247	83.830	79.256	83.051
4		108.241	84.681	79.745	85.530
6		108.707	84.799	79.308	85.554
10		109.583	85.301	79.948	86.189
0	ЦП, %	6	11	21	41
1		6	11	21	41
2		6	11	21	41
3		6	11	21	41
4		6	11	21	41
6		6	11	21	41
10		6	11	21	41
0	ОЗУ, КБ	387.000	414.000	400.000	651.000
1		345.000	340.000	481.000	572.000
2		336.000	345.000	390.000	575.000
3		349.000	343.000	405.000	596.000
4		349.000	353.000	348.000	589.000
6		477.000	340.000	378.000	601.000
10		348.000	349.000	511.000	667.000

Table 19: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

Лица		Cascade	SSD				
			caffe	opencv			
		потоки					
		1	1	1	2	4	8
0	время, мс	44.477	35.988	92.002	93.307	69.259	57.742
1		41.343	36.387	91.575	93.286	69.259	56.897
2		45.803	36.397	92.404	94.848	70.201	56.539
3		42.319	36.666	92.849	94.278	69.759	57.688
4		44.643	36.902	92.570	93.950	69.807	58.025
6		47.556	37.278	93.548	96.349	70.186	57.610
10		51.064	37.888	93.433	95.404	70.580	58.501
0	ЦП, %	9	6	5	8	13	22
1		9	6	5	8	13	22
2		12	6	5	8	12	22
3		10	6	5	8	12	22
4		13	6	5	8	12	21
6		11	6	5	8	12	21
10		7	6	5	8	13	22
0	ОЗУ, КБ	1499.000	187.000	170.000	157.000	424.000	601.000
1		1559.000	99.000	146.000	163.000	434.000	439.000
2		1805.000	122.000	161.000	155.000	392.000	374.000
3		1449.000	126.000	152.000	130.000	187.000	606.000
4		1954.000	123.000	170.000	163.000	358.000	445.000
6		1663.000	225.000	177.000	157.000	503.000	402.000
10		1445.000	120.000	228.000	238.000	233.000	387.000
0	ГП, %	51	36	0	0	0	0
1		43	43	0	0	0	0
2		52	44	0	0	0	0
3		46	46	0	0	0	0
4		49	47	0	0	0	0
6		51	43	0	0	0	0
10		39	35	0	0	0	0

0	Память ГП, КБ	2048.000	0.000	0.000	0.000	0.000	0.000
1		2048.000	0.000	0.000	0.000	0.000	0.000
2		2048.000	0.000	0.000	0.000	0.000	0.000
3		2048.000	0.000	0.000	0.000	0.000	0.000
4		2048.000	0.000	0.000	0.000	0.000	0.000
6		2048.000	0.000	0.000	0.000	0.000	0.000
10		2048.000	0.000	0.000	0.000	0.000	0.000

Таблица 20: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	38.354	33.890	31.007	30.542
1		39.040	33.963	31.547	31.607
2		38.823	33.626	31.755	31.482
3		39.415	34.087	31.995	31.852
4		39.522	34.193	32.018	31.907
6		39.604	34.380	32.547	32.059
10		40.119	35.086	32.847	32.741
0	ЦП, %	6	11	21	41
1		6	11	20	41
2		6	11	21	41
3		6	11	21	40
4		6	11	21	41
6		6	11	21	41
10		6	11	21	41
0	ОЗУ, КБ	91.000	296.000	280.000	478.000
1		29.000	173.000	213.000	287.000
2		221.000	304.000	269.000	29.000
3		291.000	308.000	302.000	30.000
4		223.000	30.000	220.000	251.000
6		214.000	222.000	264.000	31.000
10		204.000	225.000	28.000	307.000
0	ГП, %	18	19	22	22
1		13	19	17	22
2		16	19	16	17
3		13	19	22	22
4		17	19	15	22
6		12	19	20	22
10		16	19	15	20

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Таблица 21: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	26.551	25.426	25.508	25.776
1		26.381	25.210	25.193	25.745
2		26.267	25.193	25.073	25.901
3		26.257	25.121	25.597	25.637
4		25.909	24.889	24.623	25.038
6		25.822	25.027	24.879	25.274
10		30.268	29.275	29.542	30.098
0	ЦП, %	5	10	20	40
1		5	10	20	40
2		5	10	20	40
3		5	10	20	40
4		5	10	20	40
6		5	10	20	40
10		5	10	20	40
0	ОЗУ, КБ	841.000	840.000	921.000	859.000
1		783.000	606.000	839.000	414.000
2		822.000	608.000	543.000	563.000
3		599.000	841.000	405.000	780.000
4		781.000	778.000	774.000	776.000
6		396.000	778.000	778.000	648.000
10		793.000	832.000	769.000	767.000
0	ГП, %	16	16	16	16
1		16	16	16	16
2		16	16	16	16
3		16	12	16	16
4		16	12	17	16
6		12	16	16	16
10		14	14	11	14

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Таблица 22: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Лица		Cascade	SSD				
			caffe	opencv			
		потоки					
		1	1	1	2	4	8
0	время, мс	146.518	114.308	54.870	56.073	38.950	33.706
1		104.500	110.004	55.925	55.614	39.324	33.670
2		166.950	110.062	55.716	55.510	40.059	33.062
3		122.008	109.097	55.980	55.926	39.400	33.844
4		144.218	111.813	55.876	55.738	39.898	33.448
6		170.796	112.791	56.209	56.787	40.007	33.717
10		158.898	111.079	55.568	56.943	40.724	33.972
0	ЦП, %	83	13	13	19	30	48
1		78	13	13	19	31	49
2		85	13	13	20	30	49
3		85	13	13	19	31	48
4		82	13	13	19	29	49
6		85	13	13	19	30	49
10		81	13	13	19	29	48
0	ОЗУ, КБ	0.000	218.000	0.000	0.000	0.000	0.000
1		0.000	309.000	0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000	0.000	0.000
4		0.000	197.000	0.000	0.000	0.000	0.000
6		0.000	241.000	0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000	0.000	0.000

Таблица 23: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	114.326	78.904	65.336	82.213
1		111.698	78.529	66.410	81.344
2		115.195	78.334	66.709	82.127
3		116.240	84.572	64.867	78.779
4		108.844	84.172	65.862	82.259
6		117.566	79.107	67.719	82.422
10		117.708	79.759	67.350	78.617
0	ЦП, %	13	26	51	101
1		13	26	51	100
2		13	26	51	100
3		13	26	51	100
4		13	26	51	100
6		13	26	51	100
10		13	26	51	100
0	ОЗУ, КБ	0.000	0.000	0.000	505.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		459.000	473.000	460.000	0.000
4		0.000	457.000	0.000	453.000
6		454.000	0.000	452.000	456.000
10		0.000	0.000	0.000	0.000

Таблица 24: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	36.131	32.893	29.257	33.821
1		35.116	31.538	28.850	33.031
2		35.693	30.621	29.099	32.623
3		36.003	30.789	29.530	32.891
4		34.544	31.567	29.939	32.729
6		34.736	31.441	29.392	32.718
10		37.954	33.358	33.553	36.490
0	ЦП, %	13	26	51	100
1		13	25	50	100
2		13	26	50	100
3		13	26	51	100
4		13	26	50	100
6		13	25	51	101
10		13	26	50	100
0	ОЗУ, КБ	507.000	509.000	0.000	580.000
1		0.000	398.000	0.000	410.000
2		430.000	0.000	0.000	0.000
3		414.000	0.000	0.000	594.000
4		0.000	407.000	428.000	0.000
6		0.000	444.000	0.000	0.000
10		0.000	0.000	696.000	402.000

Таблица 25: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Лица		SSD-Retina-quantized			
		pt			
		потоки			
		1	2	4	8
0	время, мс	71.282	51.937	47.061	53.441
1		70.997	54.707	44.307	54.458
2		71.692	55.226	42.322	54.089
3		76.499	51.717	47.471	54.617
4		74.618	50.998	45.703	54.074
6		73.783	55.827	49.047	53.969
10		75.695	51.473	48.861	55.650
0	ЦП, %	14	26	51	100
1		13	26	51	100
2		13	26	51	101
3		13	26	51	100
4		13	26	51	100
6		14	26	51	100
10		13	26	51	100
0	ОЗУ, КБ	0.000	0.000	392.000	0.000
1		0.000	340.000	0.000	0.000
2		0.000	334.000	0.000	0.000
3		337.000	0.000	337.000	0.000
4		314.000	0.000	0.000	0.000
6		0.000	328.000	332.000	0.000
10		340.000	0.000	328.000	334.000

Таблица 26: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

Лица		Cascade	SSD				
			caffe	opencv			
		потоки					
		1	1	1	2	4	8
0	время, мс	42.660	40.112	56.475	57.671	41.016	34.349
1		38.562	40.600	56.593	57.511	40.920	34.535
2		43.214	40.483	56.285	57.444	41.228	34.494
3		40.424	40.706	56.505	57.201	40.924	34.755
4		42.684	41.048	56.548	57.249	40.804	34.519
6		44.924	41.824	57.012	57.802	41.223	34.916
10		46.522	41.565	57.448	58.065	41.717	35.395
0	ЦП, %	17	14	13	18	29	46
1		16	14	13	19	28	43
2		17	13	13	18	29	44
3		15	13	13	19	29	44
4		15	13	13	18	28	43
6		14	13	13	19	29	45
10		14	13	13	19	28	47
0	ОЗУ, КБ	716.000	173.000	152.000	0.000	471.000	0.000
1		828.000	86.000	0.000	0.000	0.000	562.000
2		0.000	203.000	0.000	0.000	0.000	567.000
3		638.000	136.000	217.000	0.000	211.000	0.000
4		0.000	118.000	221.000	219.000	301.000	0.000
6		1323.000	111.000	0.000	0.000	486.000	583.000
10		0.000	136.000	193.000	0.000	0.000	508.000
0	ГП, %	57	39	0	0	0	0
1		50	39	0	0	0	0
2		56	39	0	0	0	0
3		52	39	0	0	0	0
4		55	39	0	0	0	0
6		55	38	0	0	0	0
10		50	38	0	0	0	0

0	Память ГП, КБ	2048.000	0.000	0.000	0.000	0.000	0.000
1		2048.000	0.000	0.000	0.000	0.000	0.000
2		2048.000	0.000	0.000	0.000	0.000	0.000
3		2048.000	0.000	0.000	0.000	0.000	0.000
4		2048.000	0.000	0.000	0.000	0.000	0.000
6		2048.000	0.000	0.000	0.000	0.000	0.000
10		2048.000	0.000	0.000	0.000	0.000	0.000

Таблица 27: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	32.731	27.891	26.164	29.001
1		32.731	29.202	26.562	28.736
2		33.080	28.717	26.634	29.225
3		32.921	28.926	26.499	28.582
4		34.018	28.917	26.912	28.771
6		33.075	29.138	27.718	28.812
10		33.245	29.306	29.684	29.178
0	ЦП, %	13	25	50	97
1		14	26	51	95
2		13	26	51	96
3		13	26	50	95
4		12	26	50	95
6		13	26	50	95
10		13	26	51	97
0	ОЗУ, КБ	316.000	0.000	274.000	275.000
1		0.000	27.000	277.000	206.000
2		0.000	216.000	0.000	33.000
3		196.000	237.000	270.000	0.000
4		332.000	0.000	215.000	284.000
6		19.000	210.000	0.000	285.000
10		0.000	254.000	0.000	292.000
0	ГП, %	11	13	14	13
1		12	13	14	13
2		11	13	14	13
3		11	13	13	13
4		11	13	13	13
6		11	13	13	13
10		12	13	13	13

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Таблица 28: FaceDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	24.342	21.864	21.799	25.819
1		22.486	21.593	21.481	23.489
2		22.460	21.854	21.158	22.748
3		22.808	21.806	21.450	23.632
4		23.045	21.906	21.374	22.387
6		24.003	21.747	21.402	22.588
10		25.753	26.479	24.848	26.282
0	ЦП, %	12	25	50	97
1		13	25	50	95
2		13	25	49	96
3		13	25	49	96
4		12	25	50	96
6		12	25	49	96
10		13	25	50	97
0	ОЗУ, КБ	571.000	292.000	679.000	653.000
1		422.000	458.000	828.000	827.000
2		842.000	214.000	751.000	596.000
3		777.000	664.000	595.000	603.000
4		594.000	803.000	19.000	833.000
6		802.000	555.000	409.000	608.000
10		400.000	404.000	596.000	421.000
0	ГП, %	9	10	10	8
1		10	10	9	9
2		9	9	9	9
3		9	10	9	8
4		9	9	10	9
6		9	10	10	9
10		9	9	9	8

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Трекинг

Для трекинговых тестов использовались изображения 1920x1080. Минимальный размер *лица* = 80 пикселей.

Таблица 29: MultiTracker, миним. размер *лица* = 80 пикселей, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Лица		Cascade	SSD				
			caffe	opencv			
		потоки					
		1	1	1	2	4	8
0	время, мс	61.992	35.814	20.953	12.183	9.262	8.974
1		45.517	40.898	21.989	14.860	12.287	11.954
2		98.671	46.446	22.385	16.108	13.926	16.000
3		52.486	42.863	24.336	16.831	15.441	16.705
4		73.384	45.337	28.099	18.084	17.081	21.145
6		113.428	55.961	32.676	25.384	23.510	27.972
10		117.998	37.616	19.897	13.087	10.140	11.535
0	ЦП, %	74	7	5	10	18	32
1		69	12	5	9	16	28
2		75	16	5	9	15	26
3		76	20	5	9	15	26
4		72	23	5	9	14	27
6		72	26	5	8	14	25
10		66	10	5	10	17	32
0	ОЗУ, КБ	0.000	5044.000	2136.000	2940.000	2900.000	2940.000
1		0.000	4816.000	2440.000	3400.000	3164.000	3204.000
2		0.000	5040.000	3400.000	3204.000	3400.000	3400.000
3		264.000	4776.000	2400.000	2412.000	3400.000	3204.000
4		264.000	4816.000	3232.000	3388.000	2676.000	2664.000
6		596.000	5608.000	3488.000	2676.000	3192.000	2704.000
10		596.000	5080.000	3664.000	3624.000	3732.000	3732.000

Таблица 30: MultiTracker, миним. размер лица = 80 пикселей, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	19.291	13.084	10.589	14.062
1		26.350	22.598	20.039	20.056
2		25.604	23.782	24.197	24.073
3		27.705	27.570	24.397	25.117
4		30.640	28.435	28.305	28.643
6		37.692	43.811	35.974	36.074
10		27.080	25.938	23.810	25.418
0	ЦП, %	10	17	28	45
1		17	22	29	43
2		25	29	33	42
3		29	32	38	45
4		32	35	38	43
6		38	34	39	43
10		29	32	38	45
0	ОЗУ, КБ	8340.000	1260.000	192.000	9644.000
1		8600.000	15244.000	11864.000	380.000
2		8600.000	424.000	2080.000	1740.000
3		6460.000	2440.000	4160.000	232.000
4		2564.000	516.000	5264.000	12760.000
6		8628.000	2708.000	328.000	2192.000
10		9156.000	13004.000	948.000	1244.000

Таблица 31: MultiTracker, миним. размер лица = 80 пикселей, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	7.288	6.687	6.743	10.343
1		13.835	13.065	12.802	13.287
2		16.524	16.060	14.015	17.489
3		15.934	16.593	15.162	17.965
4		7.080	6.227	6.082	11.635
6		7.094	6.171	6.303	11.138
10		7.793	6.923	7.035	13.421
0	ЦП, %	8	13	22	44
1		22	25	30	48
2		32	34	40	47
3		33	33	38	46
4		6	10	20	47
6		6	11	20	46
10		6	11	20	42
0	ОЗУ, КБ	6356.000	8120.000	7304.000	6144.000
1		6080.000	7852.000	6016.000	8016.000
2		7676.000	6308.000	5960.000	7800.000
3		7652.000	7852.000	7048.000	6188.000
4		7392.000	6028.000	6780.000	7748.000
6		7392.000	6008.000	6792.000	7752.000
10		5808.000	7588.000	6780.000	6116.000

Таблица 32: MultiTracker, миним. размер лица = 80 пикселей, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Лица		SSD-Retina-quantized			
		pt			
		потоки			
		1	2	4	8
0	время, мс	14.484	10.215	8.435	10.529
1		21.697	18.248	15.486	16.607
2		20.579	21.210	19.408	20.001
3		22.157	23.431	22.136	21.270
4		25.749	27.136	23.834	24.584
6		32.325	35.381	31.763	32.183
10		20.893	19.096	18.697	18.739
0	ЦП, %	12	19	29	47
1		21	26	34	44
2		30	31	36	44
3		35	36	40	47
4		38	38	43	46
6		43	40	43	44
10		30	35	40	47
0	ОЗУ, КБ	14580.000	22992.000	11292.000	11240.000
1		12992.000	25860.000	14232.000	24192.000
2		17992.000	13348.000	11004.000	11236.000
3		14840.000	14112.000	19548.000	11264.000
4		12752.000	13920.000	10196.000	13676.000
6		17812.000	12016.000	11180.000	13116.000
10		18300.000	13872.000	13856.000	14864.000

Таблица 33: MultiTracker, миним. размер лица = 80 пикселей, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

Лица		Cascade	SSD				
			caffe	opencv			
		потоки					
		1	1	1	2	4	8
0	время, мс	11.423	5.017	18.993	13.389	11.319	9.268
1		10.954	8.124	22.948	15.353	13.850	15.139
2		14.309	11.302	21.811	16.932	14.999	14.933
3		13.356	20.009	26.146	18.888	16.826	17.760
4		14.545	25.125	27.474	20.549	18.902	23.395
6		17.755	27.511	30.558	25.737	24.239	25.322
10		19.368	5.887	19.806	14.029	11.575	12.680
0	ЦП, %	25	12	6	10	18	32
1		27	22	6	10	16	30
2		22	40	6	9	16	26
3		23	38	6	9	15	27
4		21	36	6	9	15	25
6		19	47	6	9	14	23
10		18	19	6	10	18	33
0	ОЗУ, КБ	1848.000	268.000	4684.000	4724.000	4684.000	4684.000
1		2376.000	524.000	4948.000	5056.000	4948.000	5016.000
2		2112.000	372.000	4948.000	4948.000	4948.000	4948.000
3		2112.000	264.000	5016.000	4948.000	4948.000	5016.000
4		2640.000	448.000	4988.000	5016.000	4948.000	4932.000
6		2376.000	528.000	5044.000	4948.000	4948.000	5044.000
10		2376.000	792.000	5496.000	5212.000	5212.000	5212.000
0	ГП, %	71	76	0	0	0	0
1		63	54	0	0	0	0
2		68	46	0	0	0	0
3		61	36	0	0	0	0
4		58	31	0	0	0	0
6		57	24	0	0	0	0
10		49	72	0	0	0	0

0	Память ГП, КБ	2048.000	0.000	0.000	0.000	0.000	0.000
1		2048.000	0.000	0.000	0.000	0.000	0.000
2		2048.000	0.000	0.000	0.000	0.000	0.000
3		2048.000	0.000	0.000	0.000	0.000	0.000
4		2048.000	0.000	0.000	0.000	0.000	0.000
6		2048.000	0.000	0.000	0.000	0.000	0.000
10		2048.000	0.000	0.000	0.000	0.000	0.000

Таблица 34: MultiTracker, миним. размер лица = 80 пикселей, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	4.477	3.793	3.558	3.361
1		7.929	6.889	7.097	6.721
2		10.894	9.792	9.245	8.991
3		12.488	11.307	11.061	10.426
4		15.239	14.078	13.464	13.185
6		21.235	21.297	21.012	19.400
10		11.502	10.451	10.049	9.855
0	ЦП, %	15	20	27	39
1		23	28	30	38
2		28	30	35	39
3		36	35	37	42
4		40	36	38	41
6		40	36	36	40
10		36	36	39	43
0	ОЗУ, КБ	12012.000	6008.000	11984.000	12040.000
1		12100.000	9104.000	9196.000	6528.000
2		12296.000	12784.000	9160.000	11888.000
3		9172.000	12004.000	12264.000	772.000
4		12632.000	12340.000	9140.000	6840.000
6		12300.000	12308.000	11736.000	11964.000
10		13328.000	12796.000	9692.000	1324.000
0	ГП, %	32	38	43	43
1		19	22	21	22
2		13	16	16	16
3		12	14	14	15
4		11	11	11	12
6		8	8	8	8
10		17	16	19	19

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Таблица 35: MultiTracker, миним. размер лица = 80 пикселей, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	2.860	2.627	2.545	2.462
1		5.974	6.080	5.643	5.523
2		7.981	7.807	7.734	7.540
3		7.941	7.561	7.599	7.522
4		2.654	2.524	2.392	2.464
6		2.799	2.565	2.435	2.363
10		3.698	3.141	3.177	3.091
0	ЦП, %	11	14	20	30
1		28	28	32	34
2		30	32	34	38
3		30	32	33	38
4		6	9	16	29
6		6	9	16	30
10		6	9	14	24
0	ОЗУ, КБ	528.000	1572.000	792.000	1056.000
1		264.000	504.000	808.000	264.000
2		264.000	528.000	1560.000	264.000
3		1056.000	264.000	284.000	264.000
4		4.000	1324.000	268.000	968.000
6		1040.000	4.000	1316.000	4.000
10		4.000	4.000	4.000	260.000
0	ГП, %	35	39	39	40
1		17	17	17	18
2		13	13	13	13
3		13	13	13	13
4		36	40	40	40
6		35	39	40	41
10		30	31	32	32

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Таблица 36: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Лица		Cascade	SSD				
			caffe	opencv			
		потоки					
		1	1	1	2	4	8
0	время, мс	66.287	41.761	19.205	19.706	13.803	10.630
1		49.501	44.072	22.103	22.503	16.524	13.606
2		94.243	46.129	23.670	23.943	18.042	15.144
3		73.100	47.601	25.047	25.742	19.537	16.587
4		70.982	50.767	27.487	27.867	22.133	19.135
6		97.526	57.520	32.961	33.305	27.783	24.810
10		86.384	42.066	19.964	20.458	14.386	11.433
0	ЦП, %	65	6	6	8	15	29
1		67	6	5	8	13	23
2		61	7	5	8	13	21
3		60	8	5	8	13	22
4		65	9	5	8	12	19
6		60	9	5	7	11	17
10		58	6	5	8	14	28
0	ОЗУ, КБ	1312.000	564.000	236.000	28.000	1300.000	1380.000
1		1748.000	3348.000	772.000	832.000	1552.000	2848.000
2		1188.000	1488.000	1284.000	1348.000	1424.000	1992.000
3		1424.000	2152.000	1368.000	1480.000	1616.000	1808.000
4		2408.000	2252.000	1444.000	1024.000	0.000	1964.000
6		2752.000	264.000	1644.000	1648.000	1760.000	2124.000
10		1796.000	1864.000	1084.000	1816.000	1868.000	2560.000

Таблица 37: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	42.465	31.101	27.115	28.013
1		44.240	34.531	30.119	31.602
2		47.470	36.866	32.843	34.231
3		49.176	37.443	33.596	35.375
4		52.134	41.198	37.578	39.323
6		59.290	48.027	44.093	46.050
10		48.550	36.138	33.033	34.650
0	ЦП, %	6	11	21	41
1		6	13	23	42
2		10	14	23	43
3		10	15	25	44
4		9	17	27	45
6		11	19	27	47
10		9	14	26	45
0	ОЗУ, КБ	1844.000	1824.000	1820.000	1864.000
1		2860.000	2324.000	2308.000	2312.000
2		2308.000	2332.000	2364.000	2324.000
3		2320.000	2424.000	2308.000	2324.000
4		2304.000	2304.000	2348.000	2328.000
6		3376.000	2368.000	2356.000	2392.000
10		3960.000	2872.000	2892.000	2860.000

Таблица 38: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	10.528	8.924	8.844	9.921
1		13.376	11.915	11.797	13.086
2		15.444	13.680	13.557	14.858
3		15.085	13.350	13.288	14.537
4		10.154	8.602	8.399	9.507
6		10.472	8.600	8.477	9.562
10		11.262	9.692	9.582	10.603
0	ЦП, %	5	11	21	41
1		8	17	24	46
2		18	22	27	49
3		17	17	26	50
4		5	11	20	41
6		5	11	20	40
10		6	11	21	41
0	ОЗУ, КБ	2792.000	3236.000	2832.000	2844.000
1		2608.000	2648.000	2600.000	2788.000
2		3132.000	2628.000	2632.000	2628.000
3		2668.000	2820.000	2612.000	2616.000
4		2156.000	2576.000	2672.000	2172.000
6		2624.000	2144.000	2160.000	2136.000
10		2668.000	2092.000	2308.000	2128.000

Таблица 39: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Лица		SSD-Retina-quantized			
		pt			
		потоки			
		1	2	4	8
0	время, мс	22.538	16.070	14.642	15.729
1		25.862	19.506	18.053	19.797
2		28.398	22.441	20.583	22.529
3		29.740	23.211	21.615	23.209
4		33.179	27.703	25.911	27.307
6		39.590	34.172	32.328	34.596
10		28.798	23.008	21.709	22.544
0	ЦП, %	6	11	21	41
1		8	15	22	42
2		10	15	25	46
3		10	20	28	47
4		14	21	28	49
6		12	23	30	50
10		11	17	29	47
0	ОЗУ, КБ	1544.000	1600.000	1644.000	2624.000
1		2400.000	2408.000	2384.000	3596.000
2		1872.000	2416.000	2352.000	4256.000
3		3136.000	1884.000	3000.000	3460.000
4		2396.000	2936.000	2612.000	4212.000
6		2360.000	2364.000	2744.000	4156.000
10		2392.000	2360.000	2392.000	3280.000

Таблица 40: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

Лица		Cascade	SSD				
			caffe	opencv			
		потоки					
		1	1	1	2	4	8
0	время, мс	11.110	5.413	19.226	19.719	13.789	10.742
1		10.711	8.938	22.064	22.425	16.617	13.476
2		14.018	11.315	23.594	23.947	18.114	15.134
3		13.187	13.858	25.056	25.592	19.622	16.539
4		14.590	18.175	27.563	27.802	21.996	19.129
6		17.501	25.486	33.110	33.804	27.879	25.077
10		19.779	6.387	20.074	20.524	14.509	11.465
0	ЦП, %	8	6	5	8	15	27
1		7	14	5	8	13	23
2		7	19	5	8	13	22
3		7	17	5	8	12	21
4		7	16	5	8	12	19
6		7	24	5	7	11	17
10		7	11	5	8	13	28
0	ОЗУ, КБ	1476.000	276.000	552.000	840.000	1632.000	1616.000
1		1800.000	1308.000	1244.000	1488.000	1760.000	1900.000
2		1472.000	628.000	1740.000	1532.000	2196.000	2408.000
3		1856.000	1316.000	1608.000	1536.000	2308.000	2608.000
4		1528.000	596.000	1460.000	1564.000	2656.000	2556.000
6		1192.000	1272.000	2008.000	2240.000	2404.000	2784.000
10		1744.000	904.000	1692.000	1648.000	2520.000	2324.000
0	ГП, %	75	80	0	0	0	1
1		71	44	0	0	0	1
2		66	45	0	0	0	0
3		60	44	0	0	0	0
4		61	40	0	0	0	0
6		52	27	0	0	0	0
10		45	78	0	0	0	1

0	Память ГП, КБ	2048.000	0.000	0.000	0.000	0.000	0.000
1		2048.000	0.000	0.000	0.000	0.000	0.000
2		2048.000	0.000	0.000	0.000	0.000	0.000
3		2048.000	0.000	0.000	0.000	0.000	0.000
4		2048.000	0.000	0.000	0.000	0.000	0.000
6		2048.000	0.000	0.000	0.000	0.000	0.000
10		2048.000	0.000	0.000	0.000	0.000	0.000

Таблица 41: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	6.127	4.768	4.154	3.928
1		9.579	8.191	7.683	7.692
2		12.101	10.776	10.558	10.198
3		13.323	11.885	11.287	11.530
4		16.562	15.395	14.708	15.427
6		23.672	21.772	21.312	22.425
10		12.305	11.027	10.644	10.977
0	ЦП, %	6	11	21	41
1		13	17	28	46
2		15	18	36	50
3		27	23	38	57
4		19	27	35	54
6		24	23	33	53
10		19	22	36	54
0	ОЗУ, КБ	32.000	740.000	724.000	56.000
1		1228.000	1304.000	1304.000	628.000
2		1368.000	2424.000	2716.000	1300.000
3		1776.000	1220.000	1272.000	2044.000
4		1668.000	1300.000	2008.000	544.000
6		1216.000	1296.000	1304.000	568.000
10		1840.000	1916.000	1824.000	3432.000
0	ГП, %	27	35	38	40
1		19	21	22	22
2		15	16	16	16
3		12	15	16	16
4		10	12	12	12
6		9	9	9	9
10		19	20	20	19

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Таблица 42: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	3.100	2.813	2.758	2.803
1		6.318	6.039	5.948	6.173
2		7.949	7.651	7.599	7.886
3		7.752	7.671	7.561	7.591
4		2.940	2.626	2.559	2.567
6		2.959	2.628	2.592	2.586
10		4.054	3.727	3.724	3.776
0	ЦП, %	8	12	20	41
1		16	23	30	51
2		19	24	33	55
3		20	26	41	53
4		5	10	20	40
6		5	10	20	40
10		5	10	20	40
0	ОЗУ, КБ	3004.000	3724.000	3732.000	3704.000
1		3592.000	2044.000	3596.000	3704.000
2		3568.000	2052.000	2032.000	3540.000
3		4488.000	3848.000	3668.000	2060.000
4		2340.000	3200.000	3096.000	3032.000
6		3284.000	3032.000	2372.000	1540.000
10		2344.000	3144.000	3064.000	1600.000
0	ГП, %	33	36	37	37
1		16	18	18	17
2		13	14	14	14
3		14	14	14	14
4		34	38	40	40
6		34	38	39	39
10		26	27	28	27

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Таблица 43: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Лица			Cascade	SSD				
				caffe	opencv			
			потоки					
			1	1	1	2	4	8
0		время, мс	57.032	25.422	11.220	11.397	7.239	5.628
1			48.483	26.616	13.014	13.203	9.063	7.598
2			77.117	27.751	14.093	14.141	10.074	8.684
3			58.709	28.412	15.010	15.200	11.124	9.767
4			66.741	30.311	16.633	16.741	12.720	11.465
6			77.557	34.361	20.300	20.350	16.419	15.319
10			77.610	25.465	11.799	11.865	7.761	6.126
0		ЦП, %	89	13	13	20	37	65
1			82	14	13	19	33	55
2			92	14	13	19	31	50
3			88	15	13	19	27	49
4			87	15	13	18	27	48
6			89	15	13	18	25	44
10			77	13	13	20	35	63
0		ОЗУ, КБ	0.000	0.000	0.000	0.000	0.000	0.000
1			0.000	0.000	0.000	0.000	0.000	0.000
2			0.000	0.000	0.000	0.000	0.000	0.000
3			0.000	0.000	0.000	0.000	0.000	0.000
4			0.000	3132.000	0.000	0.000	0.000	0.000
6			0.000	0.000	0.000	0.000	0.000	0.000
10			0.000	1056.000	0.000	0.000	0.000	0.000

Таблица 44: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	25.594	16.954	14.153	17.069
1		27.458	19.873	17.043	19.039
2		29.669	21.595	18.550	20.085
3		29.740	21.527	16.728	20.431
4		32.131	24.024	18.934	23.249
6		34.873	27.034	24.847	27.619
10		29.912	21.803	17.626	20.869
0	ЦП, %	13	26	51	100
1		14	26	51	99
2		14	27	52	99
3		14	28	54	99
4		14	28	53	100
6		15	29	55	98
10		15	27	53	100
0	ОЗУ, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	2260.000	0.000
2		2280.000	2272.000	0.000	2412.000
3		0.000	0.000	0.000	0.000
4		2320.000	2304.000	0.000	0.000
6		0.000	0.000	2596.000	2444.000
10		2840.000	2800.000	0.000	3624.000

Таблица 45: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	6.211	5.193	4.807	5.231
1		7.883	6.799	6.471	6.993
2		8.886	7.896	7.368	8.079
3		8.713	7.751	7.272	7.879
4		5.979	4.992	4.587	6.233
6		6.052	5.115	4.637	5.153
10		6.802	5.754	5.495	5.930
0	ЦП, %	13	25	51	100
1		15	27	52	100
2		16	29	54	100
3		17	30	54	100
4		13	25	51	96
6		13	26	51	100
10		13	25	50	100
0	ОЗУ, КБ	0.000	0.000	0.000	0.000
1		2020.000	0.000	2212.000	0.000
2		3668.000	2852.000	3016.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	1560.000	1776.000	1788.000
6		0.000	0.000	1616.000	0.000
10		0.000	0.000	1624.000	0.000

Таблица 46: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Лица		SSD-Retina-quantized			
		pt			
		потоки			
		1	2	4	8
0	время, мс	15.545	10.602	8.342	9.936
1		17.578	12.194	10.995	11.686
2		18.994	13.581	12.240	13.653
3		19.806	14.355	12.906	14.073
4		22.277	16.137	14.095	16.066
6		25.659	20.651	18.247	19.971
10		19.800	14.122	12.201	13.502
0	ЦП, %	13	26	51	100
1		14	27	52	99
2		15	28	53	100
3		14	29	53	99
4		16	31	56	99
6		16	29	54	99
10		16	30	53	99
0	ОЗУ, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		1832.000	0.000	1828.000	0.000
3		1816.000	0.000	2636.000	1972.000
4		2576.000	0.000	0.000	2872.000
6		2588.000	2360.000	0.000	0.000
10		1796.000	0.000	0.000	2904.000

Таблица 47: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

Лица		Cascade	SSD				
			caffe	opencv			
		потоки					
		1	1	1	2	4	8
0	время, мс	11.421	6.901	11.228	11.407	7.244	5.542
1		11.209	8.964	13.087	13.319	9.108	7.608
2		13.546	10.863	14.032	14.256	10.075	8.700
3		13.098	11.201	15.026	15.214	11.130	9.718
4		13.992	12.883	16.718	16.850	12.728	11.371
6		16.037	16.680	20.318	20.513	16.479	15.285
10		17.470	7.542	11.765	11.966	7.776	6.106
0	ЦП, %	15	14	13	20	36	69
1		15	15	13	19	31	54
2		15	12	13	19	29	56
3		15	16	13	19	30	54
4		15	19	13	18	27	43
6		14	18	13	17	25	38
10		14	14	13	20	32	60
0	ОЗУ, КБ	0.000	92.000	608.000	0.000	0.000	0.000
1		5400.000	836.000	0.000	0.000	0.000	0.000
2		0.000	520.000	0.000	0.000	0.000	0.000
3		0.000	1456.000	1232.000	0.000	0.000	2996.000
4		3388.000	532.000	0.000	1796.000	0.000	0.000
6		1580.000	584.000	1744.000	1796.000	0.000	2896.000
10		1692.000	1744.000	1964.000	1688.000	0.000	0.000
0	ГП, %	73	52	3	3	2	4
1		66	43	2	2	3	2
2		67	38	2	2	2	3
3		62	37	2	2	1	3
4		59	31	1	1	0	3
6		59	23	1	1	1	1
10		51	49	0	2	3	6

0	Память ГП, КБ	2048.000	0.000	0.000	0.000	0.000	0.000
1		2048.000	0.000	0.000	0.000	0.000	0.000
2		2048.000	0.000	0.000	0.000	0.000	0.000
3		2048.000	0.000	0.000	0.000	0.000	0.000
4		2048.000	0.000	0.000	0.000	0.000	0.000
6		2048.000	0.000	0.000	0.000	0.000	0.000
10		2048.000	0.000	0.000	0.000	0.000	0.000

Таблица 48: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

Лица		SSD-Retina			
		pt			
		потоки			
		1	2	4	8
0	время, мс	5.174	4.080	3.672	3.885
1		7.146	5.943	5.637	5.875
2		8.583	7.522	6.966	7.593
3		9.142	8.151	7.713	8.157
4		11.081	10.014	9.420	9.938
6		15.766	14.137	13.529	14.253
10		9.116	8.064	7.323	7.442
0	ЦП, %	15	26	52	93
1		16	28	53	97
2		16	30	53	97
3		18	30	54	98
4		17	33	55	98
6		20	29	55	98
10		18	30	54	98
0	ОЗУ, КБ	0.000	760.000	848.000	988.000
1		1528.000	1280.000	1316.000	1288.000
2		1296.000	568.000	1256.000	224.000
3		1292.000	500.000	628.000	1560.000
4		1284.000	1328.000	1236.000	268.000
6		1584.000	1356.000	1312.000	0.000
10		1920.000	1864.000	388.000	2008.000
0	ГП, %	19	24	27	24
1		14	17	17	17
2		11	13	14	13
3		11	13	13	12
4		10	11	11	10
6		7	8	8	8
10		13	13	15	15

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Таблица 49: MultiTracker, миним. размер лица = 80 пикселей, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

Лица		SSD-Retina-ShuffleNet			
		pt			
		потоки			
		1	2	4	8
0	время, мс	2.859	2.561	2.461	2.490
1		4.637	4.391	4.474	4.504
2		5.523	5.804	5.192	5.490
3		5.322	5.149	5.077	5.772
4		2.674	2.379	2.296	2.413
6		2.705	2.396	2.323	2.809
10		3.512	3.316	3.227	3.439
0	ЦП, %	13	26	49	97
1		14	27	53	98
2		18	30	54	98
3		17	29	55	98
4		13	25	50	97
6		13	25	48	97
10		13	25	48	97
0	ОЗУ, КБ	3772.000	3924.000	2204.000	3940.000
1		3524.000	2856.000	2104.000	1928.000
2		2336.000	1244.000	1968.000	3544.000
3		3032.000	2152.000	1280.000	2060.000
4		2332.000	3028.000	2264.000	1684.000
6		764.000	2440.000	3004.000	2324.000
10		728.000	2228.000	1560.000	3260.000
0	ГП, %	19	20	21	21
1		12	12	12	12
2		9	10	12	12
3		10	12	11	12
4		20	22	22	22
6		20	22	22	22
10		16	16	17	15

0	Память ГП, КБ	0.000	0.000	0.000	0.000
1		0.000	0.000	0.000	0.000
2		0.000	0.000	0.000	0.000
3		0.000	0.000	0.000	0.000
4		0.000	0.000	0.000	0.000
6		0.000	0.000	0.000	0.000
10		0.000	0.000	0.000	0.000

Обнаружение ключевых точек

Таблица 50: LandmarkDetector, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Правильное вращение		DLIB	MTCNN				
			caffe	opencv			
		потоки					
		4	1	1	2	4	8
False	время, мс	0.379	4.418	2.055	1.754	1.595	1.769
True		8.703	11.570	14.232	11.497	12.062	13.135
False	ЦП, %	6	6	5	10	19	38
True		29	18	6	8	12	18
False	ОЗУ, КБ	0.000	1.176	0.000	0.000	0.824	0.000
True		1.176	0.824	0.824	0.824	0.824	0.824

Таблица 51: LandmarkDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Правильное вращение		DLIB	MTCNN				
			caffe	opencv			
		потоки					
		4	1	1	2	4	8
False	время, мс	0.410	4.146	2.232	2.411	2.263	2.124
True		10.472	12.901	17.632	14.402	12.710	11.250
False	ЦП, %	5	5	5	8	12	17
True		12	13	5	8	10	14
False	ОЗУ, КБ	1.294	5.529	0.471	1.176	15.765	9.529
True		363.294	463.059	9.882	74.235	129.765	325.294

Таблица 52: LandmarkDetector, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Правильное вращение		DLIB	MTCNN				
			caffe	opencv			
		потоки					
		4	1	1	2	4	8
False	время, мс	0.256	2.537	1.253	1.311	1.134	1.060
True		6.555	7.888	11.488	8.806	7.597	6.725

False	ЦП, %	13	13	13	17	25	42
True		33	40	13	19	23	31
False	ОЗУ, КБ	2.706	3.176	0.353	0.824	0.353	18.353
True		170.941	237.059	0.824	65.176	170.235	216.235

Создание модели

Таблица 53: FIREngine, SlenderFAN-D3, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

	caffe	opencv			
	потоки				
	1	1	2	4	8
время, мс	95.630	89.586	66.465	49.788	48.969
ЦП, %	6	5	10	19	33
ОЗУ, КБ	1096.000	0.000	0.000	0.000	5932.000

Таблица 54: FIREngine, SlenderFAN-D3, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

	caffe	opencv			
	потоки				
	1	1	2	4	8
время, мс	10.153	82.151	72.060	55.154	49.454
ЦП, %	6	6	10	19	33
ОЗУ, КБ	0.000	28.000	40.000	40.000	0.000
ГП, %	47	0	0	0	0
Память ГП, КБ	0.000	0.000	0.000	0.000	0.000

Таблица 55: FIREngine, SlenderFAN-D3, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

	caffe	opencv			
	потоки				
	1	1	2	4	8
время, мс	103.561	85.794	90.621	62.658	55.305
ЦП, %	5	5	8	14	20
ОЗУ, КБ	1192.000	600.000	740.000	840.000	1624.000

Таблица 56: FIREngine, SlenderFAN-D3, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

	caffe	opencv			
		потоки			
	1	1	2	4	8
время, мс	16.861	88.405	90.886	63.881	55.921
ЦП, %	5	5	8	12	20
ОЗУ, КБ	24.000	608.000	552.000	640.000	1244.000
ГП, %	38	0	0	0	0
Память ГП, КБ	0.000	0.000	0.000	0.000	0.000

Таблица 57: FIREngine, SlenderFAN-D3, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

	caffe	opencv			
		потоки			
	1	1	2	4	8
время, мс	67.716	49.796	51.621	34.323	30.094
ЦП, %	13	13	19	33	52
ОЗУ, КБ	0.000	0.000	0.000	0.000	0.000

Таблица 58: FIREngine, SlenderFAN-D3, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

	caffe	opencv			
		потоки			
	1	1	2	4	8
время, мс	12.365	49.957	52.022	34.482	30.195
ЦП, %	13	13	19	33	53
ОЗУ, КБ	32.000	0.000	0.000	0.000	0.000
ГП, %	33	0	0	0	0
Память ГП, КБ	0.000	0.000	0.000	0.000	0.000

Таблица 59: FIREngine, tpaminet3-ao256a, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

	caffe	opencv			
		потоки			
	1	1	2	4	8
время, мс	340.182	501.566	258.674	188.275	169.235
ЦП, %	6	6	10	20	37
ОЗУ, КБ	1320.000	524.000	524.000	588.000	552.000

Таблица 60: FIREngine, tpaminet3-ao256a, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

	caffe	opencv			
		потоки			
	1	1	2	4	8
время, мс	15.187	511.590	295.962	213.703	149.592
ЦП, %	6	6	10	20	37
ОЗУ, КБ	0.000	524.000	0.000	524.000	524.000
ГП, %	69	0	0	0	0
Память ГП, КБ	0.000	0.000	0.000	0.000	0.000

Таблица 61: FIREngine, tpaminet3-ao256a, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

	caffe	opencv			
		потоки			
	1	1	2	4	8
время, мс	362.034	303.516	327.365	213.661	159.572
ЦП, %	5	5	8	13	23
ОЗУ, КБ	1756.000	8.000	0.000	596.000	544.000

Таблица 62: FIREngine, tpaminet3-ao256a, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

	caffe	opencv			
		потоки			
	1	1	2	4	8
время, мс	19.277	303.755	328.756	212.316	158.981
ЦП, %	5	5	8	13	24
ОЗУ, КБ	32.000	608.000	612.000	52.000	68.000
ГП, %	64	0	0	0	0
Память ГП, КБ	0.000	0.000	0.000	0.000	0.000

Таблица 63: FIREngine, tpaminet3-ao256a, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

	caffe	opencv			
		потоки			
	1	1	2	4	8
время, мс	227.057	183.789	185.265	111.234	80.896
ЦП, %	13	13	20	33	62
ОЗУ, КБ	1164.000	8.000	584.000	12.000	4.000

Таблица 64: FIREngine, tpaminet3-ao256a, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

	caffe	opencv			
		потоки			
	1	1	2	4	8
время, мс	13.984	184.595	187.167	113.270	81.164
ЦП, %	13	13	20	33	59
ОЗУ, КБ	20.000	16.000	608.000	0.000	52.000
ГП, %	54	0	0	0	0
Память ГП, КБ	0.000	0.000	0.000	0.000	0.000

Таблица 65: FIREngine, IR_SE_38_256, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	163.264	108.128	86.708	88.980
ЦП, %	6	10	20	40
ОЗУ, КБ	3192.000	28.000	28.000	2808.000

Таблица 66: FIREngine, IR_SE_38_256, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	13.194	11.511	11.634	11.682
ЦП, %	6	9	14	25
ОЗУ, КБ	0.000	0.000	0.000	0.000
ГП, %	60	60	60	63
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 67: FIREngine, IR_SE_38_256, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	215.847	153.845	134.393	133.289
ЦП, %	5	10	20	41
ОЗУ, КБ	3620.000	3192.000	3256.000	3220.000

Таблица 68: FIREngine, IR_SE_38_256, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	17.520	17.483	17.323	17.865
ЦП, %	5	10	20	40
ОЗУ, КБ	40.000	328.000	52.000	60.000
ГП, %	55	54	51	54
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 69: FIREngine, IR_SE_38_256, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	143.159	95.459	75.087	91.222
ЦП, %	13	25	50	100
ОЗУ, КБ	68.000	3488.000	3224.000	3232.000

Таблица 70: FIREngine, IR_SE_38_256, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

	pt			
	потоки			
	1	2	4	8
время, мс	12.050	11.872	12.167	12.173
ЦП, %	13	25	49	95
ОЗУ, КБ	180.000	36.000	176.000	44.000
ГП, %	40	40	40	39
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 71: FIREngine, FastNet22F, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	170.512	102.761	75.860	68.092
ЦП, %	6	10	20	40
ОЗУ, КБ	9660.000	28.000	272.000	260.000

Таблица 72: FIREngine, FastNet22F, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	10.832	9.881	10.174	9.898
ЦП, %	6	9	15	29
ОЗУ, КБ	24.000	0.000	0.000	0.000
ГП, %	50	55	51	54
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 73: FIREngine, FastNet22F, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	218.106	148.924	117.916	108.894
ЦП, %	5	11	20	41
ОЗУ, КБ	2176.000	1640.000	2236.000	1060.000

Таблица 74: FIREngine, FastNet22F, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	16.340	16.117	16.563	16.733
ЦП, %	5	10	20	40
ОЗУ, КБ	32.000	32.000	32.000	60.000
ГП, %	41	41	40	41
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 75: FIREngine, FastNet22F, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	179.021	107.519	76.516	71.921
ЦП, %	13	26	51	100
ОЗУ, КБ	24.000	152.000	2356.000	2968.000

Таблица 76: FIREngine, FastNet22F, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

	pt			
	потоки			
	1	2	4	8
время, мс	9.747	9.733	9.605	10.401
ЦП, %	13	25	50	97
ОЗУ, КБ	24.000	0.000	24.000	0.000
ГП, %	28	28	28	26
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 77: FIREngine, VFN_B, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	52.842	45.992	47.300	58.275
ЦП, %	6	10	20	40
ОЗУ, КБ	3960.000	2132.000	1340.000	2072.000

Таблица 78: FIREngine, VFN_B, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	36.457	36.862	37.912	38.242
ЦП, %	6	7	11	17
ОЗУ, КБ	0.000	0.000	0.000	24.000
ГП, %	62	66	65	65
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 79: FIREngine, VFN_B, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	67.705	59.297	60.128	65.590
ЦП, %	5	10	21	41
ОЗУ, КБ	3328.000	3328.000	2872.000	2912.000

Таблица 80: FIREngine, VFN_B, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	43.735	43.943	45.623	47.063
ЦП, %	5	10	20	40
ОЗУ, КБ	44.000	48.000	32.000	36.000
ГП, %	54	61	59	58
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 81: FIREngine, VFN_B, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	40.680	34.402	31.738	35.484
ЦП, %	13	26	50	99
ОЗУ, КБ	0.000	0.000	0.000	2036.000

Таблица 82: FIREngine, VFN_B, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

	pt			
	потоки			
	1	2	4	8
время, мс	35.592	36.247	41.209	37.933
ЦП, %	13	25	48	97
ОЗУ, КБ	48.000	32.000	44.000	32.000
ГП, %	42	41	41	41
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 83: FIREngine, VFN_BM, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	52.737	45.723	47.471	57.920
ЦП, %	5	10	20	40
ОЗУ, КБ	1876.000	1848.000	3708.000	1068.000

Таблица 84: FIREngine, VFN_BM, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	36.180	36.796	37.544	37.976
ЦП, %	6	7	11	17
ОЗУ, КБ	0.000	24.000	0.000	24.000
ГП, %	66	65	66	63
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 85: FIREngine, VFN_BM, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	67.035	59.446	59.988	65.446
ЦП, %	6	10	21	41
ОЗУ, КБ	3020.000	3292.000	2968.000	3444.000

Таблица 86: FIREngine, VFN_BM, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	44.133	44.363	45.639	46.890
ЦП, %	5	10	20	40
ОЗУ, КБ	40.000	32.000	44.000	40.000
ГП, %	60	58	54	57
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 87: FIREngine, VFN_BM, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	40.481	34.366	32.266	37.470
ЦП, %	13	26	50	100
ОЗУ, КБ	1992.000	2264.000	2088.000	2148.000

Таблица 88: FIREngine, VFN_BM, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

	pt			
	потоки			
	1	2	4	8
время, мс	34.939	35.134	35.566	40.057
ЦП, %	13	25	49	97
ОЗУ, КБ	44.000	36.000	24.000	36.000
ГП, %	42	42	42	41
Память ГП, КБ	0.000	0.000	0.000	0.000

Верификация

Таблица 89: Verification, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц (статистика на базе 10000 сравнений)

	Длина FIR	
	256	512
время, мс	624.612	629.019
ЦП, %	6	5
ОЗУ, КБ	20.560	0.000

Таблица 90: Verification, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц (статистика на базе 10000 сравнений)

	Длина FIR	
	256	512
время, мс	743.410	758.438
ЦП, %	6	5
ОЗУ, КБ	0.000	0.000

Таблица 91: Верификация, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц (статистика на базе 10000 сравнений)

	Длина FIR	
	256	512
время, мс	207.184	194.378
ЦП, %	13	13
ОЗУ, КБ	0.000	0.000

Идентификация

Галерея

Таблица 92: IdentificationGallery, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Размер галереи		Длина FIR							
		256				512			
		FIR для идентификации							
		1	10	100	200	1	10	100	200
1000	время, мс	0.262	1.239	10.165	19.568	0.312	1.442	12.928	24.294
10000		1.381	9.971	93.830	180.767	2.023	12.090	115.387	219.433
100000		19.012	113.412	1141.821	2293.743	24.798	147.435	1500.208	3029.062
1000000		292.373	1779.733	16135.940	32021.969	405.135	2289.310	20111.510	42070.000
1000	ЦП, %	5	5	5	5	6	5	5	5
10000		5	5	5	5	5	5	5	5
100000		5	5	5	6	5	5	5	6
1000000		6	5	5	5	6	5	5	5
1000	ОЗУ, КБ	2516.000	724.000	304.000	260.000	2512.000	676.000	560.000	260.000
10000		608.000	512.000	2428.000	2372.000	276.000	340.000	2592.000	2636.000
100000		260.000	2636.000	0.000	556.000	0.000	1580.000	0.000	4.000
1000000		260.000	24544.000	0.000	0.000	0.000	21380.000	0.000	4.000

Таблица 93: IdentificationGallery, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

Размер галереи		Длина FIR							
		256				512			
		FIR для идентификации							
		1	10	100	200	1	10	100	200
1000	время, мс	2.761	4.120	15.613	26.148	2.861	4.145	15.090	26.226
10000		4.086	11.967	83.172	156.393	3.877	12.384	83.924	165.675
100000		15.912	104.225	1011.710	2032.697	22.871	147.144	1446.793	2930.164
1000000		308.738	1680.663	14349.718	29157.294	470.390	2445.603	21402.848	42056.463
1000	ЦП, %	6	6	6	6	6	6	6	6
10000		6	6	6	6	6	6	6	6
100000		6	6	6	6	6	6	6	6
1000000		6	6	6	6	6	6	6	6
1000	ОЗУ, КБ	3604.000	792.000	568.000	260.000	0.000	944.000	572.000	260.000
10000		28.000	0.000	260.000	2108.000	292.000	0.000	0.000	0.000
100000		0.000	0.000	21376.000	0.000	0.000	264.000	17420.000	0.000
1000000		0.000	19532.000	232760.000	15136.000	0.000	8440.000	0.000	156.000
1000	ГП, %	5	6	4	3	6	9	6	5
10000		5	3	1	1	6	4	2	2
100000		3	2	1	0	4	1	0	1
1000000		1	1	2	4	1	1	2	4
1000	Память ГП, КБ	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10000		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
100000		0.000	0.000	2048.000	4096.000	0.000	0.000	2048.000	4096.000
1000000		0.000	2048.000	20480.000	40960.000	0.000	2048.000	20480.000	40960.000

Таблица 94: IdentificationGallery, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Размер галереи		Длина FIR							
		256				512			
		FIR для идентификации							
		1	10	100	200	1	10	100	200
1000	время, мс	0.368	1.259	10.172	19.844	0.395	1.379	10.655	20.806
10000		1.303	9.488	79.937	156.437	1.546	10.636	83.438	162.435
100000		11.955	83.385	986.385	1944.530	15.670	97.485	983.909	1972.529
1000000		111.958	1025.162	9964.583	20039.690	149.390	1167.421	10328.106	20576.727
1000	ЦП, %	5	6	5	5	5	5	5	5
10000		5	5	5	5	5	5	6	6
100000		5	6	6	5	5	6	5	6
1000000		6	5	6	5	6	6	5	6
1000	ОЗУ, КБ	3348.000	924.000	684.000	1060.000	3308.000	1044.000	972.000	1340.000
10000		936.000	152.000	3400.000	4416.000	760.000	156.000	3772.000	4004.000
100000		112.000	2668.000	6564.000	9340.000	136.000	2604.000	7008.000	9188.000
1000000		656.000	8668.000	96708.000	0.000	116.000	8236.000	99804.000	105472.000

Таблица 95: IdentificationGallery, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

Размер галереи		Длина FIR							
		256				512			
		FIR для идентификации							
		1	10	100	200	1	10	100	200
1000	время, мс	1.401	2.459	10.833	20.797	1.418	2.598	11.210	20.730
10000		2.197	9.683	79.676	154.292	2.191	9.694	78.746	153.489
100000		10.260	80.445	908.597	1826.291	10.457	81.358	908.565	1828.857
1000000		84.460	958.498	9439.663	18871.900	88.998	983.752	9424.430	18763.835
1000	ЦП, %	9	8	6	6	9	8	6	6
10000		8	6	6	6	8	6	6	6
100000		6	6	6	6	6	6	5	6
1000000		6	5	6	6	5	6	6	6
1000	ОЗУ, КБ	4700.000	1248.000	212.000	368.000	4656.000	1260.000	200.000	356.000
10000		840.000	284.000	2816.000	3720.000	924.000	164.000	2680.000	3552.000
100000		200.000	2656.000	5968.000	9108.000	224.000	2636.000	6108.000	8908.000
1000000		1700.000	8404.000	94004.000	76616.000	1552.000	8940.000	95936.000	103732.000
1000	ГП, %	10	10	8	5	11	14	10	6
10000		8	6	1	1	10	7	2	1
100000		5	2	1	1	9	5	2	1
1000000		5	3	1	3	9	4	2	4
1000	Память ГП, КБ	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10000		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
100000		0.000	0.000	2048.000	4096.000	0.000	0.000	2048.000	4096.000
1000000		0.000	2048.000	20480.000	40960.000	0.000	2048.000	20480.000	40960.000

Таблица 96: IdentificationGallery, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Размер галереи		Длина FIR							
		256				512			
		FIR для идентификации							
		1	10	100	200	1	10	100	200
1000	время, мс	0.147	0.683	5.788	11.515	0.159	0.749	6.139	12.281
10000		0.712	5.787	53.729	107.537	0.890	6.661	57.715	114.853
100000		7.568	56.759	678.591	1362.398	10.409	66.901	717.123	1426.102
1000000		76.734	731.914	6949.874	13899.055	106.858	822.618	7327.473	14578.709
1000	ЦП, %	13	13	13	13	13	13	13	13
10000		13	13	13	13	13	13	13	13
100000		13	13	13	13	13	13	13	13
1000000		13	13	13	13	13	13	13	13
1000	ОЗУ, КБ	3292.000	1024.000	1352.000	1656.000	3200.000	1000.000	1604.000	1972.000
10000		872.000	632.000	7252.000	6504.000	804.000	432.000	7004.000	6732.000
100000		284.000	7712.000	16824.000	0.000	496.000	7780.000	25296.000	0.000
1000000		1048.000	116.000	312.000	56.000	112.000	4.000	404.000	300.000

Таблица 97: IdentificationGallery, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

Размер галереи		Длина FIR							
		256				512			
		FIR для идентификации							
		1	10	100	200	1	10	100	200
1000	время, мс	2.907	4.509	12.949	21.860	3.011	4.642	14.004	25.864
10000		3.989	9.816	62.303	128.438	3.904	9.645	60.059	118.849
100000		9.028	59.540	675.452	1359.828	9.389	59.150	668.664	1353.448
1000000		61.242	685.927	6725.458	13675.606	63.837	694.873	6742.061	13421.755
1000	ЦП, %	15	16	16	17	13	13	11	14
10000		14	13	13	13	15	15	14	14
100000		14	13	13	13	14	13	13	13
1000000		13	13	13	13	13	13	13	13
1000	ОЗУ, КБ	4436.000	1236.000	584.000	1632.000	4516.000	1168.000	500.000	1264.000
10000		944.000	752.000	7088.000	9908.000	908.000	972.000	6960.000	6868.000
100000		932.000	7212.000	29688.000	0.000	600.000	7248.000	30644.000	0.000
1000000		1924.000	4976.000	43936.000	0.000	2356.000	4976.000	43948.000	49968.000
1000	ГП, %	1	2	3	4	1	2	4	5
10000		1	2	2	3	2	3	3	4
100000		6	3	2	3	7	4	3	3
1000000		6	3	8	16	9	6	9	18
1000	Память ГП, КБ	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
10000		0.000	0.000	0.000	2048.000	0.000	0.000	0.000	0.000
100000		0.000	0.000	4928.000	9792.000	0.000	0.000	4928.000	9792.000
1000000		0.000	4928.000	48832.000	97664.000	0.000	4928.000	48832.000	97664.000

Галерея индексов

Обратите внимание, что для галереи индексов сравнение FIR по разным номерам не проводится., потому что в случае галереи индексов FIR идентифицируются один за другим, поэтому зависимость времени от количества FIR является линейной.

Таблица 98: IdentificationIndexGallery, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Размер галереи		Длина FIR	
		256	512
1000	время, мс	0.132	0.210
10000		0.432	0.715
100000		0.992	0.959
1000000		1.076	1.307
1000	ЦП, %	6	5
10000		5	5
100000		5	5
1000000		5	5
1000	ОЗУ, КБ	0.000	0.000
10000		0.000	0.000
100000		0.000	0.000
1000000		0.000	0.000

Таблица 99: IdentificationIndexGallery, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Размер галереи		Длина FIR	
		256	512
1000	время, мс	0.121	0.163
10000		0.485	0.715
100000		1.143	1.127
1000000		1.454	1.586
1000	ЦП, %	6	5
10000		5	5
100000		5	5
1000000		5	5
1000	ОЗУ, КБ	84.000	84.000
10000		64.000	88.000
100000		68.000	68.000
1000000		176.000	72.000

Таблица 100: IdentificationIndexGallery, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Размер галереи		Длина FIR	
		256	512
1000	время, мс	0.068	0.100
10000		0.344	0.497
100000		0.778	0.722
1000000		1.086	1.091
1000	ЦП, %	13	13
10000		13	13
100000		13	13
1000000		13	13
1000	ОЗУ, КБ	104.000	72.000
10000		68.000	68.000
100000		72.000	136.000
1000000		108.000	108.000

Портретные характеристики

Ниже приведены таблицы со статистикой производительности методов PortraitEngine. Все эти методы, кроме compute_characteristics, используют ленивые вычисления, поэтому ресурсы потребляются только при первом вызове. "face_param" – это псевдоним для методов: no_dark_glasses, no_glasses, no_beard, no_mustache, mouth_closed, male, female, no_smile (ресурсы потребляются только при первом вызове одного из этих методов). "eye_open" – это псевдоним для методов: left_eye_open, right_eye_open (ресурсы потребляются только при первом вызове одного из этих методов).

Таблица 101: PortraitEngine, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

Метод		caffe	opencv			
		потоки				
		1	1	2	4	8
angles	время, мс	85.786	82.589	58.492	42.580	32.105
compute_characteristics		0.878	0.901	0.882	0.864	0.891
deviation_from_uniform_lighting		3.254	2.926	2.980	2.924	2.942
exposure		4.213	3.992	4.374	3.815	3.996
eye_open		22.107	25.231	15.323	14.079	14.763
face_param		92.162	76.992	59.486	45.701	40.651
frontal_pose_deviation		0.099	0.101	0.099	0.100	0.099
grayscale_density		9.206	8.936	8.917	8.939	10.179
hotspots		11.165	10.957	11.037	11.041	11.099
sharpness		1.952	1.720	1.725	1.726	1.733
angles	ЦП, %	6	6	10	20	38
compute_characteristics		6	6	6	6	6
deviation_from_uniform_lighting		5	6	5	5	5
exposure		5	5	6	5	6
eye_open		6	6	10	17	31
face_param		6	6	10	19	34
frontal_pose_deviation		5	6	5	5	6
grayscale_density		6	6	6	6	6
hotspots		5	5	6	6	6
sharpness		5	5	5	6	5

angles	ОЗУ, КБ	58.356	52.804	46.448	47.312	49.724
compute_characteristics		0.004	0.004	0.004	0.004	0.004
deviation_from_uniform_lighting		4.756	4.596	4.624	4.600	4.632
exposure		1850.656	1858.404	1857.848	1857.792	1858.496
eye_open		2.848	14.084	7.652	7.844	10.528
face_param		56.428	126.672	120.392	119.056	122.580
frontal_pose_deviation		0.244	0.252	0.232	0.100	0.308
grayscale_density		1857.660	1858.120	1858.564	1858.580	1857.596
hotspots		1965.160	1860.180	1860.508	1860.548	1860.168
sharpness		391.328	393.380	393.276	393.268	393.296

Таблица 102: PortraitEngine, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

Метод		caffe	opencv			
		потоки				
		1	1	2	4	8
angles	время, мс	4.557	87.850	66.160	46.012	32.557
compute_characteristics	время, мс	1.268	1.294	1.237	1.236	1.067
deviation_from_uniform_lighting	время, мс	3.069	3.060	3.078	3.114	3.088
exposure	время, мс	4.275	4.152	4.384	4.137	4.114
eye_open	время, мс	5.079	25.763	17.268	14.137	13.974
face_param	время, мс	6.971	77.841	64.954	49.593	35.379
frontal_pose_deviation	время, мс	0.191	0.191	0.187	0.186	0.187
grayscale_density	время, мс	9.143	9.650	9.134	9.108	9.122
hotspots	время, мс	11.226	11.156	11.222	11.155	11.259
sharpness	время, мс	1.833	1.855	1.834	1.829	1.830
angles	ЦП, %	6	6	10	20	38
compute_characteristics	ЦП, %	5	5	5	5	5
deviation_from_uniform_lighting	ЦП, %	6	6	6	6	6
exposure	ЦП, %	6	6	6	6	6
eye_open	ЦП, %	6	6	10	17	30
face_param	ЦП, %	6	6	10	19	34
frontal_pose_deviation	ЦП, %	6	5	6	5	5
grayscale_density	ЦП, %	6	6	6	6	6
hotspots	ЦП, %	6	6	6	6	6
sharpness	ЦП, %	6	6	6	6	6
angles	ОЗУ, КБ	2.440	53.244	40.868	40.748	44.872
compute_characteristics	ОЗУ, КБ	0.004	0.004	0.004	0.004	0.004
deviation_from_uniform_lighting	ОЗУ, КБ	4.724	4.572	4.760	4.684	4.600
exposure	ОЗУ, КБ	1857.700	1858.260	1858.388	1858.592	1858.396
eye_open	ОЗУ, КБ	0.664	14.636	2.480	1.456	6.440
face_param	ОЗУ, КБ	2.616	126.632	114.588	117.104	118.604
frontal_pose_deviation	ОЗУ, КБ	0.080	0.340	0.264	0.268	0.080
grayscale_density	ОЗУ, КБ	1857.212	1858.664	1858.592	1858.676	1858.588
hotspots	ОЗУ, КБ	1860.084	1860.496	1861.416	1860.900	1861.324
sharpness	ОЗУ, КБ	393.036	393.656	393.504	393.164	393.724

angles	ГП, %	77	0	0	0	0
compute_characteristics		0	3	2	2	2
deviation_from_uniform_lighting		0	0	0	0	0
exposure		0	0	0	0	0
eye_open		48	0	0	0	0
face_param		56	0	0	0	0
frontal_pose_deviation		0	0	0	0	0
grayscale_density		0	0	0	0	0
hotspots		0	0	0	0	0
sharpness		0	0	0	0	0
angles	Память ГП, КБ	180224.000	0.000	0.000	0.000	0.000
compute_characteristics		0.000	0.000	0.000	0.000	0.000
deviation_from_uniform_lighting		0.000	0.000	0.000	0.000	0.000
exposure		0.000	0.000	0.000	0.000	0.000
eye_open		43008.000	0.000	0.000	0.000	0.000
face_param		180224.000	0.000	0.000	0.000	0.000
frontal_pose_deviation		0.000	0.000	0.000	0.000	0.000
grayscale_density		0.000	0.000	0.000	0.000	0.000
hotspots		0.000	0.000	0.000	0.000	0.000
sharpness		0.000	0.000	0.000	0.000	0.000

Таблица 103: PortraitEngine, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

Метод		caffe	opencv			
		потоки				
		1	1	2	4	8
angles	время, мс	92.064	80.357	78.933	48.797	33.801
compute_characteristics		2.930	2.821	2.928	2.821	2.847
deviation_from_uniform_lighting		4.534	4.352	4.357	4.357	4.336
exposure		6.606	6.628	6.488	6.426	6.431
eye_open		23.203	16.651	17.405	13.121	10.001
face_param		93.911	72.278	72.647	48.273	35.525
frontal_pose_deviation		0.094	0.083	0.083	0.082	0.082
grayscale_density		10.441	10.656	10.575	10.627	10.469
hotspots		12.492	12.464	12.507	12.555	12.429
sharpness		1.802	1.784	1.784	1.780	1.784
angles	ЦП, %	5	5	8	13	25
compute_characteristics		5	6	5	6	5
deviation_from_uniform_lighting		5	5	5	5	5
exposure		5	6	5	5	6
eye_open		5	6	8	13	28
face_param		5	5	8	13	24
frontal_pose_deviation		5	5	5	5	5
grayscale_density		5	5	5	5	5
hotspots		5	5	5	5	6
sharpness		5	5	5	5	5
angles	ОЗУ, КБ	56.988	53.096	52.984	53.576	53.520
compute_characteristics		8104.031	8104.031	8104.031	8104.031	8104.031
deviation_from_uniform_lighting		0.564	0.556	0.540	0.604	0.556
exposure		2029.160	2029.244	2029.316	2029.252	2029.248
eye_open		159.900	160.392	160.420	160.436	161.068
face_param		203.708	199.076	199.580	200.336	201.176
frontal_pose_deviation		0.292	0.288	0.292	0.320	0.292
grayscale_density		2029.236	2029.304	2029.236	2029.268	2029.236
hotspots		2030.024	2030.792	2030.744	2030.740	2030.660
sharpness		480.776	480.960	480.820	480.936	480.908

Таблица 104: PortraitEngine, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

Метод		caffe	opencv			
		потоки				
		1	1	2	4	8
angles	время, мс	4.650	71.743	71.319	48.200	33.387
compute_characteristics		2.996	2.907	3.000	3.003	2.911
deviation_from_uniform_lighting		4.576	4.457	4.444	4.455	4.418
exposure		6.635	6.526	6.500	6.531	6.502
eye_open		5.073	16.879	17.694	13.394	10.350
face_param		6.955	72.215	75.078	49.156	35.859
frontal_pose_deviation		0.174	0.157	0.153	0.163	0.154
grayscale_density		10.537	10.524	10.529	10.568	10.560
hotspots		12.532	12.490	12.493	12.605	12.499
sharpness		1.881	1.872	1.868	1.865	1.878
angles	ЦП, %	5	5	8	13	25
compute_characteristics		5	5	5	5	5
deviation_from_uniform_lighting		5	5	5	5	5
exposure		5	5	5	5	5
eye_open		5	5	8	13	25
face_param		5	6	8	13	24
frontal_pose_deviation		5	5	5	5	5
grayscale_density		5	5	5	5	5
hotspots		5	5	5	5	5
sharpness		5	5	5	5	5
angles	ОЗУ, КБ	4.672	53.272	53.220	53.460	53.556
compute_characteristics		8104.027	8104.027	8104.031	8104.027	8104.027
deviation_from_uniform_lighting		0.568	0.652	0.616	0.640	0.560
exposure		2029.308	2029.316	2029.184	2029.180	2029.352
eye_open		146.932	160.504	160.112	159.928	161.288
face_param		151.288	199.216	199.348	200.212	201.932
frontal_pose_deviation		0.336	0.456	0.468	0.304	0.360
grayscale_density		2029.264	2029.304	2029.176	2029.432	2029.384
hotspots		2030.492	2030.656	2030.616	2030.992	2030.424
sharpness		481.100	481.192	481.080	480.860	480.948

angles	ГП, %	79	0	0	0	0
compute_characteristics		1	1	1	1	2
deviation_from_uniform_lighting		0	0	0	0	0
exposure		0	0	0	0	0
eye_open		47	0	0	0	0
face_param		57	0	0	0	0
frontal_pose_deviation		0	0	0	0	0
grayscale_density		0	0	0	0	0
hotspots		0	0	0	0	0
sharpness		0	0	0	0	0
angles	Память ГП, КБ	180224.000	0.000	0.000	0.000	0.000
compute_characteristics		0.000	0.000	0.000	0.000	0.000
deviation_from_uniform_lighting		0.000	0.000	0.000	0.000	0.000
exposure		0.000	0.000	0.000	0.000	0.000
eye_open		43008.000	0.000	0.000	0.000	0.000
face_param		180224.000	0.000	0.000	0.000	0.000
frontal_pose_deviation		0.000	0.000	0.000	0.000	0.000
grayscale_density		0.000	0.000	0.000	0.000	0.000
hotspots		0.000	0.000	0.000	0.000	0.000
sharpness		0.000	0.000	0.000	0.000	0.000

Таблица 105: PortraitEngine, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

Метод		caffe	opencv			
		потоки				
		1	1	2	4	8
angles	время, мс	60.302	41.851	41.925	24.758	17.073
compute_characteristics	время, мс	1.814	1.829	1.879	1.856	1.890
deviation_from_uniform_lighting	время, мс	4.637	4.491	4.486	4.487	4.485
exposure	время, мс	5.056	5.242	5.311	5.150	5.188
eye_open	время, мс	14.704	10.241	10.410	7.198	5.671
face_param	время, мс	61.711	43.184	43.398	25.599	18.199
frontal_pose_deviation	время, мс	0.052	0.048	0.048	0.049	0.048
grayscale_density	время, мс	7.383	7.397	7.496	7.434	7.388
hotspots	время, мс	9.247	9.206	9.186	9.232	9.158
sharpness	время, мс	1.156	1.152	1.155	1.153	1.152
angles	ЦП, %	13	13	20	33	61
compute_characteristics	ЦП, %	13	13	13	13	13
deviation_from_uniform_lighting	ЦП, %	13	13	13	13	13
exposure	ЦП, %	13	13	13	13	13
eye_open	ЦП, %	13	13	19	33	66
face_param	ЦП, %	13	13	20	33	61
frontal_pose_deviation	ЦП, %	11	13	13	13	13
grayscale_density	ЦП, %	13	13	13	13	13
hotspots	ЦП, %	13	13	13	13	13
sharpness	ЦП, %	13	13	13	13	13
angles	ОЗУ, КБ	56.616	53.088	53.084	53.368	53.012
compute_characteristics	ОЗУ, КБ	8104.035	8104.031	8104.031	8104.031	8104.031
deviation_from_uniform_lighting	ОЗУ, КБ	0.616	0.600	0.532	0.532	0.532
exposure	ОЗУ, КБ	2029.792	2029.240	2029.240	2029.316	2029.248
eye_open	ОЗУ, КБ	159.760	160.300	160.236	160.520	161.044
face_param	ОЗУ, КБ	203.432	199.336	199.440	200.436	201.200
frontal_pose_deviation	ОЗУ, КБ	0.292	0.292	0.292	0.292	0.292
grayscale_density	ОЗУ, КБ	2029.096	2029.228	2029.264	2029.240	2029.232
hotspots	ОЗУ, КБ	2030.008	2030.492	2030.804	2030.432	2030.728
sharpness	ОЗУ, КБ	480.932	481.024	480.756	480.900	480.848

Таблица 106: PortraitEngine, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

Метод		caffe	opencv			
		потоки				
		1	1	2	4	8
angles	время, мс	6.159	43.476	43.861	26.048	18.463
compute_characteristics		3.652	3.534	3.557	3.586	3.659
deviation_from_uniform_lighting		6.260	6.258	6.241	6.249	6.277
exposure		7.032	7.149	6.895	6.874	6.881
eye_open		5.750	12.202	12.436	9.217	7.760
face_param		7.890	45.136	45.340	27.597	20.299
frontal_pose_deviation		1.853	1.856	1.856	1.859	1.850
grayscale_density		9.118	9.127	9.103	9.176	9.242
hotspots		11.159	11.004	10.990	10.907	10.897
sharpness		2.954	2.953	2.956	2.952	2.948
angles	ЦП, %	13	13	19	31	55
compute_characteristics		12	12	12	12	12
deviation_from_uniform_lighting		13	13	13	13	13
exposure		13	13	13	13	13
eye_open		13	13	17	26	41
face_param		13	13	20	30	52
frontal_pose_deviation		13	13	13	13	13
grayscale_density		13	13	13	13	13
hotspots		13	13	13	13	13
sharpness		13	13	13	13	13
angles	ОЗУ, КБ	5.344	53.240	53.104	53.308	52.924
compute_characteristics		8104.043	8104.039	8104.031	8104.051	8104.047
deviation_from_uniform_lighting		0.724	0.644	0.828	0.648	0.844
exposure		2029.280	2029.416	2029.424	2029.540	2029.364
eye_open		147.204	160.464	160.092	160.532	161.556
face_param		151.752	199.684	199.452	200.432	201.672
frontal_pose_deviation		0.424	0.560	0.484	0.432	0.304
grayscale_density		2029.188	2029.420	2029.376	2029.416	2029.380
hotspots		2030.580	2030.508	2030.512	2030.496	2030.472
sharpness		480.776	480.716	480.784	480.556	480.620

angles	ГП, %	40	0	0	0	0
compute_characteristics		0	1	1	1	1
deviation_from_uniform_lighting		0	0	0	0	0
exposure		0	0	0	0	0
eye_open		20	0	0	0	0
face_param		34	0	0	0	0
frontal_pose_deviation		0	0	0	0	0
grayscale_density		0	0	0	0	0
hotspots		0	0	0	0	0
sharpness		0	0	0	0	0
angles	Память ГП, КБ	158592.000	0.000	0.000	0.000	0.000
compute_characteristics		0.000	0.000	0.000	0.000	0.000
deviation_from_uniform_lighting		0.000	0.000	0.000	0.000	0.000
exposure		0.000	0.000	0.000	0.000	0.000
eye_open		37632.000	0.000	0.000	0.000	0.000
face_param		158592.000	0.000	0.000	0.000	0.000
frontal_pose_deviation		0.000	0.000	0.000	0.000	0.000
grayscale_density		0.000	0.000	0.000	0.000	0.000
hotspots		0.000	0.000	0.000	0.000	0.000
sharpness		0.000	0.000	0.000	0.000	0.000

Оценка качества изображения лица

Таблица 107: FaceQualityAssessor, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	313.935	208.399	184.707	179.475
ЦП, %	7	12	21	39
ОЗУ, КБ	99.033	118.667	88.533	59.433

Таблица 108: FaceQualityAssessor, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	24.830	19.821	18.671	16.374
ЦП, %	13	16	21	31
ОЗУ, КБ	108.467	108.667	106.733	104.200
ГП, %	42	47	47	49
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 109: FaceQualityAssessor, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	510.886	327.083	244.309	227.063
ЦП, %	6	11	21	41
ОЗУ, КБ	24.167	37.933	38.933	37.267

Таблица 110: FaceQualityAssessor, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	29.441	22.508	19.272	18.624
ЦП, %	6	11	21	41
ОЗУ, КБ	0.167	0.233	0.067	0.067
ГП, %	42	43	45	48
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 111: FaceQualityAssessor, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	317.736	198.432	137.084	140.415
ЦП, %	13	26	51	100
ОЗУ, КБ	33.633	30.900	33.667	36.933

Таблица 112: FaceQualityAssessor, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

	pt			
	потоки			
	1	2	4	8
время, мс	22.682	18.873	16.707	16.910
ЦП, %	15	25	51	99
ОЗУ, КБ	0.000	0.000	0.000	0.067
ГП, %	25	25	33	26
Память ГП, КБ	0.000	0.000	0.000	0.000

Обнаружение живости фотографии

Таблица 113: PhotoLivenessDetector, calc_liveness, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	85.752	80.096	73.613	85.588
ЦП, %	8	13	22	41
ОЗУ, КБ	412.118	274.118	150.353	143.294

Таблица 114: PhotoLivenessDetector, calc_liveness, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	15.960	15.299	14.999	15.196
ЦП, %	13	15	19	26
ОЗУ, КБ	0.824	1.529	0.824	0.824
ГП, %	40	40	40	40
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 115: PhotoLivenessDetector, calc_liveness, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	205.756	151.556	136.823	141.869
ЦП, %	6	11	21	41
ОЗУ, КБ	109.529	158.118	134.235	123.176

Таблица 116: PhotoLivenessDetector, calc_liveness, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	31.004	30.295	30.809	31.342
ЦП, %	6	11	21	40
ОЗУ, КБ	26.588	28.000	26.353	26.471
ГП, %	22	22	22	22
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 117: PhotoLivenessDetector, calc_liveness, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7- 9700 (8 ядер) @ 3.00ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	130.384	90.498	75.697	92.551
ЦП, %	13	26	51	98
ОЗУ, КБ	141.529	122.471	122.941	67.059

Таблица 118: PhotoLivenessDetector, calc_liveness, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7- 9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

	pt			
	потоки			
	1	2	4	8
время, мс	26.012	24.981	24.983	28.662
ЦП, %	13	25	51	98
ОЗУ, КБ	0.000	17.765	0.000	0.000
ГП, %	12	13	13	11
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 119: PhotoLivenessDetector, calc_liveness_with_classification, CentOS Linux 7 (Core),ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	83.983	63.859	68.272	79.524
ЦП, %	9	14	23	41
ОЗУ, КБ	55.647	27.059	47.294	26.353

Таблица 120: PhotoLivenessDetector, calc_liveness_with_classification, CentOS Linux 7 (Core), ЦП Intel(R) Xeon(R) Gold 5215 (20 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	16.464	15.687	15.465	15.711
ЦП, %	12	14	18	25
ОЗУ, КБ	39.412	15.176	1.529	22.706
ГП, %	40	40	40	40
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 121: PhotoLivenessDetector, calc_liveness_with_classification, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	204.226	150.031	137.271	141.319
ЦП, %	6	11	21	41
ОЗУ, КБ	162.706	162.353	168.471	166.588

Таблица 122: PhotoLivenessDetector, calc_liveness_with_classification, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Xeon(R) Gold 5215 (10 ядер) @ 2.50ГГц, Tesla T4

	pt			
	потоки			
	1	2	4	8
время, мс	31.937	31.331	32.152	31.975
ЦП, %	6	10	21	41
ОЗУ, КБ	100.353	99.765	78.118	98.471
ГП, %	24	25	24	24
Память ГП, КБ	0.000	0.000	0.000	0.000

Таблица 123: PhotoLivenessDetector, calc_liveness_with_classification, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц

	pt			
	потоки			
	1	2	4	8
время, мс	128.751	89.637	75.500	91.000
ЦП, %	13	26	51	98
ОЗУ, КБ	129.882	114.235	162.118	146.824

Таблица 124: PhotoLivenessDetector, calc_liveness_with_classification, Microsoft Windows Server 2016 Standard, ЦП Intel(R) Core(TM) i7-9700 (8 ядер) @ 3.00ГГц, Quadro RTX 5000

	pt			
	потоки			
	1	2	4	8
время, мс	24.923	24.669	24.663	27.172
ЦП, %	13	26	50	97
ОЗУ, КБ	25.059	89.176	19.647	24.471
ГП, %	13	13	13	12
Память ГП, КБ	0.000	0.000	0.000	0.000