Общее руководство пользователя системы «РоадАР Лица»

Содержание

Введение

Система “РоадАР Лица” – система, разработанная компанией ООО “РоадАР”, предназначенная для распознавания лиц людей в системах фотовидеофиксации и видеонаблюдения.

Текущая версия системы “РоадАР Лица” позволяет решать следующие задачи:

- идентификация изображений для анализа;
- поиск на изображении лица с помощью нейронной сети;
- уведомление о результатах поиска.

Система “РоадАР Лица” поддерживает механизмы взаимодействия со сторонними информационными системами. Текущая версия системы предполагает выбор и реализацию конкретного механизма взаимодействия с внешними системами по согласованию с заказчиками.

Функционал системы:

- обработка изображений;
- определение лиц людей;
- информационный обмен с внешними системами.

1 Назначение и условия применения система “РоадАР Лица”

1.1 Назначение программы

- Обработка изображений для определения лиц людей.

1.2 Базовый функционал система “РоадАР Лица” включает:

- обработка изображений;
- определение человеческих лиц;
- информационный обмен с внешними системами.

1.3 Программное обеспечение рабочего места

Библиотека может быть встроена в программное обеспечение, исполняемое на рабочем месте пользователя. В этом случае клиентская часть системы “РоадАР Лица” может использоваться на любом рабочем месте, имеющем подключение к сети Internet (или сети передачи данных предприятия) и установленный браузер (в среде операционной системы Windows 7 и выше). Разрешающая способность видеосистемы и монитора – не ниже 1280x1024. Рекомендуется широкоформатный монитор.

Вид ПО	Программный продукт
ОС (приведены варианты)	Windows 7 и выше Apple Mac OS X 10.6 и выше Linux Android 4.0 и выше iOS 7.0 и выше

Таблица 1. Системные требования

Данные требования могут меняться в зависимости от особенностей программного обеспечения, которое использует библиотеку системы “РоадАР Лица”.

1.4 Виды пользователей Системы

Пользователем системы является пользователь программного обеспечения, в которое встраивается библиотека системы “РоадАР Лица”. Ниже перечислены базовые минимальные пользователи, необходимые для функционирования программного обеспечения на базе библиотеки, при этом могут присутствовать другие виды пользователей. Предусмотрены следующие виды пользователей:

- *Пользователь «Администратор».* Может обладать правами на изменение основных настроек системы “РоадАР Лица”.

- *Пользователь*. Учётные записи этих пользователей создаются, удаляются и редактируются Администратором. Такой пользователь может только просматривать основные данные.

Описание доступных возможностей API выполняется для пользователя «Администратор», как обладающего максимально возможными правами по доступу к программе.

1.5 Описание Системы

API Системы написано на C++ и предоставляет несколько основных классов для распознавания лиц. Для публичного `API` используется `VideoProcess`, `FaceDB`. Все остальные классы являются внутренними.

Класс VideoProcess

Основной класс, инкапсулирующий всю логику распознавания. В случае если распознавание ведется из нескольких потоков или источников, каждый поток или источник должен иметь свой экземпляр класса `VideoProcess`.

Инициализация обработчика с информацией о нейросетях и фото пользователей системы:

```
Config config; // тут содержатся папки с фотографиями
пользователей системы, а также пути до подключаемых нейросетей
VideoProcess proc(config);
```

Чтобы обработать видео, необходимо передать кадр по очереди внутрь функции `process`. Картинка должна быть в формате `BGR(CV_8UC3)`. На выход получим текущие активные треки и при запросе отрисовку работы обратно на кадр:

```
Mat img, outImg;
img = ..; // чтение картинок из видео
RecogResult result = proc.process(img, &outImg);
```

Для получения всех треков после обработки видео, можно воспользоваться функцией:

```
std::vector<HumanTrack> getHumanTracks() const;
```

Для того чтобы получить все зарегистрированные лица в системе, можно запросить класс `FaceDB`. Через объект этого класса можно работать с зарегистрированными лицами в системе:

```
std::shared_ptr<FaceDB> faces = proc.getFaceDb();
const std::vector<FaceInfo> &faces = faceDb->getFaces();
```

Класс FaceDB

Класс для работы с базой зарегистрированных лиц в системе. Описание доступных методов ниже:

```
class FaceDB {
public:
    // @param sameFaceThresh acos расстояние между векторами, ниже
которого лица совпадают
    FaceDB(float sameFaceThresh = 0.30);

    // Поиск человека из базы по текущему распознаванию
    const FaceInfo & findPerson(const std::vector<float> &vec,
float &score);

    // Получение всех зарегистрированных лиц в системе
    const std::vector<FaceInfo> &getFaces() const;

    // Поиск лица в базе по его id
    const FaceInfo &getFace(const int faceId) const;

// ———————— Внутренние методы ————————————
// внутренний метод для добавления лиц в базу из папки
    void addImagesFromFolder(/* */);
};
```

Класс FacePoseTracker

Класс для трекинга детектов между кадрами и распознавания людей:

```
class FacePoseTracker {
public:
    FacePoseTracker(std::shared_ptr<face_detector::FaceRecognizer>
recognizer,
                    std::shared_ptr<FaceDB> faceDb);
    /// Обрабатываем новый пришедший кадр вместе с детектами(лица,
позы)
    void process(/* параметры */);
    /// Удаляем треки с историей меньше 5 кадров
    int removeTrackLessNFrames = 5;
    /// Если мы не находим человека больше N кадров, завершаем трек
    int finishTrackMissNFrames = 15;
    /// Как часто запускаем распознавание лиц для трека
    int minDistBetweenRecog = 5;
    std::vector<HumanTrack> getAllTracks();
    const std::vector<HumanTrack> &getActiveTracks();
};
```

Классы для запуска нейросетей HumanPoseEstimator, FaceRecognizer, FaceDetector

Данные классы инкапсулируют всю логику работы с нейросетями: инициализацию, препроцессинг, инференс, постпроцессинг. Все они имеют схожий интерфейс взаимодействия. Пример с `FaceDetector`:

```
class FaceDetector {
public:
    /// Количество точек на лице которое выдает детектор
    static const size_t keypointsNumber;

    /**
    * @param modelPath информация о модели: описание и веса
    * @param targetDevice запуск на CPU или GPU
    * @param maxBatch максимальный размер batch size при прогоне
нейросети
    * @param enablePerformanceReport при удалении выводим
информацию о медленных участках при инференсе модели
    */
    FaceDetector(const ModelWeightPath& modelPath,
                 const NN::Device targetDevice,
                 int maxBatch,
                 bool enablePerformanceReport = false);

    /// Прогон нейросети на оригинальном изображении
    std::vector<FaceDetect> estimate(const cv::Mat& image,
cv::Vec4i pad = cv::Vec4i());

    /// Данная функция позволяет менять разрешение для прогона
нейросети.
    /// Актуально когда к нам приходят вертикальные или
горизонтальные изображения.
    /// Можно адаптивно менять размер
    void setInputSize(cv::Size inputSize);

};
```