Общее руководство пользователя системы «РоадАР Свет фар»

Содержание

Введение

Система “РоадАР Свет фар” – система, разработанная компанией ООО “РоадАР”, предназначенная для распознавания Свет фар транспортных средств (далее - ТС) в системах фотовидеофиксации и видеонаблюдения.

Текущая версия системы “РоадАР Свет фар” позволяет решать следующие задачи:

- идентификация изображений для анализа;
- поиск на изображении ТС;
- определение Свет фар ТС с помощью нейронной сети;
- возврат сообщения о состоянии фар ТС.

Система “РоадАР Свет фар” 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 включённого света фар;
- информационный обмен с внешними системами.

1 Назначение и условия применения система “РоадАР Свет фар”

1.1 Назначение программы

- Обработка изображений для определения Свет фар ТС.

1.2 Базовый функционал система “РоадАР Свет фар” включает:

- обработка изображений;
- определение Свет фар;
- 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 Свет фар” 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 Свет фар”.

1.4 Виды пользователей Системы

Пользователем системы является пользователь программного обеспечения, в которое встраивается библиотека системы “РоадАР Свет фар”. 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 Свет фар”.

- *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 номеров.

Класс `Detector`

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 `Detector`.

Инициализация детектора без аргументов:

```
StatusInit status;
Detector det(status, std::cout);
```

Конструктор

Конструктор принимает 8 аргументов, 6 из них опциональные:

- `status` Статус инициализации детектора ГРЗ.

- `errorMessage` Выходной поток для сообщений от детектора.

- `maxTrackSize` Максимальная длина трека в кадрах, после которого трек обрывается. Нулевое значение не ограничивает длину треку.

- `numThreads` Максимальное число потоков доступных для работы. Если значение равно 0, то алгоритм будет работать без распараллеливания.

- `enableVehicleTracking` Флаг для распознавания транспортных средств отдельно от номеров

- `recognizerType` Режим работы детектора

- `roi` Область интереса на котором будет работать поиск ГРЗ

- `plateCountries` Вектор списка ГРЗ стран, которые необходимо распознавать.

После инициализации значение `status` установится в одно из следующих состояний:

```
OK                          ///< Инициализация корректна
NOT_FOUND_LICENSE_FILE      ///< Нет файла лицензии
LICENSE_INCORRECT           ///< Лицензия не верна
HARDWARE_FEATURE_NOT_DETECT ///< Нет доступа
UNKNOWN_ERROR               ///< Неизвестная ошибка
```

Метод `processFrame`

Поиск и распознавание ГРЗ на кадре.

```
std::vector<TrackResult> processFrame(const void* inputImage, int w, int h,
        bool color, const uint64_t &frameLabel);
```

Параметры:

- `inputFrame` - Входная матрица изображения
- `w` ширина изображения
- `h` высота изображения
- `color` цветная или ч/б
- `frameLabel` - метка кадра. Это числовое значение которое необходимо увеличивать перед каждым вызовом processFrame. В дальнейшем у полученных результатов будет доступна эта метка.

Использование:

Каждый вызов `processFrame` не обязан возвращать какой-то результат, он будет возвращен после того, как Detector

проследит трек номера на нескольких последовательных кадрах и проведет анализ распознанных результатов.

Пример : в видео, с кадра номер 1 по кадр номер 10 виден один и тот же номер.

Соответствующий ему `TrackResult` будет возвращен спустя несколько кадров после исчезновения номера из поля зрения.

Метод `processFrameAsync`

Отправка кадра для асинхронной обработки кадра по поиску и распознаванию ГРЗ на кадре.

```
void processFrameAsync(const void* inputImage, int w, int h, bool color, const uint64_t &frameLabel);
```

Параметры:

- `inputFrame` - Входная матрица изображения
- `w` ширина изображения
- `h` высота изображения
- `color` цветная или ч/б
- `frameLabel` - метка кадра. Это числовое значение которое необходимо увеличивать перед каждым вызовом processFrame.

В дальшейшем у полученных результатов будет доступна эта метка.

Использование:

Асинхронная обработка кадров, результаты приходят с задержкой в 6-24 кадра в зависимости от fps камеры и производительности устройства.

Для получения результатов, нужно вызывать метод getLastResults в другом потоке.

Метод `getLastResults`

Получение последних результатов распознавания. Нужно вызывать, если используется метод processFrameAsync

```

std::vector<TrackResult> getLastResults(uint64_t &frameLabel, bool &frameEnd);

```

Параметры

- `frameLabel` метка с последнего обработанного кадра

- `frameEnd` возвращает true, если был вызван метод stop и кадры перестают обрабатываться.

Метод `start`

Запуск детектора в асинхронном режиме

Метод `stop`

Остановка всех потоков для корректного завершения работы детектора. Необходимо вызывать перед вызовом деструктора детектора, если был включен асинхронный режим.

Метод `getCurrentUnfinalizedTracks`

Возвращает текущие активные треки.

`std::vector<UnfinalizedTrack>getCurrentUnfinalizedTracks();`

Использование:

Метод следует использовать чтобы получить промежуточные результаты распознавания. К примеру, в целях отладки или для визуализации.

Если подобных задач нет, то вызов этого метода не желателен, так как создает лишнюю нагрузку.

Возвращаться будут треки с минимальной длинной в 2 срабатывания. При этом трек может вернуться и позже если до этого не удавалось достаточно точно его распознать.

Метод `getVehicleTracks`

Возвращает треки всех ТС.
`std::vector<VehicleTrack> getVehicleTracks();`

Использование:

Метод следует использовать чтобы получить все треки ТС, даже без ГРЗ.

Метод `getCurrentCarTracks`

Возвращает текущие треки ТС.
`std::vector<VehicleTrack> getCurrentCarTracks();`

Использование:

Метод следует использовать чтобы получить все текущие треки ТС.

Метод `getNotRecognizedTracks`

Возвращает треки, которые не смогли распознаться.
```
std::vector<TrackResult> getNotRecognizedTracks(int minFrames,
        float minPlateSize, float minShift);
```

Использование:

Метод следует использовать, чтобы получить треки с плохо читаемыми ГРЗ.

Если подобных задач нет, то вызов этого метода не желателен, так как создает лишнюю нагрузку.

Метод принимает следующие параметры:

- `minFrames` - минимальное количество кадров необходимое чтобы трек вернулся.

- `minPlateSize` - минимальный размер ГРЗ в пикселях.

- `minShift` - минимальная геометрическая длинна трека(смещение ГРЗ между кадрами) в пикселях.

Класс `VehicleRecognizer`

Класс инкапсулирующий логику распознавания марок-моделей, состояния фар и Свет фар ТС. В случае, если распознавание ведется из нескольких потоков или источников, каждый поток или источник должен иметь свой экземпляр класса `VehicleRecognizer`.

Инициализация распознавателя с минимальным набором аргументов:

```
StatusInit vehiclesStatus;
StatusInit headlightsStatus;
StatusInit colorStatus;
Detector det(vehiclesStatus, headlightsStatus, colorStatus, std::cout);
```

Конструктор

Конструктор принимает 5 аргументов, 2 из них опциональные:

- `vehiclesStatus` Статус инициализации распознавателя марок/моделей.

- `headlightsStatus ` Статус инициализации распознавателя состояния фар.

- `colorStatus` Статус инициализации распознавателя Свет фар ТС.

- `errorMessage` Выходной поток для сообщений от распознавателя.

- `enableBrandModelType` Флаг для инициализации распознавателя марок/моделей.

- `enableHeadlights` Флаг для инициализации распознавателя состояния фар.

- `enableColor` Флаг для инициализации распознавателя Свет фар ТС.

После инициализации, значения `vehiclesStatus, headlightsStatus`, `colorStatus` установятся в одно из следующих состояний:

```
OK                          ///< Инициализация корректна
NOT_FOUND_LICENSE_FILE      ///< Нет файла лицензии
LICENSE_INCORRECT           ///< Лицензия не верна
HARDWARE_FEATURE_NOT_DETECT ///< Нет доступа
UNKNOWN_ERROR               ///< Неизвестная ошибка
```

Метод `recognizeHeadlightStatus`

Распознавание состояния передних фар на изображении по угловым точкам ГРЗ

```
HeadlightStatus recognizeHeadlightStatus(const void* inputImage, int w,
        int h,    bool color, const std::vector<RNPoint> &quadrangle,
        bool correct4Points = true);
```

Параметры:

- `inputImage` - Входная матрица изображения
- `w` ширина изображения
- `h` высота изображения
- `color` цветная или ч/б

- `quadrangle` Примерные угловые точки ГРЗ `{tl, tr, br, bl}`

- `correct4Points` Нужно ли уточнять угловые точки (повышает точность распознавания, но немного замедляет)

Возвращает состояние передних фар автомобиля.

```
UNDEFINED ///< Неопределенное
ENABLE.   ///< Фары работают
DISABLE   ///< Фары отключены
```