iANPR 2.6.0

Содержание

1. Назначение и лицензии

Назначение: iANPR2 – это комплект средств разработки (SDK) для распознавания автомобильных номеров. Возможности библиотеки включают обработку изображений. Основной язык использования библиотеки – C/C++.

Связь с прошлой версией: Несмотря на то, что при разработке iANPR2 был учтен опыт iANPR версии 1.8, iANPR2 – совершенно новый продукт, в котором полностью с нуля написан код.

Технические требования: TRT и ORT версии работают только при наличии GPU ускорителя NVIDIA.

Версии CPU работают без видеокарты.

Лицензии: В настоящей версии нет разделения по странам. Предоставляется только лицензия с максимальными возможностями, допускающая распространение со своим продуктом

iANPR2 TRT WINDOWS
iANPR2 TRT LINUX
iANPR2 ORT WINDOWS
iANPR2 ORT LINUX
iANPR2 CPU WINDOWS
iANPR2 CPU LINUX
iANPR2 ARM FULL

Ценообразование представлено на сайте: https://intbusoft.com/iANPR2

Библиотеки можно запустить без лицензии со следующими ограничениями:

- дополнительная 30-секундная задержка при инициализации объекта распознавания;
- режим 5 + 5 – первые 5 кадров распознаются, в последующих 5 кадрах вместо номеров выдается «DDDDDD», и так по циклу;
- даже с такими ограничениями вы можете использовать библиотеку лишь в ознакомительных целях согласно лицензии.

2. Функции и структуры SDK

Все функции и структуры определяются в файле **iANPR2.h**.

Объект iANPR2Object

Объект является указателем типа void и определяется следующим образом:

```
typedef void* iANPR2Object;
```

В объекте *iANPR2Object* инициализируются режимы распознавания, загружаются конфигурационные файлы и шаблоны, движки TensorRT на видеокарту или модели на центральный процессор.

Объект iANPR2TrajectoryObject

Объект является указателем типа void и определяется следующим образом:

```
typedef void* iANPR2TrajectoryObject;
```

В объекте *iANPR2TrajectoryObject* хранятся и анализируются траектории номеров отдельно от объекта iANPR2Object.

Структура iANPR2Setting

Предназначена для настройки режимов распознавания и выдачи результатов

```
struct iANPR2Setting
{
	float detectConfThresh = 0.25f;
	int minPlateHeight = 12;
	int minPlateWidth = 60;
	std::vector<std::string> templateCountries;
	bool parallelCPUfunc = false;
	bool turnNumber = true;
	int memoryNumberFrames = 4;
	int memoryNumberRepeat = 2;
	int minSymbolsPlate = 6;
	int maxSymbolsPlate = 15;
	bool mem = false;
	bool expandedResult = false;
	int typeResultNumberStr = iANPR2_RESULT_NUMBER_LATIN;
};
```

detectConfThresh - пороговое значение детектирования номера (Рекомендуется 0.25) в интервале (0,1). Используется моделью, детектирующей автомобильный номер и другие объекты.

minPlateHeight - минимальная высота номера в пикселях, меньше которой номер отбрасывается. Рекомендуется 12.

minPlateWidth - минимальная ширина номера в пикселях, меньше которой номер отбрасывается. Рекомендуется 60.

templateCountries - шаблоны стран, задается вектором строк по типу шаблонов страны, определенного в файле шаблонов, например - {"ru"}.

turnNumber – включает поворот номера. По умолчанию TRUE. Если сделать этот параметр FALSE, то функция не будет пытаться найти оптимальный поворот номера, а будет распознавать на нем текст, как есть. В случае если камера принимает четко горизонтально ориентированный номер лучше устанавливать в FALSE.

memoryNumberFrames - количество последних кадров, результаты распознавания которых хранятся в памяти для накапливаемого распознавания.

memoryNumberRepeat – количество повторений номера в памяти для накапливаемого распознавания.

minSymbolsPlate – минимальное количество символов в номере. Номера с меньшим количеством символов отбрасываются.

maxSymbolsPlate – максимальное количество символов в номере. Номера с большим количеством символов отбрасываются.

mem – Включает накапливаемое распознавание.

expandedResult – включить расширенное возвращение строки номера. В строки номера добавляются:

символ * - обозначает перенос строки в многострочных номеров;

символы [] – показывают зону региона.

typeResultNumberStr – тип содержимого строки возвращаемых номеров во всех функциях:

iANPR2_RESULT_NUMBER_FULL 0	**Все символы кроме расширенных, цифр и латинских букв возвращаются, как &0000, где 0000 - код в UTF-16. На настоящий момент это относится только к двум символам - Б и Г для номеров Беларуси**
iANPR2_RESULT_NUMBER_UTF8 1	Возвращается номер в UTF-8
iANPR2_RESULT_NUMBER_1251 2	Возвращаются номер в Windows-1251
iANPR2_RESULT_NUMBER_LATIN 3	Возвращаются только латинские символы, другие заменяются знаком '?'

Структура NumberResult

Предназначена для кадрового вывода результата распознавания – содержит в себе информацию о распознанном номере.

```
struct NumberResult
{
	cv::Rect rect;
	std::vector<std::string> strings;
	std::vector<std::string> templates;
	std::vector<float> confidence;
	float score;
};
```

rect – область распознанного номера.

strings – вероятные номера. 0-ая строка в векторе – наиболее вероятный номер, последующие возможные номера для распознанной зоны убывают по вероятности.

templates – наименование шаблонов для каждой вероятной строки соответственно. Если шаблон не найден – то имя шаблона = none. Подробнее о шаблонах в пункте 3.

confidence – уверенность (достоверность) распознавания для каждого номера из вероятных соответственно. Не является точным показателем.

score – вероятность (confidence) при детектировании номера.

Структура ObjectResult

Предназначена для кадрового вывода дополнительных результатов распознавания (не номеров).

```
struct ObjectResult
{
	cv::Rect rect;
	float score;
	int classId
	int indexNum;
};
```

rect – область детектированного объекта.

score – вероятность (confidence) при детектировании объекта.

classId – метка объекта: 1 – человек, 2 – легковой автомобиль, 3 – мотоцикл, 4 – автобус, 5 – грузовой транспорт.

indexNum – индекс номера в последовательности возвращенных номеров для того же изображения, к которому относится этот объект. Принимает значение -1 в случае, если нет соответствующего номера.

Структура NumberResultMem

Предназначена для вывода накопительного результата распознавания.

```
struct NumberResultMem
{
	std::string numberString;
	std::string numberTemplate;
	std::string numberZone;
};
```

numberString – распознанный номер.

numberTemplate – шаблон номера.

numberZone – страна.

Структура TrajectoryResult

Предназначена для вывода траектории движения номера.

```
struct TrajectoryResult
{
	std::vector<PointTrajectory> points;
	std::string strNumber;
	std::string templateNumber;
	std::string zone;
	unsigned long weights;
	unsigned long ID;
};
```

points – вектор точек в следующем формате:

```
struct PointTrajectory
{
	cv::Point point;	// Точка в кадре (левый верхний край номера)
	unsigned long weight;// Вес точки - сколько кадров в этой точке
	long time;		// Время кадра в миллисекундах от инициализации траектории
};
```

strNumber – результирующий номер для этой траектории.

templateNumber – шаблон для результирующего номера.

zone – страна или пусто, если не найден шаблон.

weights – общий вес траектории, который определяется, в скольких кадрах был распознан номер.

ID – идентификатор траектории.

Функция iANPR2Version

Возвращает строку с информацией о версии библиотеки.

```
extern "C" bool iANPR2Version(
	char* outBuf,
	int sizeBuf
);
```

Параметры:

outBuf – выходной буфера для строки версии;

sizeBuf – размер выходного буфера.

Возвращает 0 при успехе или код ошибки. Пример строки версии:

GPU - 2.5.0.0.GPUTensorRT

CPU - 2.5.0.0.CPU

Функция iANPR2Init

Инициализация объекта iANPR2.

```
extern "C" iANPR2Object iANPR2Init(
	char* configFile,
	char*license,
	int* error
);
```

Параметры:

configFile – путь до конфигурационного файла, формат которого описан ниже.

license – строка лицензии.

error – код возвращаемой ошибки.

Возвращает объект iANPR2Object или NULL (в этом случае в error возвращается код ошибки).

Формат конфигурационного файла (JSON):

```
{
    "versionconfig":2,
    "detectionmodelpath": "data/20x/platedetectionyoloxf190923.engine",
    "batchSize": 1,
    "classificationmodelpath": "data/20x/classificationf07082023.engine",
    "symbolsdetectmodelpath": "data/20x/symbolsmodelyoloxf090224.engine",
    "templatesfile": "data/platetemplates.json"
}
```

versionconfig – версия формата конфигурационного файла, оставлять как есть для текущей версии iANPR.

detectionmodelpath – путь до TensorRT движка или модели детектирования.

batchSize – размер пактов модели детектирования (должно совпадать с движком детектирования).

classificationmodelpath - путь до TensorRT движка или модели классификации.

symbolsdetectmodelpath – путь до TensorRT движка или модели детектирования символов.

templatesfile – путь до файла шаблонов номеров.

Функция iANPR2Init2

Инициализация объекта iANPR2. Альтернатива функции iANPR2Init, в которой все параметры конфигурационного файла передаются через параметры.

```
extern "C" iANPR2Object iANPR2Init2(
	char* detectionModelPath,
	int batchSize,
	char* symbolsDetectModelPath,
	char* classificationModelPath,
	char* plateTemplatesStringInJSON,
	char* license,
	int* error
);
```

Параметры:

detectionModelPath – – путь до TensorRT движка или модели детектирования.

batchSize – размер пактов модели детектирования (должно совпадать с движком детектирования).

symbolsDetectModelPath – путь до TensorRT движка или модели детектирования символов.

classificationModelPath - путь до TensorRT движка или модели классификации.

plateTemplatesStringInJSON – шаблоны номеров в JSON формате – прочитанный JSON файл шаблонов в буфер char*.

license – строка лицензии.

error – код возвращаемой ошибки.

Возвращает объект iANPR2Object или NULL (в этом случае в error возвращается код ошибки).

Функции iANPR2Release и iANPR2ReleaseP

Удаление iANPR2 объекта из памяти.

```
extern "C" void iANPR2Release(
	iANPR2Object* object
);

extern "C" bool iANPR2ReleaseP(
	iANPR2Object object
);
```

iANPR2Release принимает на вход указатель на объект, очищает его память и обнуляет object.

iANPR2ReleaseP принимает на вход объект, очищает его память и возвращает TRUE при успехе или FALSE, если объект пустой.

Функция iANPR2Settings

Установка настроек распознавания для объекта iANPR2.

```
extern "C" bool iANPR2Settings(
	iANPR2Object object,
	iANPR2Setting settings
);
```

Параметры:

object – объект iANPR2.

settings – заполненная структура iANPR2Setting.

Возвращает TRUE при успехе или FALSE, если объект пустой.

Функция iANPR2SettingsJSON

Установка настроек распознавания для объекта iANPR2 через строку в формате JSON.

```
extern "C" bool iANPR2SettingsJSON(
	iANPR2Object object,
	char* json
);
```

Параметры:

object – объект iANPR2.

json – char* строка в формате JSON. Пример такой строки в стиле Python ниже.

Возвращает TRUE при успехе или FALSE, если объект пустой или не удалось спарсить строку json. Пример строки в стиле Python:

```
settings = {
        "detectConfThresh":0.25,
           "minPlateHeight": 12,
           "minPlateWidth": 60,
           "templateCountries":[
            "ru"
            ],
           "parallelCPUfunc":0, # 0 or 1
           "memoryNumberFrames":8,
           "memoryNumberRepeat":4,
           "minSymbolsPlate": 6,
           "maxSymbolsPlate": 15,
           "mem":param2, # 0 or 1
           "expandedResult":1,# 0 or 1
           "typeResultNumberStr":3, # 0-3
    }
```

Все параметры соответствуют структуре iANPR2Setting.

Функция anpr2PLate

Принимает изображение и выполняет распознавание автомобильных номеров.

```
extern "C" int anpr2PLate(
	iANPR2Object object,
	std::vector<cv::Mat> images
);
```

Параметры:

object – объект iANPR2.

images – вектор изображений cv::Mat в формате BGR 3 канала по 8 бит. Размер вектора соответствует batchSize в конфигурации инициализации объекта.

Возвращает 0 или код ошибки. Альтернативой этой функции является комбинация функций anpr2AddImage и anpr2inference.

Функция anpr2AddImage

Первая часть альтернативы anpr2PLate, предназначенная для таких языков программирования, как Python. Помещает изображение в буфер.

```
extern "C" int anpr2AddImage(
```

```
        iANPR2Object object,
        const char* image_bytes,
        long size_image
);
```

Параметры:

object – объект iANPR2.

images_bytes - массив байт, представляющий формат BMP, JPEG, PNG или TIFF;

size_image - размер массива.

Возвращает 0 или код ошибки. Если batchSize > 1, то необходимо вызвать функцию batchSize раз, чтобы заполнить буфер.

Функция anpr2inference

Вторая часть альтернативы anpr2PLate, предназначенная для таких языков программирования, как Python. Распознает помещенные ранее изображение(-я) функцией anpr2AddImage во внутренний буфер.

```
extern "C" int anpr2inference(
        iANPR2Object object
);
```

Параметры:

object – объект iANPR2.

Возвращает 0 или код ошибки.

Функция anpr2GetResult

Получить результат распознавания.

```
std::vector<std::vector<NumberResult>> anpr2GetResult(
        iANPR2Object object
);
```

Параметры:

object – объект iANPR2.

Возвращает вектор векторов структуры NumberResult. Наружный вектор - разделяет результаты распознавания по изображениям (если оно одно, то его размер – 1). Внутренний вектор – распознанные номера в изображении.

Функция anpr2SetMemStream

Установить поток, в котором хранятся номера в памяти.

```
extern "C" int anpr2SetMemStream(
        iANPR2Object object,
        int indexStream
);
```

Параметры:

object – объект iANPR2.

indexStream – индекс потока.

Как работает: устанавливается перед добавлением изображения и его распознаванием, а также перед получением результата из памяти anpr2GetResultMem. Если всё это делается в один проход кода без переключения на другой видеопоток, то выполняется один раз. Если эту функцию не устанавливать, то поток будет всегда нулевым. Минимальный номер потока – 0, максимальный 99.

Возвращает 0 при успехе, иначе код ошибки.

Функция anpr2GetResultJSON

Получить результат распознавания в формате JSON.

```
extern "C" int anpr2GetResultJSON(
	iANPR2Object object,
	char* outBuf,
	int sizeBuf
);
```

Параметры:
object – объект iANPR2.
outBuf – выходной буфера для JSON;
sizeBuf – размер выходного буфера.

Возвращает 0 при успехе или код ошибки. При успехе в буфер записывается результат распознавания а формате JSON. Пример результата для файла image2.jpg:


```
{
	"images":[
		[
			{
				"x":233,
				"y":286,
				"width":192,
				"height":54,
				"strings":[
					"Y758CC56"
				],
				"templates":[
					"base_1"
				],
				"zones":[
					"ru"
				],
				"confidence":[
					0.983965
				],
				"score":0.889966
			}
		]
	]
}
```


Наименования соответствуют структуре NumberResult (кроме rect, который расписан отдельно).

Функция anpr2GetAddResult

Получить результат распознавания дополнительных объектов.

```
std::vector<std::vector<ObjectResult>> anpr2GetAddResult(
	iANPR2Object object
);
```

Параметры:
object – объект iANPR2.

Возвращает вектор векторов структуры ObjectResult. Наружный вектор - разделяет результаты распознавания дополнительных объектов по изображениям (если оно одно, то его размер – 1). Внутренний вектор – распознанные дополнительные объекты в изображении.

Функция anpr2GetAddResultJSON

Получить результат распознавания дополнительных объектов в формате JSON.

```
extern "C" int anpr2GetAddResultJSON(
	iANPR2Object object,
	char* outBuf,
	int sizeBuf
);
```

Параметры:
object – объект iANPR2.
outBuf – выходной буфера для JSON;
sizeBuf – размер выходного буфера.

Возвращает 0 при успехе или код ошибки. При успехе в буфер записывается результат распознавания а формате JSON. Пример результата для файла image2.jpg:

```
{
	"images":[
		[
			{
				"x":66,
				"y":5,
				"width":563,
				"height":403,
				"score":0.943103,
				"classid":2,
				"imagenum":0
			}
		]
	]
}
```

Наименования соответствуют структуре ObjectResult (кроме rect, который расписан отдельно).

Функция anpr2GetResultMem

Получить результат распознавания из суммирования в памяти.

```
std::vector<std::vector<NumberResultMem>> anpr2GetResultMem(
	iANPR2Object object
);
```

Параметры:
object – объект iANPR2.

Возвращает вектор векторов структуры NumberResultMem. Наружный вектор - разделяет результаты распознавания по изображениям (если оно одно, то его размер – 1). Внутренний вектор – распознанные номера в изображении.

Функция anpr2GetResultMemJSON

Получить результат распознавания из суммирования в памяти в формате JSON.

```
extern "C" int anpr2GetResultMemJSON(
	iANPR2Object object,
	char* outBuf,
	int sizeBuf
);
```

Параметры:

object – объект iANPR2.
outBuf – выходной буфера для JSON;
sizeBuf – размер выходного буфера.
Возвращает 0 при успехе или код ошибки. При успехе в буфер записывается результат распознавания а формате JSON. Пример результата:

```
{
	"numbers":[
		[
			{
				"string":"A222AA02",
				"template":"base_1",
				"zone":"ru"
			}
		]
	]
}
```

Функция iANPR2Trajectory

Создает объект iANPR2TrajectoryObject для хранения и анализа траектории. Один объект соответствует одному видеопотоку – последовательности кадров с одной камеры или видеофайла.

```
extern "C" iANPR2TrajectoryObject iANPR2Trajectory(
	int widthImage,
	int heightImage,
	int tLimitTime,
	int tLimitFrames,
	int typeSpeed
);
```

Параметры:
widthImage – ширина изображения видеопотока.
heightImage – высота изображения видеопотока.
tLimitTime – время до устаревания (в миллисекундах), например 10000.
tLimitFrames – количество кадров до устаревания, например 100.
typeSpeed – тип при вычислении скорости. 0 - по времени, 1 - по кадру.
Возвращает объект iANPR2TrajectoryObject.

Функция iANPR2TrajectoryRelease

Уничтожает объект iANPR2TrajectoryObject.

```
extern "C" bool iANPR2TrajectoryRelease(
	iANPR2TrajectoryObject object
);
```

Принимает на вход объект, очищает его память и возвращает TRUE при успехе или FALSE, если объект пустой.

Функция iANPR2TrajectoryProcess

Добавление результата распознавания кадра в траекторию.

```
bool iANPR2TrajectoryProcess(
	iANPR2TrajectoryObject object,
	std::vector<NumberResult> numbers,
	std::vector<ObjectResult> objects
);
```

Параметры:

object – объект iANPR2TrajectoryObject.
numbers – распознанные номера для одного изображения.
objects – распознанные дополнительные объекты для одного изображения.

Функция iANPR2TrajectoryProcessJSON

Добавление результата распознавания кадра в траекторию с помощью строки JSON.

```
extern "C" bool iANPR2TrajectoryProcessJSON(
	iANPR2TrajectoryObject object,
	char* jsonNumbers,
	char* jsonObjects
);
```

Параметры:
object – объект iANPR2TrajectoryObject.
jsonNumbers – распознанные номера в формате JSON, возвращенные anpr2GetResultJSON в формате UTF-8 или ASCII. При получении результата anpr2GetResultJSON, typeResultNumberStr в iANPR2Setting должно принимать следующие допустимые значения: iANPR2_RESULT_NUMBER_FULL, iANPR2_RESULT_NUMBER_UTF8, iANPR2_RESULT_NUMBER_LATIN. T.e. 0, 1 или 3.
jsonObjects – распознанные дополнительные объекты в формате JSON, возвращенные anpr2GetAddResultJSON в формате ASCII.
Возвращает TRUE при успехе или FALSE, если объект пустой или невозможно прочитать JSON строку.

Функция iANPR2TrajectoryGetResult

Получить текущие траектории из объекта траекторий iANPR2TrajectoryObject.

```
std::vector<TrajectoryResult> iANPR2TrajectoryGetResult(
	iANPR2TrajectoryObject object
);
```

Параметры:
object – объект iANPR2TrajectoryObject.
Возвращает вектор траекторий Структура TrajectoryResult.

Функция iANPR2TrajectoryGetResultJSON

Получить текущие траектории из объекта траекторий iANPR2TrajectoryObject в формате JSON.

```
extern "C" int iANPR2TrajectoryGetResultJSON(
	iANPR2TrajectoryObject object,
	char* outBuf,
	int sizeBuf
);
```

Параметры:
object – объект iANPR2TrajectoryObject.
outBuf – выходной буфера для JSON.
sizeBuf – размер выходного буфера.
Возвращает 0 при успехе или код ошибки. При успехе в буфер записывается результат распознавания а формате JSON. Пример результата:

```
{
```

```
"trajectories":[
	{
		"points":[
		[2298,1596,1,147],[2298,1596,6,1096],[2298,1606,2,1410],[2316,1620,1,
	1571],[2334,1630,1,1739],[2328,1640,1,1901],[2346,1650,1,2073],[2346,1660,1
	,-2061],[2352,1667,1,-1903],[2376,1677,1,-1750],[2388,1684,1,-
	1595],[2394,1694,2,-1285],[2406,1707,1,-1129],[2424,1717,1,-
	966],[2430,1721,1,-806],[2436,1728,1,-635],[2448,1741,1,-
	476],[2472,1755,1,-322],[2472,1768,1,-171],[2484,1778,1,-
	14],[2502,1798,1,145],[2520,1812,1,298],[2532,1825,1,459],[2538,1836,1,624]
	,[2556,1852,1,791],[2568,1869,1,961],[2574,1879,1,1128],[2592,1893,1,1293],
	[2604,1910,1,1452],[2622,1927,1,1614],[2640,1940,1,1780],[2652,1957,1,1976]
	,[2664,1971,1,-2146],[2682,1987,1,-1943],[2694,2008,1,-1774],[2718,2028,1,-
	1598],[2736,2041,1,-1429],[2754,2065,1,-1249],[2778,2085,1,-
	1074],[2790,2109,1,-
	903]],"strNumber":"LM07MK0","templateNumber":"none","weights":47,"ID":0},{"
	points":[[1368,1866,1,-733],[1362,1866,1,-567],[1362,1863,2,-
	226],[1362,1859,1,-
	45],[1362,1859,1,142],[1362,1856,1,324],[1362,1856,3,860]],"strNumber":"0F5
	6WAA","templateNumber":"none","weights":10,"ID":1}]}
```


Формат points:
[x,y,weight,time]

Функция iANPR2CreateLineIntersection

Создать линии пересечения для траектории. На самом деле линия состоит из двух линий - line1 и line2. И пересечение фиксируется только тогда, когда пересекаются обе(сделано для исключения ложных срабатываний.

```
extern "C" int iANPR2CreateLineIntersection(
	iANPR2TrajectoryObject object,
	int line1X1,
	int line1Y1,
	int line1X2,
	int line1Y2,
	int line2X1,
	int line2Y1,
	int line2X2,
	int line2Y2
);
```

Параметры:
object – объект iANPR2TrajectoryObject.
line1X1 – X координата 1 точки 1 линии.
line1Y1 – Y координата 1 точки 1 линии.
line1X2 – X координата 2 точки 1 линии.
line1Y2 – Y координата 2 точки 1 линии.
line2X1 – X координата 1 точки 2 линии.
line2Y1 – Y координата 1 точки 2 линии.
line2X2 – X координата 2 точки 2 линии.
line2Y2 – Y координата 2 точки 2 линии.
Возвращает 0 или ошибку.

Функция iANPR2GetLineIntersection

Узнать есть ли пересечение линии для конкретного индекса траектории. Общее количество траекторий и сами траектории берутся из iANPR2TrajectoryGetResult или iANPR2TrajectoryGetResultJSON

```
extern "C" int iANPR2GetLineIntersection(
```

```
        iANPR2TrajectoryObject object,
        int indexTr
);
```

Параметры:

object – объект iANPR2TrajectoryObject.

indexTr – индекс траектории.

Возвращает 0 - не пересекло, 1 - пересек сверху-вниз, 2 - снизу вверх, -1 или пустой объект или неправильный индекс.

3. Шаблоны номеров и правила их построения

Предназначение шаблонов:

- определить, что в номере: буква О или цифра 0 (ноль), т.к. при классификации эти символы не отличаются;
- определение шаблона, говорит о большей вероятности этого номера, чем номер без шаблона;
- немного повысить достоверность распознавания за счет выбрасывания недопустимых символов.

Сейчас в стандартном файле шаблонов platetemplates.json описаны шаблоны следующих стран:

- Российская Федерация (ru).
- Азербайджан (az).
- Армения (am).
- Республика Беларусь (by).
- Казахстан (kz).
- Кыргызстан (kg).
- Таджикистан (tj).
- Узбекистан (uz).

Однако распознавание работает и без шаблонов, но в этом случае всегда будет выдаваться цифра 0 (буквы О никогда не будет).

Файл шаблонов в формате JSON достаточно прост и можно дополнить его для номеров другой страны самостоятельно.

ВНИМАНИЕ: В шаблонах используйте только буквы латинского алфавита.

При этом на настоящий момент алфавит классификации следующий:
`0123456789ABCDEFGHIJKLMNPQRSTUVWXYZБГ`

ВНИМАНИЕ: Помимо этих символов в распознавании могут выдаваться символы #(знак, что обнаружен иероглиф) и ?(символ не определен).

versiontemplates

versiontemplates определяет версию формата шаблона, нужно оставлять подходящую для текущей версии iANPR.

description

description – информационный блок, не влияет на распознавание.

zones

zones определяет зоны распознавания, в перспективе их можно указывать несколько при распознавании и тогда по всем зонам будет осуществляться попытка соответствия шаблону.

```
"zones": [
        "ru"
    ],
```

Для каждой зоны обязательны ключи в JSON файле:
«Имя зоны», например `"ru"`
«Имя зоныtree», например `"rutree"`

Имя зоныtree

В «Имя зоныtree» представлена иерархия шаблонов номеров этой страны, верхним разделением является разделение по отношению яркости символа к фону:
"base" – символы темные на светлом фоне;
"inverted" – символы светлые на темном фоне.

Цвета номеров, такие как красный, синий и т.п. пока не используются. Номер обрабатывается в градациях серого. Вторым разделением является разделение на однострочные и двустрочные номера:
"lines1";
"lines2".

Далее перечислены имена шаблонов.

Имя зоны

В ключе «Имя зоны» перечислены описания для всех имен данной зоны, описанных в «Имя зоныtree». Например, описание шаблона base_1A:

```
 }
     "name": "base_1A",
     "alphabet": "rubase",
     "lines": 2,
     "struct": "A000*AA|100",
     "alignment": "DDDD*DD|UUU",
     "mandatory": "####*##|###",
     "height": "1111*11|222",
     "heights": [ 0.927, 0.707 ],
     "Block2Size": 0.44,
     "type_left_sign": -1,
     "prob": 0.5
 }
```

Здесь:

name – имя шаблона.

alphabet – имя алфавита (см. про алфавит ниже).

lines – количество строк номера (1 или 2).

struct – структура номера. A – обозначает местоположение буквы; 0 – местоположение цифры; 1 – местоположение цифры, которой может и не быть;* - разделение строк номера; | - указывает на отделение блока (региона) от текущего расположения вправо до конца строки,] - указывает на отделение блока (региона) от текущего расположения влево до начала строки. Пример с регионом]:

```
"struct": "000*00]AAA",
```

order – необязательная строка - порядок возвращаемых символов. По умолчанию порядок последовательный и эта строка не нужна. Однако, например для двухстрочных номеров Казахстана это необходимо. В случае, если добавляется строка order, то в ней сохраняется структура * и |, и] аналогично struct. На месте символов, которые остаются последовательными в возвращении – символ 0. На месте перемещаемых вправо 1, 2 и т.п.. Пример, который переносит первые два символа второй строки вправо:

```
"order": "000*12]000",
```

alignment – выравнивание. D – символы выравниваются по нижней границе символа; U – символы выравниваются по верхней границе символа, * и |, и] аналогично struct.

mandatory – обязательность символа. # - символ может быть любой из алфавита, соответствующему цифре или символу в struct; * и |, и] аналогично struct. Если символы заданы жестко, как например в шаблоне "diplomat_9", то эта секция будет выглядеть так: "mandatory": "###CD#|##", где C и D – конкретные обязательные символы из алфавита.

height – соответствие символа номеру шаблона в heights (нумерация с 1, а не с 0). * и | аналогично struct.

heights – высоты символов относительно высоты самого номера.

widthsize – необязательный блок, предназначен для лучшего отнесения к шаблонам при похожих по структуре номерах. Например, uzbase2_2008_org и uztrailer2_2008 отличаются только тем, что последний символ в верхней строке у одного буква, а у другого шаблона – цифра. И естественно они могут быть цифра 0 или буква O. Но поскольку расстояние между символами отличается, то можно добавить этот блок для точного отнесения к шаблону. В этом случае 0 – означает не учитывать символ (обязательно для левых символов), 1 и 2 – разные расстояния до предыдущего символа, относительные расстояния в widthsizes.

widthsizes – заданные относительные расстояния между символами. Относительные, означает, что они относительны друг друга. Так [10, 20] – означает расстояние 1 и 2.

Block2Size – максимальный размер второй секции (региона). *Пока не используется*.

type_left_sign – есть ли в номере слева информация о стране. -1 - *не используется*. 1 – используется номеров, где слева название страны ограничено блоком, напоминающем символ I. Пока используется только для однострочных номеров Армении:

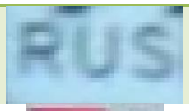
prob – вероятность шаблона номера. *Пока не используется.*

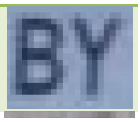
usefeatures

usefeatures – показывает использовать ли особенности номера при отнесении к зоне. 0 – не использовать, 1 – использовать.

zonesfeatures

zonesfeatures – содержит разбитые особенности по странам. На текущий момент есть такие особенности:

Название особенности	Картинка особенности
feat_ru	
feat_am	
feat_ams	
feat_az	

feat_by	
feat_kg	
feat_kz	
feat_tj	
feat_uz	
feat_blue	 или  и т.п.

Особенности используются как необязательный признак. Т.е. если какая-то особенность найдена, то этот номер принадлежит именно этой стране. Но если не найдена никакая особенность, то перебираются шаблоны всех стран. Добавлять свои особенности нельзя – они прописаны в модели обучения.

alphabets

В ключе `"alphabets"` перечислены названия алфавитов, которые могут быть использованы в шаблонах.

alphabetsinfo

В ключе `"alphabetsinfo"` – алфавиты для этих названий, при этом алфавит применим только для латинских символов из полного набора классификации, а символы цифр не задаются никогда, т.к. считаются, что они всегда используются.

Для нелатинских символов используется код &0000, где 0000 - код в UTF-16. Пример алфавита для номеров полиции Республики Беларусь:

```
"BCHKMOT&0411&0413"
```

&0411 – обозначает букву Б, &0413 – обозначает букву Г. Другие нелатинские символы пока не поддерживаются.

ВНИМАНИЕ: При использовании шаблонов можно столкнуться с ситуацией, когда номера других стран пытаются подстроиться под шаблоны, например буква A заменяется на 4, буква B заменяется на 8 и т.п. Способов решения 2: 1) расширить шаблоны под все встречающиеся номера 2) использовать дополнительно второй необработанный результат распознавания, который обязательно возвращается в таком случае.

ВНИМАНИЕ: Не все номера, поставляемые в шаблоне нужно использовать. Например, система распознавания находится в России, но могут заезжать, например, номера из Казахстана. В этом случае в `kztree` можно удалить все шаблоны номеров, которые принципиально не могут появиться в России.

Например, очевидно, что номера полиции Казахстана не могут встречаться в России, а вот номер грузовиков – вполне.

4. Инсталляция и использование в Windows

Весь необходимый для тестового запуска набор библиотек (кроме распространяемого пакета Visual Studio) находится в папке minrun. Там можно проверить работу утилиты.

Инсталляция для версии TRT

Если необходимо не только запускать, но и конвертировать из ONNX моделей в движки TensorRT, то нужно установить все пакеты.

Необходимые пакеты и ссылки на них (ссылки с течением времени могут меняться)

1. Visual Studio 2019 redist (распространяемый пакет):
https://aka.ms/vs/17/release/vc_redist.x64.exe
2. OpenCV 4.7:
https://github.com/opencv/opencv/releases/download/4.7.0/opencv-4.7.0-windows.exe
3. CUDA 12.9:
https://developer.download.nvidia.com/compute/cuda/12.9.0/local_installers/cuda_12.9.0_576.02_windows.exe
4. cuDNN 9.10(Вы должны согласиться с лицензией):
https://developer.download.nvidia.com/compute/cudnn/9.10.1/local_installers/cudnn_9.10.1_windows.exe
5. TensorRT 10.11 (Вы должны согласиться с лицензией):
https://developer.nvidia.com/downloads/compute/machine-learning/tensorrt/10.11.0/zip/TensorRT-10.11.0.33.Windows.win10.cuda-12.9.zip

Если создание движков не требуется, то вы можете пользоваться dll, идущими в поставке.

Инсталляция для CPU

Не требуется. Используется пакет OpenCV 4.7 (см. выше) и OpenVINO 2024.2.0:

https://storage.openvinotoolkit.org/repositories/openvino/packages/2024.2/

Инсталляция ONNX Runtime

Не требуется. Используется пакет OpenCV 4.7 (см. выше), CUDA 12.9 (см. выше), cuDNN 9.10 (см. выше) и ONNX Runtime GPU 1.22:

https://github.com/microsoft/onnxruntime/releases/download/v1.22.0/onnxruntime-win-x64-gpu-1.22.0.zip

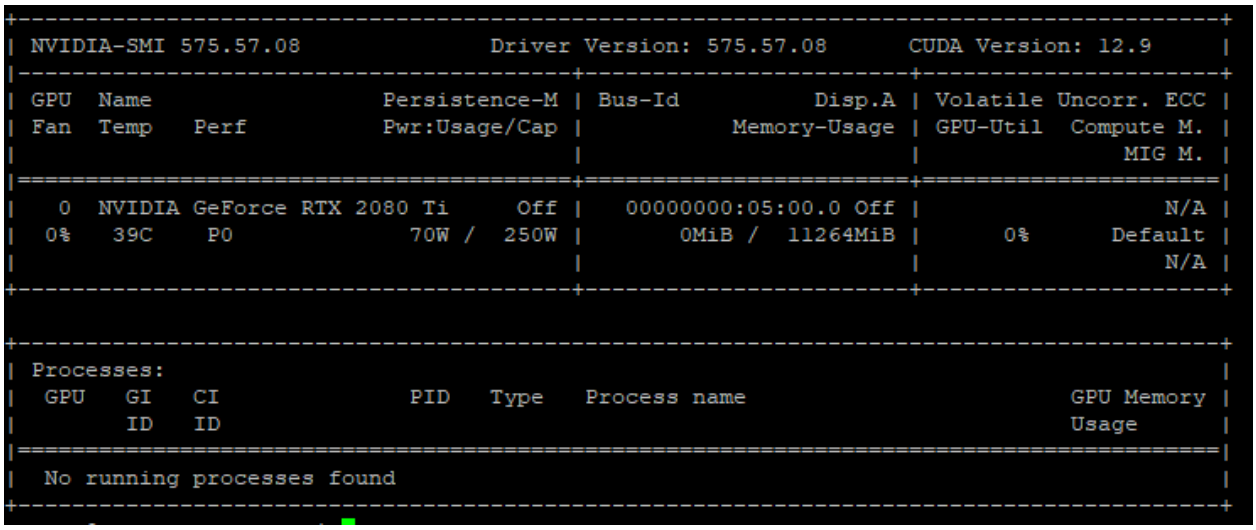
Присоединение библиотеки к проекту C++:

Необходимо подключение opencv_world470.lib и ianpr2.lib (или ianpr2cpu.lib, или ianpr2ort.dll).

5. Инсталляция и использование в Linux

5.1. Инсталляция TRT версии

Пример инсталляции для Ubuntu 22.04 с установленным уже драйвером nvidia-driver-575, CUDA 12.9, cuDNN 8.9.7. Версия C++ компилятора: g++ 11.4.0.



Шаг 1. Компиляция и установка OpenCV (необязателен, если вы просто скопируете нужные so файлы и include из релиза iANPR2).

Процесс компиляции без FFMPEG описан здесь:
https://docs.opencv.org/4.x/d7/d9f/tutorial_linux_install.html

```
# Install minimal prerequisites
sudo apt update && sudo apt install -y cmake g++ wget unzip
# Download and unpack sources
wget -O opencv.zip https://github.com/opencv/opencv/archive/4.7.0.zip
unzip opencv.zip
# Create build directory
mkdir -p build && cd build
# Configure
cmake  -DBUILD_opencv_world=ON  ../opencv-4.7.0
# Build
cmake --build .
```

При простом копировании из релиза нужно скопировать so файлы в папку /usr/local/lib:
cp /home/ubuntu/ianpr2/opencvso/libopencv_world* /usr/local/lib

Шаг 2. Установка TensorRT

Процесс установки описан здесь:
https://docs.nvidia.com/deeplearning/tensorrt/install-guide/index.html

Пакет для установки находится тут:

https://developer.download.nvidia.com/compute/tensorrt/10.11.0/local_installers/nv-tensorrt-local-repo-ubuntu2204-10.11.0-cuda-12.9_1.0-1_amd64.deb

Закачиваем этот пакет.

В результате у вас должен появиться файл:

```
nv-tensorrt-local-repo-ubuntu2204-10.11.0-cuda-12.9_1.0-1_amd64.deb
```

Далее добавляем этот пакет в видимый:

```
os="ubuntu2204"
tag="10.11.0-cuda-12.9"
sudo dpkg -i nv-tensorrt-local-repo-${os}-${tag}_1.0-1_amd64.deb
sudo cp /var/nv-tensorrt-local-repo-${os}-${tag}/*-keyring.gpg
/usr/share/keyrings/
sudo apt-get update
```

И инсталлируем его:

```
sudo apt-get install tensorrt
```

Шаг 3. Установка iANPR 2

Загрузите файл ianpr2.zip в корень домашнего каталога (у нас это /home/ubuntu). И разархивируйте:

```
unzip ianpr2.zip
```

Откомпилируйте пример C++:

```
cd ianpr2/testutils/testutil
make
```

Файл testutil должен создаться в папке /home/ubuntu/ianpr2/bin

Скопируйте файл lic2.key с лицензией в туже папку.

Теперь остались только движки TensorRT, которые можно или получить из ONNX моделей или скопировать готовые в папку

```
/home/ubuntu/ianpr2/bin/data/card
```

Проверьте работоспособность, запустив пример:

```
./testutil -i ianpr2dataconfig.json image2.jpg
```

Последняя строка вывода примера должна быть такая:

Y758CC56

Для запуска примера на Python, предварительно установите пакет OpenCV для Python:

```
pip install opencv-python
```

После чего:

```
python3 ianpr2testL.py
```

Должен быть такой результат:

```
INFO:root:Object iANPR2 loaded
INFO:root:iANPR2Settings installed
INFO:root:Image added
INFO:root:Inference true
{"images":[[{"x":233,"y":286,"width":192,"height":54,"strings":["Y758CC56"],"template
s":["base_1"],"confidence":[0.983901],"score":0.889448}]]}
```

```
INFO:root:Object iANPR2 deleted
```

5.2. Инсталляция CPU версии

Шаг 1.
Такой же, как для TRT версии.
Шаг 2. Установка OpenVINO
Ссылка на документацию установки:
https://docs.openvino.ai/2024/get-started/install-openvino/install-openvino-archive-linux.html
Используйте версию 2024.2.0. Для Ubuntu 20 – это:

```
curl
https://storage.openvinotoolkit.org/repositories/openvino/packages/2024.2/linux/l_openvino_toolkit_ubuntu20_2024.2.0.15519.5c0f38f83f6_x86_64.tgz --output
openvino_2024.2.0.tgz
```

Распакуйте в /opt/intel/

```
tar -xf openvino_2024.2.0.tgz
sudo mv l_openvino_toolkit_ubuntu20_2024.2.0.15519.5c0f38f83f6_x86_64
/opt/intel/openvino_2024.2.0
```

Установите дополнительные требования:

```
cd /opt/intel/openvino_2024.2.0
sudo -E ./install_dependencies/install_openvino_dependencies.sh
```

Установка ссылки для простоты:

```
cd /opt/intel
sudo ln -s openvino_2024.2.0 openvino_2024
```

Конфигурация окружения:

```
source /opt/intel/openvino_2024/setupvars.sh
```

Шаг 3.
Такой же, как для TRT версии.

5.3. Инсталляция ONNX Runtime версии

Используется то же драйвер, что и TRT-версии. CUDA 12.9, CUDNN 9.10.2. Пример установки CUDNN:

```
wget
https://developer.download.nvidia.com/compute/cudnn/9.10.2/local_installers/cudnn-local-repo-ubuntu2204-9.10.2_1.0-1_amd64.deb
sudo dpkg -i cudnn-local-repo-ubuntu2204-9.10.2_1.0-1_amd64.deb
sudo cp /var/cudnn-local-repo-ubuntu2204-9.10.2/cudnn-*-keyring.gpg
/usr/share/keyrings/
sudo apt-get update
sudo apt-get -y install cudnn
```

Если у вас до этого стоял CUDNN 8, то его надо удалить:

```
sudo apt purge libcudnn8* libcudnn9* -y
sudo apt autoremove -y
```

Шаг 1.
Такой же, как для TRT версии
Шаг 2. Установка ONNX Runtime

Скачать нужную версию:
wget https://github.com/microsoft/onnxruntime/releases/download/v1.22.0/onnxruntime-linux-x64-gpu-1.22.0.tgz
Распаковка и перенос в /opt/onnxruntime:
tar -xzf onnxruntime-linux-x64-gpu-1.22.0.tgz
sudo mv onnxruntime-linux-x64-gpu-1.22.0 /opt/onnxruntime
Добавьте пути к библиотекам:
echo 'export ONNXRUNTIME_HOME=/opt/onnxruntime' >> ~/.bashrc
echo 'export LD_LIBRARY_PATH=$ONNXRUNTIME_HOME/lib:$LD_LIBRARY_PATH' >> ~/.bashrc
source ~/.bashrc

Шаг 3.

Такой же, как для TRT версии.

5.4. Инсталляция на ARM микроПК с нуля

Использовалось два тестовых устройства:
Orange Pi 3 LTS
Orange Pi 5B
Использовалась операционная система Armbian:
Orange Pi 3 LTS: https://stpete-mirror.armbian.com/archive/orangepi3-lts/archive/Armbian_23.8.3_Orangepi3-lts_jammy_current_6.1.53_minimal.img.xz
Orange Pi 5B: https://dl.armbian.com/orangepi5/archive/Armbian_24.2.1_Orangepi5_jammy_legacy_5.10.160_xfce-amazingfated_desktop.img.xz

Шаг 1. Образ заливается на SD-карту.
Это можно сделать, например, с помощью Rufus:
https://rufus.ie/ru/
Для работы с WI-FI в Orange Pi 5B нужно добавить строку в файл /boot/armbianEnv.txt:
overlays=orangepi-5-ap6275p
Шаг 2. Устанавливаем минимальные пакеты

```
sudo apt update
sudo apt install -y cmake g++ wget unzip
sudo apt install build-essential
```

Шаг 3. Установка OpenVINO

Качаем OpenVINO и устанавливаем:
curl -L https://storage.openvinotoolkit.org/repositories/openvino/packages/2023.3/linux/l_openvino_toolkit_ubuntu18_2023.3.0.13775.ceeafaf64f3_arm64.tgz -O openvino_2023.3.0.tgz
Распакуйте в /opt/intel/
Установите дополнительные требования:
cd /opt/intel/openvino_2023.3.0
sudo -E ./install_dependencies/install_openvino_dependencies.sh

Установка ссылки для простоты:
cd /opt/intel
sudo ln -s openvino_2023.3.0 openvino_2023

Конфигурация окружения:
source /opt/intel/openvino_2023/setupvars.sh

Шаг 4. Установка OpenCV

Можно как раньше, или просто скопируйте libopencv_world.so.4.7.0 в /usr/local/lib и сделайте на него ссылку

sudo ln –s libopencv_world.so.4.7.0 libopencv_world.so

Затем ldconfig

5.5. Установка в Astra Linux 1.7

В целом работа под Astra Linux не отличается от Linux версии. Однако необходимо самостоятельно откомпилировать OpenVINO или при возможности просто скопировать из релиза so файлы библиотеки.

Установка OpenVINO

Клонирование репозитория:

```
git clone -b releases/2023/3 --single-branch https://github.com/openvinotoolkit/openvino.git
```

Установка дополнительных модулей:

```
cd openvino
git submodule update --init --recursive
```

Подготовка папки для сборки:

```
mkdir build && cd build
```

Подготовка к сборке:

```
/usr/local/bin/cmake -DCMAKE_BUILD_TYPE=Release ..
```

Сборка:

```
/usr/local/bin/cmake --build .
```

Установка:

```
sudo make install
```

Посмотреть куда установились so файлы. Могут быть тут:

```
/usr/local/runtime/lib/intel64
```

Прописываем этот путь в какой-либо файл в /etc/ld.so.conf с набираем:

ldconfig

6. Примеры C++

Все примеры работы находятся в одном файле ianpr2test.cpp. Информацию о возможностях примера можно получить запустив без параметров ianpr2test.exe (Windows) или testutil (Linux). Будет выдана информация:

```
(C) 2023. IntBuSoft. ianpr2test - test util for iANPR2
Modes:
 -i    - one image test: ianpr2test -i <path config> <path to image> [-w]
 -i2   - one image test with objects and extended result: ianpr2test -i2 <path config> <path to image>
 -v - video test: ianpr2test -v  <path to video>
 -vt   - video trajectory test: ianpr2test -v  <path to video>
 -p1   - performance 1 batch test: ianpr2test -p1 <path config>
 -p1pc - performance 1 batch test with dll CPU parallel: ianpr2test -p1pc <path config>
 -p1o  - performance 1 batch test with object parallel: ianpr2test -p1o <path config>
 -p4   - performance 4 batch test: ianpr2test -p4 <path config>
 -p4pc - performance 4 batch test with dll CPU parallel: ianpr2test -p4pc <path config>
 -p4o  - performance 4 batch test with object parallel: ianpr2test -p4o <path config>
 -d - directory of images test: ianpr2test -d <path config> <path to directory> <path to output directory>
```

6.1. Тест изображения –i и -i2

Вызывает функцию test1Image(). На основе этого примера покажем, как происходит работа с iANPR2.

Процесс инициализации iANPR2:

```
// Инициализация iANPR2
int error;
iANPR2Object ia2 = iANPR2Init((char*)cfgname, license,&error);
```

Здесь cfgname имя с путем до конфигурационного файла, формата описанного тут. license – это текст лицензии, для примера он загружается из lic2.key таким образом:

```
string lic;
char* license;

ifstream in("lic2.key");
if (!in.is_open())
{
    cout << "Cann't find lic2.key file!:\n";
    return 0;
}
getline(in, lic);
in.close();
license = (char*)lic.c_str();
```

В error же будут возвращаться ошибки или 0, если ошибок нет. Проверка безошибочности инициализации:

```
if (ia2 == NULL || error != int(IANPR2::iANPR2Errors::IA_OK))
{
    std::cout << "Cann't init iANPR2Object! Error = " << error << " \n";
    return;
}
```

После этого нужно установить настройки распознавания (необязательно менять все, т.к. они определен по умолчанию в **iANPR2.h**

```
iANPR2Setting settings{ 0.25, 12, 60,{"ru"},parallelDll };
iANPR2Settings(ia2, settings);
```

Изображения формируются в вектор, даже если оно всего одно. Размер вектора зависит от batchSize, определенный при инициализации.

```
std::vector <cv::Mat> imgs;
imgs.push_back(cv::imread(name));
```

После этого изображение(-я) в векторе передаются на распознавание:

```
int result = anpr2PLate(ia2, imgs);
if (result != int(IANPR2::iANPR2Errors::IA_OK))
    std::cout << "Inference problem! Error = " << result << " \n";
```

Результаты распознавания получаются следующим образом:

```
vector<vector<NumberResult>> numbers = anpr2GetResult(ia2);
```

Ну и в конце работы происходит освобождение объекта iANPR2:

```
iANPR2Release(&ia2);
```

Запуск –i2 реализован в test1ImageFull и отличается установкой расширенной выдачи результата:

```
settings.expandedResult = true;
```

А также получением дополнительных объектов:

```
vector<vector<ObjectResult>> objects = anpr2GetAddResult(ia2);
```

6.2. Тест производительности одного изображения -p1

Данный тест запускает несколько раз распознавание изображения image2.jpg и высчитывает на каждом тесте среднее время. Пример теста на Windows 10 конфигурация i7-6700K 4.0ГГц 4 ядра и RTX 2080 Super:

TRT

```
D:\ianpr2\ianpr2CPP\bin>ianpr2test.exe -p1 ianpr2dataconfig.json
[Test0=0.00927]
[Test1=0.00840]
[Test2=0.00835]
[Test3=0.00834]
[Test4=0.00860]
[Test5=0.00857]
[Test6=0.00823]
[Test7=0.00831]
[Test8=0.00826]
[Test9=0.00834]
```

CPU (small)

```
[Test0=0.02823]
[Test1=0.02810]
[Test2=0.02841]
[Test3=0.02902]
```

```
[Test4=0.02871]
[Test5=0.02814]
[Test6=0.02851]
[Test7=0.02845]
[Test8=0.02843]
[Test9=0.02826]
ORT (small)
[Test0=0.01669]
[Test1=0.01045]
[Test2=0.01055]
[Test3=0.01123]
[Test4=0.01037]
[Test5=0.01045]
[Test6=0.01027]
[Test7=0.01017]
[Test8=0.01038]
[Test9=0.01018]
Y758CC56
```

Для CPU все остальные тесты не имеют смысла. На самом деле ORT (ONNX Runtime) работает еще более медленнее, т.к. в TensorRT используется более «тяжелая» модель детектирования номеров.

6.3. Тест производительности одного изображения с внутренним распараллеливанием -p1pc

Не поддерживается в версии 2.6

6.4. Тест производительности одного изображения с разными объектами -p1o

Распознавание производится не только на GPU, но и частично на CPU. Т.е. это классическая схема – распараллеливание на GPU, потом один поток, затем опять распараллеливание на GPU, затем один поток. С учетом этого и того, что доступ к GPU асинхронный, то оптимальным использованием библиотеки будет создание нескольких объектов в памяти, пример функции `performanceTest1ImageP`:

```
int error;
iANPR2Object ia2[4];
const int objects = 4;
for (int i = 0; i < objects; i++)
{
    ia2[i] = iANPR2Init((char*)cfgname, license, &error);

    if (ia2[i] == NULL || error != int(IANPR2::iANPR2Errors::IA_OK))
    {
        std::cout << "Cann't init iANPR2Object! Error = " << error << " \n";
        return;
    }
}
```

Далее можно создать 4 программных потока, которые будут наполняться изображениями. В примере тестируется одно изображение и используется async для распараллеливания:

```
std::vector <cv::Mat> imgs;
imgs.push_back(cv::imread("image2.jpg"));
for (int i = 0; i < countTests; i++)
{
    auto t1 = std::chrono::high_resolution_clock::now();
    for (int j = 0; j < countRepeat; j++)
    {

        std::future<int> a[objects];
        for (int k = 0; k < objects; k++)
            a[k] = std::async(std::launch::async, anpr2PLate, ia2[k], imgs);

        for (int k = 0; k < objects; k++)
            a[k].wait();

        for (int k = 0; k < objects; k++)
            if (a[k].get() != int(IANPR2::iANPR2Errors::IA_OK))
                std::cout << "Inference problem! Error = " << a[k].get() << " \n";
    }
}
```

Пример теста на Windows 10 конфигурация i7-6700K 4.0ГГц RTX 2080 Super:

```
D:\ianpr2\ianpr2CPP\bin>ianpr2test.exe -p1o ianpr2dataconfig.json
TRT
[run=0.00579]
[run=0.00495]
[run=0.00495]
[run=0.00498]
[run=0.00500]
[run=0.00496]
[run=0.00503]
[run=0.00499]
[run=0.00509]
[run=0.00505]
ORT (small)
[run=0.01244]
[run=0.00580]
[run=0.00578]
[run=0.00572]
[run=0.00572]
[run=0.00580]
[run=0.00574]
[run=0.00573]
[run=0.00572]
[run=0.00572]
Y758CC56
```

На самом деле ORT (ONNX Runtime) работает еще более медленнее, т.к. в TensorRT используется более «тяжелая» модель детектирования номеров.

6.5. Тест производительности 4-х изображений –p4

Данный тест запускает несколько раз распознавание изображения image2.jpg и высчитывает на каждом тесте среднее время. Пример теста на Windows 10 конфигурация i7-6700K 4.0ГГц RTX 2080 Super:

```
ianpr2test.exe -p4 ianpr2dataconfigb4.json
[Test0=0.02781;0.006954]
[Test1=0.02710;0.006775]
[Test2=0.02827;0.007068]
[Test3=0.02927;0.007317]
[Test4=0.02987;0.007468]
[Test5=0.02918;0.007295]
[Test6=0.02953;0.007381]
[Test7=0.02973;0.007432]
[Test8=0.02980;0.007450]
[Test9=0.02966;0.007414]
Y758CC56
```

6.6. Тест производительности 4-х изображений с 4-мя объектами --p4o

Тестируем одновременное batchSize = 4 и 4 параллельных объекта. Пример теста на Windows 10 конфигурация i7-6700K 4.0ГГц RTX 2080 Super:

```
ianpr2test.exe -p4o ianpr2dataconfigb4.json
[Test0=0.06015;0.003759]
[Test1=0.05703;0.003565]
[Test2=0.05759;0.003599]
[Test3=0.05682;0.003551]
[Test4=0.05719;0.003575]
[Test5=0.05724;0.003578]
[Test6=0.05721;0.003576]
[Test7=0.05728;0.003580]
[Test8=0.05702;0.003564]
[Test9=0.05765;0.003603]
Y758CC56
```

6.7. Тест видео -v

Данные тест, реализованный в функции videoTest(), показывает, как работать с запоминанием номеров. Собственно дополнительно требуется очень мало. Заполняется структура:

```
iANPR2Setting settings{ 0.25, 12, 60,{"ru"},parallelDll,4,2,6,15,true };
```

И в процессе распознавания вызывается не только anpr2GetResult(), но и anpr2GetResultMem() для чтения результатов из памяти:

```
vector<vector<NumberResultMem>> mem = anpr2GetResultMem(ia2);
```

Видео выдается в окно OpenCV на экран и записывается в файл. Пример вызова:

```
ianpr2test.exe -v ianpr2dataconfig.json D:\data\auto\video.mp4
```

6.8. Тест видео с траекторией -vt

Данный пример в функции videoTestTraj показывает, как работать с траекторией. Для этого можно использовать файл pexels-george-morina-5222550 (2160p).mp4 из публичного dataseta:
https://www.kaggle.com/datasets/nimishshandilya/car-number-plate-video

Инициализация траектории при первом кадре:

```
traj = iANPR2Trajectory(image.rows, image.cols, 10000, 100, 1);
```

Добавление в объект траектории номеров и дополнительных объектов (в Python пока дополнительные объекты не стоит добавлять):

```
vector<vector<NumberResult>> numbers = anpr2GetResult(ia2);
vector<vector<ObjectResult>> objects = anpr2GetAddResult(ia2);
vector<NumberResult> tn;
vector<ObjectResult> on;
if (!numbers.empty())
    tn = numbers[0];
if (!objects.empty())
    on = objects[0];

// Добавить в траекторию
iANPR2TrajectoryProcess(traj, tn, on);
```

Получение траектории:

```
vector<TrajectoryResult> tr = iANPR2TrajectoryGetResult(traj);
```

Вывод траектории на изображение:

```
    int i = 0;
    for (auto t : tr)
    {
        string s = t.strNumber + ":" + t.templateNumber + ":" + to_string(t.weights) + "|ID:" +
to_string(t.ID);
        cv::Point p = cv::Point(0, 30 * (i + 1));
        int baseline = 0;
        cv::Size textSize = cv::getTextSize(s, cv::FONT_HERSHEY_SIMPLEX, 1, 1, &baseline);
        cv::rectangle(image, Rect(0, 30 * i, textSize.width, 30), cv::Scalar(255, 255, 255), cv::FILLED);
        cv::putText(image, s, p, cv::FONT_HERSHEY_SIMPLEX, 1, cv::Scalar(0, 0, 0), 2);
        // Рисование самой траектории
        if (!t.points.empty())
        {
            cv::circle(image, t.points[0].point, 5, cv::Scalar(0, 0, 255), 3);
            for (size_t j = 1; j < t.points.size(); j++)
            {
                cv::line(image, t.points[j - 1].point, t.points[j].point, cv::Scalar(0, 0, 255), 3);
                cv::circle(image, t.points[j].point, 5, cv::Scalar(0, 0, 255), 3);
            }
        }
        i++;
    }
```

6.9. Тест директории -d

Предназначен для тестирования результатов распознавания (в данном случае визуального). Указывается путь до папки с изображениями и путь для выходных изображений:

```
ianpr2test.exe -d ianpr2dataconfig.json D:\ianpr2\images\rus\test D:\ianpr2\images\rus\out
```

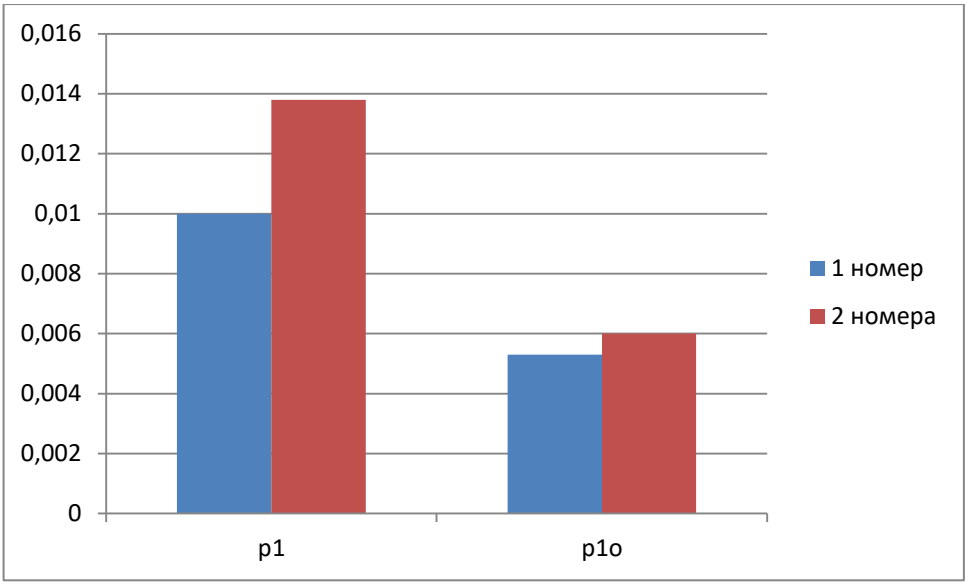
6.10. Результаты тестов GPU, CPU и ARM

Тесты с micro моделями ARM в расчете на один кадр:
- Orange PI 3 LTS – 0.655c
- Orange PI 5 – 0.21c

Тесты с nano моделями ARM в расчете на один кадр:
- Orange PI 3 LTS – 0.48c
- Orange PI 5 – 0.17c

Тесты CPU i7-6700K 4.0ГГц в расчете на один кадр:
- detection – 0.058c
- detectionsmall – 0.029c
- detectionmicro – 0.021c
- detectionnano – 0.015c

Указанные тесты получены при одном номере в кадре. Если в кадре несколько номеров, то для них будет тратиться дополнительное время на распознавание. Для примера зависимости можно посмотреть результаты тестирования TRT версии на NVIDIA RTX 2080 Super:



p1 – режим тестирования с 1 объектом распознавания, p1o – с 4-мя объектами распознавания. Видно, что при 2-х номерах в кадре время распознавания увеличивается в 1.5 раза.

7. Примеры Python

Примеры Python предоставлены в первую очередь для демонстрации того, как подключить библиотеку iANPR2 к своему проекту.

ianpr2class.py

В этом файле представлен класс iANPR2, в котором показано, как используясь ctypes подключить все функции из библиотеки. Функции:

```
# bool iANPR2Version(char* outBuf, int sizeBuf);
self.iANPR2Version = self.lib.iANPR2Version
self.iANPR2Version.restype = ctypes.c_bool
self.iANPR2Version.argtypes = (ctypes.c_char_p,ctypes.c_int,)
# void* iANPR2Init(char* configFile, char*license, int* error);
self.iANPR2Init = self.lib.iANPR2Init
self.iANPR2Init.restype = ctypes.c_void_p
self.iANPR2Init.argtypes = (ctypes.c_char_p,ctypes.c_char_p,ctypes.POINTER(ctypes.c_int),)
# void iANPR2Release(iANPR2Object* object);
self.iANPR2ReleaseP = self.lib.iANPR2ReleaseP
self.iANPR2ReleaseP.restype = ctypes.c_bool
self.iANPR2ReleaseP.argtypes = (ctypes.c_void_p,)
# bool iANPR2SettingsJSON(iANPR2Object object, char* json);
self.iANPR2SettingsJSON = self.lib.iANPR2SettingsJSON
self.iANPR2SettingsJSON.restype = ctypes.c_bool
self.iANPR2SettingsJSON.argtypes = (ctypes.c_void_p,ctypes.c_char_p,)
# int anpr2AddImage(iANPR2Object object, const char* image_bytes, long size_image);
self.anpr2AddImage = self.lib.anpr2AddImage
self.anpr2AddImage.restype = ctypes.c_int
self.anpr2AddImage.argtypes = (ctypes.c_void_p,ctypes.c_char_p,ctypes.c_long,)
# int anpr2inference(iANPR2Object object);
self.anpr2inference = self.lib.anpr2inference
self.anpr2inference.restype = ctypes.c_int
self.anpr2inference.argtypes = (ctypes.c_void_p,)
# int anpr2GetResultJSON(iANPR2Object object, char* outBuf, int sizeBuf);
self.anpr2GetResultJSON = self.lib.anpr2GetResultJSON
self.anpr2GetResultJSON.restype = ctypes.c_int
self.anpr2GetResultJSON.argtypes = (ctypes.c_void_p,ctypes.c_char_p,ctypes.c_int,)
# int anpr2GetAddResultJSON(iANPR2Object object, char* outBuf, int sizeBuf);
self.anpr2GetAddResultJSON = self.lib.anpr2GetAddResultJSON
self.anpr2GetAddResultJSON.restype = ctypes.c_int
self.anpr2GetAddResultJSON.argtypes = (ctypes.c_void_p,ctypes.c_char_p,ctypes.c_int,)
#int anpr2GetResultMemJSON(iANPR2Object object, char* outBuf, int sizeBuf);
self.anpr2GetResultMemJSON = self.lib.anpr2GetResultMemJSON
self.anpr2GetResultMemJSON.restype = ctypes.c_int
self.anpr2GetResultMemJSON.argtypes = (ctypes.c_void_p,ctypes.c_char_p,ctypes.c_int,)
#void* iANPR2Trajectory(int widthImage, int heightImage, int tLimitTime, int tLimitFrames, int typeSpeed);
self.iANPR2Trajectory = self.lib.iANPR2Trajectory
self.iANPR2Trajectory.restype = ctypes.c_void_p
self.iANPR2Trajectory.argtypes = (ctypes.c_int,ctypes.c_int,ctypes.c_int,ctypes.c_int,)
#bool iANPR2TrajectoryRelease(void* object);
self.iANPR2TrajectoryRelease = self.lib.iANPR2TrajectoryRelease
self.iANPR2TrajectoryRelease.restype = ctypes.c_bool
self.iANPR2TrajectoryRelease.argtypes = (ctypes.c_void_p,)
#bool iANPR2TrajectoryProcessJSON(void* object, char* jsonNumbers, char* jsonObjects);
self.iANPR2TrajectoryProcessJSON = self.lib.iANPR2TrajectoryProcessJSON
self.iANPR2TrajectoryProcessJSON.restype = ctypes.c_bool
self.iANPR2TrajectoryProcessJSON.argtypes = (ctypes.c_void_p,ctypes.c_char_p,ctypes.c_char_p,)
#int iANPR2TrajectoryGetResultJSON(void* object, char* outBuf, int sizeBuf);
self.iANPR2TrajectoryGetResultJSON = self.lib.iANPR2TrajectoryGetResultJSON
self.iANPR2TrajectoryGetResultJSON.restype = ctypes.c_int
self.iANPR2TrajectoryGetResultJSON.argtypes = (ctypes.c_void_p,ctypes.c_char_p,ctypes.c_int,)
#int iANPR2CreateLineIntersection(void* object,int line1X1, int line1Y1, int line1X2, int line1Y2,
#int line2X1, int line2Y1, int line2X2, int line2Y2);
self.iANPR2CreateLineIntersection = self.lib.iANPR2CreateLineIntersection
self.iANPR2CreateLineIntersection.restype = ctypes.c_int
self.iANPR2CreateLineIntersection.argtypes = (ctypes.c_void_p,ctypes.c_int,ctypes.c_int, ctypes.c_int, \
ctypes.c_int,ctypes.c_int,ctypes.c_int,ctypes.c_int,ctypes.c_int,)
```

```
#int iANPR2GetLineIntersection(void* object, int indexTr);
self.iANPR2GetLineIntersection = self.lib.iANPR2GetLineIntersection
self.iANPR2GetLineIntersection.restype = ctypes.c_int
self.iANPR2GetLineIntersection.argtypes = (ctypes.c_void_p,ctypes.c_int,)
```

Для упрощения доступа к этим функциям написаны обертки.

ianpr2test.py (Windows) и ianpr2testL.py (Linux)

В примере реализованы 4 режима – распознавание 1-го изображения, распознавание 4-х изображений, распознавание видео и распознавание видео с траекторией. Пример последовательности действий при распознавании изображений рассмотрен ниже.

Инициализация класса:

```
import ianpr2class as ia2

ianpr2 = ia2.iANPR2('D:\\ianpr2\\ianpr2CPP\\bin\\ianpr2.dll')
```

Установка настроек в формате словаря, пример показан тут.

Загрузка файла лицензии:

```
lic = ""
with open("lic2.key", "r") as f:
    for line in f:
        lic = line.strip()
        break
```

1) Инициализация объекта:

```
ianpr2.initiANPR2Object(config,lic)
```

В классе iANPR2, доступ к Си функции реализован так:

```
    def initiANPR2Object(self,dataConfig,lic) -> bool:
        er = ctypes.c_int()
        self.ianpr2 = self.iANPR2Init(bytes(dataConfig,'ascii'),bytes(lic,'ascii'),ctypes.byref(er))
        if self.ianpr2 != None:
            logging.info("Object iANPR2 loaded")
            return True
        logging.error("Object iANPR2 not loaded: "+ str(er))
        return False
```

2) Загрузка в объект настроек:

```
ianpr2.setSettings(settings)
```

В классе iANPR2, доступ к Си функции реализован так:

```
    def setSettings(self, settings) -> bool:
        s = json.dumps(settings)
        if self.iANPR2SettingsJSON(self.ianpr2,bytes(s,'ascii')):
            logging.info("iANPR2Settings installed")
            return True
        logging.error("iANPR2Settings not loaded")
        return False
```

3) Передача изображения:

```
ianpr2.putImage(img)
```

В классе iANPR2, доступ к Си функции реализован так:

```
    def putImage(self, img) -> bool:
        img_str = cv2.imencode('.bmp', img)[1].tostring()
        res = self.anpr2AddImage(self.ianpr2,img_str,len(img_str))
        if res == 0:
            logging.info("Image added")
            return True
        logging.error("Image not added. Error = "+str(res))
        return False
```

4) Вызов распознавания:

```
ianpr2.inference()
```

В классе iANPR2, доступ к Си функции реализован так:

```
    def inference(self) -> bool:
        res = self.anpr2inference(self.ianpr2)
        if res == 0:
            logging.info("Inference true")
            return True
        logging.error("Inference error = "+str(res))
        return False
```

5) Получение результатов:

```
s = ianpr2.getResult()
```

В классе iANPR2, доступ к Си функции реализован так:

```
    def getResult(self) -> str:
        size = 1000
        p = ctypes.create_string_buffer(size)
        if self.anpr2GetResultJSON(self.ianpr2,p,size) != 0:
            return ""

        return p.value.decode("ascii")
```

После этого можно спарсить JSON и получить результаты, как показано в примере.

8. Inference-сервер

Вы можете реализовать работу с iANPR2 посредством inference-сервера, код которого представлен в папке iANPR2InferenceServer, а также в виде исполняемых файлов:

- iANPR2Server – запускаемый файл сервера в виде консольного приложения;
- iANPR2Client – тестовый клиент на языке C++;
- ianprclient.py – тестовый клиент на Python.

Для старта нужно запустить исполняемый файл iANPR2Server (.exe в Windows) с параметром, например так:

iANPR2Server.exe configinserv.json

Где в качестве параметра передается настроечный файл в JSON формате:

```
{
	"name":"Config for iANPR2 inference server (IIS)",
	"versionIIS":1,
	"maxImagesInQueue":10,
	"gpuObjects":5,
	"pathToConfigForLibrary":"D:/ianpr2/ianpr2CPP/bin/ianpr2dataconfig.json",
	"detectConfThresh":0.25,
	"minPlateHeight":12,
	"minPlateWidth":60,
	"plateTemplates":["ru"],
	"minSymbolsPlate":6,
	"maxSymbolsPlate":15,
	"port":1234,
	"mode":1,
	"expandedResult":0
}
```

Параметры:
name – информационный;
versionIIS – версия формата файла;
maxImagesInQueue – максимальное количество изображений в очереди (если очередь переполнена, то клиенту будет сообщаться об ошибке);
gpuObjects – количество объектов, параллельно обрабатывающих запросы, для версии iANPR2 GPU;
pathToConfigForLibrary – путь до конфигурационного файла для iANPR2;
detectConfThresh – см. структуру iANPR2Setting;
minPlateHeight – см. структуру iANPR2Setting;
minPlateWidth – см. структуру iANPR2Setting;
plateTemplates – перечисление шаблонов номеров, без шаблонов нужно указать [];
minSymbolsPlate – см. структуру iANPR2Setting;
maxSymbolsPlate – см. структуру iANPR2Setting;
port – номер порта, который будет прослушиваться;
mode – режим запуска: 0 – без лога в консоль, 1 – с логом в консоль;

expandedResult – расширенный результат (0 или 1) см. структуру iANPR2Setting.

8.1 REST API

CheckRun

Проверить, запущен ли сервер

http://localhost/CheckRun

Responses

Имя	Описание
200	Всегда возвращает OK при доступности сервиса

ImageInference

Распознать изображения

http://localhost/ImageInference

Request Body

Имя	Обязательный	Тип	Описание
Image*	True	`"application/octet-stream"`	Бинарные данные изображения в формате BMP, JPEG, PNG или TIFF.

* Передается как multipart data

Responses

Имя	Описание
200	JSON строка ImageInferenceResult

ImageInferenceResult:

Имя	Тип	Описание
status	integer	Результат распознавания: 0 – успешно 1000 – в запросе не было параметра image 1001 – очередь изображений переполнена 1002 – внутренняя ошибка Иное – ошибка от iANPR2 библиотеки (см. код там)
result	dictionary	
timePeriod	unsigned long	Время операции в миллисекундах

Пример успешного результата:

{"status":0,"result":{"images":[[{"x":234,"y":286,"width":196,"height":52,"strings":["Y758CC56"],"templates":["base_1"],"confidence":[0.928555],"score":0.908587}]]},"timePeriod":16}

StopServer

Остановить сервер

http://localhost/StopServer

Responses

Имя	Описание
200	Всегда возвращает OK при доступности сервиса

8.2. Примеры C++ и Python

Пример отправки запроса на C++ с использованием Httplib:

```
void runClient(int port)
{
    httplib::Client cli("localhost", port);
    ifstream in("image2.jpg", std::ios::binary);
    // Читаем всё в массив
    std::vector<char> buffer(std::istreambuf_iterator<char>(in), {});
    in.close();

    httplib::MultipartFormDataItems items = {
        { "image", std::string(buffer.data(), buffer.size()), "image",
"application/octet-stream" },
    };
    for (;;)
    {
        if (auto res = cli.Post("/ImageInference", items)) {
            cout << res->status << endl;
            cout << res->get_header_value("Content-Type") << endl;
            cout << res->body << endl;
        }
        else {
            cout << "error code: " << res.error() << std::endl;
        }
    }
}
```

Пример отправки запроса на Python:

```
import requests
from requests_toolbelt import MultipartEncoder
import cv2
import time

adress = 'localhost'
port = 1234

img = cv2.imread("image2.jpg")
img_str = cv2.imencode('.jpg', img)[1].tobytes()
sess = requests.Session()

while True:
       t1 = time.time()
       multipart_data = MultipartEncoder(
       fields={
              'image': ('image', img_str, 'application/octet-stream')
       }
```

```
)
        x1 = sess.post(
                "http://{}:{}/ImageInference".format(adress,port),
                data = multipart_data,
                headers={'Content-Type': multipart_data.content_type}
                )
        print(x1.text)
```

9. Получение движков TensorRT из ONNX моделей

Для получения движков TensorRT нужно использовать утилиту, идущую в составе TensorRT (Windows или Linux) – trtexec. В Windows она находится в каталоге bin распакованного пакета. В Ubuntu, после установки пакета, ее можно найти так:

```
locate trtexec
```

Но обычно он располагается по адресу:
/usr/src/tensorrt/bin/trtexec

В составе iANPR 2 две модели – детектирования номеров и классификации. Модель детектирования – это YOLOX модель, которая поставляется в двух ONNX файлах:
detection300524.onnx – модель с batchSize = 1
detection300524_b4.onnx – модель с batchSize = 4

Команды для получения TensorRT моделей детектирования номеров такие:

```
trtexec --onnx=detection300524.onnx --fp16 --saveEngine=detection300524.engine
trtexec --onnx=detection300524_b4.onnx  --fp16 --saveEngine=detection300524_b4.engine
```

Команды для получения TensorRT модели детектирования символов:

```
trtexec --onnx=symbolsmodelD.onnx --fp16 --saveEngine=symbolsmodel05032024.engine --
minShapes=images:1x3x320x320 --optShapes=images:4x3x320x320 --maxShapes=images:8x3x320x320
```

Модель классификации одна в файле model.onnx. Команда:

```
trtexec --onnx=classification.onnx  --fp16 --saveEngine=classification100225.engine --precisionConstraints=prefer
--minShapes=input_1:1x28x28x1 --optShapes=input_1:8x28x28x1 --maxShapes=input_1:32x28x28x1
```

Имена движков необязательны, но если вы используете другие, тогда вам надо менять конфигурационные файлы ianpr2dataconfig.json и ianpr2dataconfigb4.json.

Для небольшого ускорения движков вы можете использовать ключ --useSpinWait, но это может привести к большей нагрузке на CPU.

10. Рекомендации к использованию

Основные рекомендации

Основные рекомендации для версии TRT (TensorRT) и ORT (ONNX Runtime):

1. Инициализируйте объект iANPR2Object один раз, поскольку время на инициализацию занимает несколько секунд.
2. По возможности используйте несколько параллельных объектов распознавания, это требует больше GPU памяти, но за счет асинхронности скорость может повышаться больше, чем в 2 раза в зависимости от оборудования. При этом памяти тратится мало, так что спокойно можно использовать 4 и более параллельных объектов.
3. Если вам нужен максимум скорости, то помимо параллельных объектов используйте пакетное распознавание, для которого требуется специальный движок TensorRT.
4. Для повышения качества распознавания используйте объединение результатов с нескольких кадров. Это можно сделать или с помощью функции внутренней памяти, траектории или самостоятельно.
5. Для отброса ложных срабатываний повысьте порог detectConfThresh и увеличьте минимальное значение ширины и высоты номера. Также можете увеличить значения memoryNumberRepeat для накапливаемого распознавания.
6. Используйте организационные возможности повышения качества распознавания – размещайте камеру так, чтобы не допускать наклона символов на изображении.

Основные рекомендации для версии CPU:

1. Использование параллельных объектов может сильно нагрузить процессор (до 100%) при небольшом увеличении быстродействия. При виртуальном CPU быстродействие даже снизится.

Основные требования к алгоритму распознавания:

- Номер автомобиля должен размещаться в кадре целиком.
- Угол вертикального наклона видеокамеры не более 40°.
- Угол наклона вглубь – не более 30°.
- Изображения должны быть четкими и не размытыми.
- Размер символов для надежного распознавания должен быть не менее 14 пикселей в высоту.

При получении результатов для фильтрации номеров пользуйтесь как вероятностью номера при детектировании **score**, так и вероятностью символов номера **confidence**. Так в примере указанном ниже:

1. Ложный номер с низкой детектированной достоверностью 0.59 (в принципе допустима), однако низкой достоверностью символов 0.42 и не найденным шаблоном – можно отбросить.
2. Погнутый номер. Достоверность детектирования нормальна – 0.76, как и символов 0.81. Но шаблон не найден, так как не найдены символы погнутой части номера.

Пример хорошего распознавания:

Да **score** детектирования не сильно высок, но допустим. Но вот **confidence** хорош. Как и найден шаблон номера.

Рекомендации при использовании алфавитов с дополнительными символами (например, кириллицы)

Тут описаны виды возвращаемых строк номеров: описание возвращаемых результатов. При использовании C++ интерфейса можно использовать любые типы возвращаемых результатов. По умолчанию в структуре стоит значение iANPR2_RESULT_NUMBER_LATIN, что значит следующее: нелатинские символы будут возвращаться в виде знака «?».

Если нужно возвращать символы нелатинские (номера полиции республики Беларусь), то лучше использовать тип iANPR2_RESULT_NUMBER_1251, т.к. он не увеличивает длину строки. И это лучший вариант для передачи в функцию траектории iANPR2TrajectoryProcess. В режиме iANPR2_RESULT_NUMBER_UTF8 некириллические символы занимают 2 байта длины, а в режиме iANPR2_RESULT_NUMBER_FULL – 5 байт длины (включая символ &).

При применении не C++ доступ через JSON нужно помнить, что библиотека декодирует JSON только с символами UTF-8. Т.е. при передаче в функцию iANPR2TrajectoryProcessJSON, нельзя использовать тип iANPR2_RESULT_NUMBER_1251 – он выдаст ошибку декодирования. Остальные типы можно использовать.

11. Результаты тестирования

Мы провели тестирование и сравнили результаты распознавания с предыдущей версией.

Наши выборки для тестирования:

- RUS1 – выборка для распознавания номеров России, которая изначально использовалась для тестирования iANPR1;
- KAZ1 – выборка для распознавания номеров Казахстана, которая изначально использовалась для тестирования iANPR1 – однострочные номера;
- RUS2 – вторая выборка для распознавания номеров России (ее не было при разработке iANPR1);
- KG – выборка для распознавания номеров Кыргызстана;
- KAZ2 – выборка для распознавания номеров Казахстана, в том числе содержащая двухстрочные номера;
- BY – выборка для распознавания номеров республики Беларусь;
- AM – выборка для распознавания номеров Армении;
- MD – выборка для распознавания номеров Молдовы;
- AZ – выборка для распознавания номеров Азербайджана;
- TJ – выборка для распознавания номеров Таджикистана;
- UZ – выборка для распознавания номеров Узбекистана;
- GR – выборка для распознавания номеров Грузии.

В таблице ниже представлены результаты тестирования. Достоверность результатов вычисляется по следующему показателю:

$$D = \frac{T}{A},$$

где T – правильно распознанные номера (строка номера полностью совпадает), A – все номера.

Достоверность распознавания не точно соответствует качеству для конкретной страны, так как выборки отличаются друг от друга, и одни могут быть сложнее других, а именно – серьезный поворот и наклон номера, большое изображение и маленький номер, что затрудняет детектирование.

Small, micro и nano в тестировании CPU показывает используемую модель детектирования номера. К примеру, в выборке KAZ1 при уменьшении модели до small качество немного выросло, что говорит, что на изображениях меньше маленьких номеров, чем, к примеру, в KAZ2. Чем меньше модель – тем хуже распознаются маленькие номера просто из-за изменения размера исходного изображения.

При этом результаты ложных срабатываний – обнаружение номера там, где его не было – не учитывались. Но в версии iANPR 2 они существенно ниже, чем в iANPR 1.8.

Для версии ONNX Runtime GPU также проводились тестирования, и, не смотря на те же модели, что и в CPU версии, результаты были немного другими. Однако, поскольку общий порядок результатов остался прежним, то они не приведены.

Для TensorRT выше достоверность в ряде случаев по причине использования более «тяжелой» модели детектирования номеров.

Таблица. Достоверность распознанных номеров

	iANPR 1.8	iANPR 2.6 TensorRT	iANPR 2.6 CPU	iANPR 2.6 CPU (small)	iANPR 2.6 CPU (micro)	iANPR 2.6 CPU (nano)
RUS1	0.614	0.944	0.939	0.932	0.913	0.895
KAZ1	0.479	0.947	0.944	0.944	0.941	0.890
RUS2	0.547	0.952	0.942	0.936	0.929	0.896
KG	-	0.881[4]	0.856[4]	0.818[4]	0.790[4]	0.727[4]
KAZ2	-	0.901	0.894	0.866	0.839	0.778
BY	-	0.890[2]	0.857[2]	0.855[2]	0.836[2]	0.779[2]
AM	-	0.936[3]	0.921[3]	0.910[3]	0.889[3]	0.860[3]
MD	-	0.931[1]	0.928[1]	0.901[1]	0.874[1]	0.834[1]
AZ	-	0.919	0.908	0.868	0.796	0.775
TJ	-	0.862[4]	0.851[4]	0.796[4]	0.750[4]	0.652[4]
UZ	-	0.872[4]	0.844[4]	0.798[4]	0.725[4]	0.662[4]
GR	-	0.918[1]	0.910[1]	0.905[1]	0.894[1]	0.876[1]

[1] - результаты тестирования не учитывали различия 0 (ноль) и O (буква);
[2] – в выборке были символы кириллицы, которые не поддерживаются;
[3] – в выборке нет символов нелатинского алфавита;
[4] – у KG, TJ, UZ меньшая достоверность из-за более сложной выборки и низкого качества номеров, и особенностей номеров, в частности символов региона, которые в то время, как основные символы достаточны для прочтения, очень маленькие и сливаются.

12. Возможные ошибки

Внутренние возвращаемые ошибки описаны в iANPR2Errors.h.

IA_OBJECTNULL = -1,

iANPR2 объект пустой.

IA_OK = 0,

Действие завершено успешно.

IA_TRTENGINELOADERROR = 1,

Невозможно загрузить один из движков TensorRT, указанных в конфигурации.

IA_COPYTOGPUMEMERROR = 2,

Ошибка копирования в GPU память.

IA_INFERENCEFAIL = 3,

Ошибка распознавания (инференс) нейросети.

IA_GETFROMGPUERROR = 4,

Ошибка получения результатов с GPU на CPU.

IA_CONFIGNOTFOUNDORDAMAGED = 5,

Не найден конфигурационный файл или содержит ошибки.

IA_NOIMAGES = 6,

Переданные Mat на распознавание – пустые.

IA_IMAGENOT8U = 7,

Изображения неправильного формата – не 3-х канальные 8 битные.

IA_TEMPLATELOADFALSE = 8,

Ошибка при загрузке файла шаблонов номеров.

IA_LICENSENULL = 9,

Буфер с лицензией пуст.

IA_LICENSEERROR = 10,

Невозможно расшифровать лицензию.

IA_ADDIMGCANNTDECODE = 11,

Невозможно декодировать массив байт в изображение.

IA_ADDIMGBUFFERFULL = 12,

Невозможно добавить изображение в буфер – он полон.

IA_ADDIMGNOTEQUALBATCH = 13,

Количество изображений в буфере не соответствует batchSize.

IA_OUTJSONBUFFERSMALL = 14

Выходной буфер для JSON слишком мал.

IA_OPENVINOLOADERROR = 15

OpenVINO не смогло загрузить одну из моделей.

IA_NEEDONLYBATCHSIZE1 = 16

В данном режиме поддерживается только batchSize == 1.

IA_WRONGTYPEEMERSERV = 17

Не используется в этой версии.

IA_TRAJECTORYVERTICALLINES = 18

Линии пересечения больше вертикальны, чем горизонтальны.

IA_TRAJECTORYNOTPARALLELLINES = 19

Линии пересечения не параллельны.

IA_MEMINDEXWRONGVALUE = 20

Ошибка в функции setCurrentMemIndex – неправильный индекс сохранения в памяти.

IA_ONNXRUNTIMELOADERROR = 21

ONNX Runtime не смог загрузить модели в память.

Внешние возможные ошибки описаны ниже:

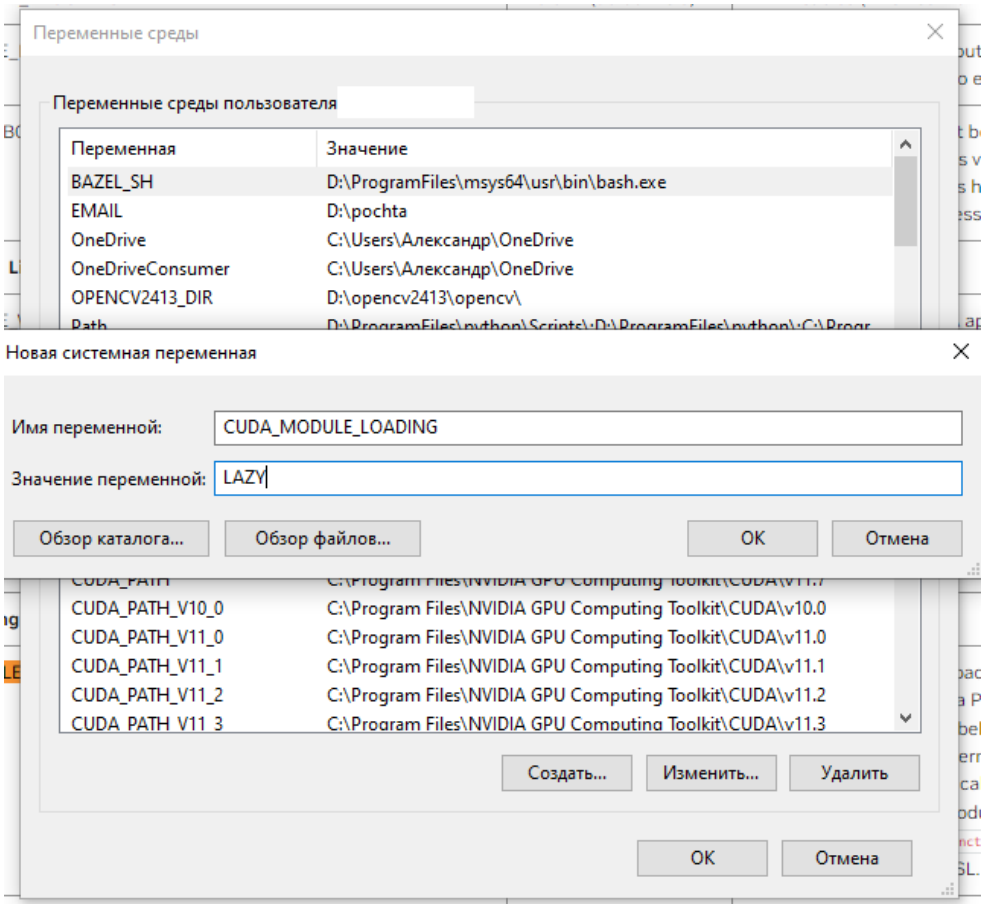
1. cudaErrorInsufficientDriver = 35

Если в процессе тестирования утилиты в консоли отображается ошибка 35, то ее инициирует CUDA. Она означает, что графический драйвер старее, чем эта версия TensorRT и нуждается в обновлении. Подробнее об ошибках, инициируемых CUDA можно почитать тут.

2. Сообщение от CUDA (Не влияет на работу):

CUDA lazy loading is not enabled. Enabling it can significantly reduce device memory usage. See `CUDA_MODULE_LOADING` in https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#env-vars

Определяется переменной окружения CUDA_MODULE_LOADING. Указывает режим загрузки модуля для приложения. Если установлено значение EAGER, все ядра из файла cubin, fatbin или PTX полностью загружаются при соответствующем вызове API cuModuleLoad*. Это то же поведение, что и во всех предыдущих выпусках CUDA. Если установлено значение LAZY, загрузка определенного ядра задерживается до момента извлечения дескриптора CUfunc с помощью вызова API cuModuleGetFunction. Этот режим позволяет снизить начальную задержку загрузки модуля и уменьшить начальное потребление памяти устройства, связанного с модулем, за счет более высокой задержки вызова API cuModuleGetFunction. Поведение по умолчанию — LAZY в Linux и EAGER в Windows и WSL. Поведение по умолчанию может измениться в будущих выпусках CUDA. В версиях CUDA ниже 11.6 это сообщение может выдаваться в любом случае. В Windows Можно удалить, установив переменную окружения:



3. Сообщение TensorRT

Using an engine plan file across different models of devices is not recommended and is likely to affect performance or even cause errors.

Файл был создан для видеокарты другого типа.

4. CUDA_ERROR_ILLEGAL_ADDRESS

Такая ошибка может свидетельствовать о попытке использовать движок TensorRT созданный для Linux под Windows и наоборот. К сожалению, текущая версия TensorRT чувствительна к операционной системе платформы.

5. Ошибка OpenVINO

OSError: libopenvino.so.2330: cannot open shared object file: No such file or directory

Настройки окружения Linux сбросились, возможно после перезагрузки. Надо установить заново с помощью:

source /opt/intel/openvino_2024/setupvars.sh

Желательно прописать в автозагрузку.