

iANPR SDK

Version 1.8
Documentation

CONTENTS

Introduction

1. Modules

1.1. Recognition module – iANPR

anprPlate

Structure ANPR_OPTIONS

Types of recognizable numbers

Type ANPR_CUSTOM_TYPE

anprPlateRect

LicenseValue

1.2. Interface module – iANPRinterface

1.3. Stream module – iANPRcapture

2. Installation and use

2.1. Windows

2.2. Linux

3. Examples on C/C++ for Windows

3.1. Image

3.2. Image_new

3.3. Image_omp

3.4. Capture

3.5. Capture (iANPRcapture)

3.6. (iANPRcapture_motion)

3.7 Persptrans

4. Examples in other programming languages for Windows

4.1. C#

4.1.1. Example iANPRcapture_motion in C# for iANPR SDK

4.2. Delphi

5. Recommendations for use

6. How to use the demo version of the iANPR SDK

Conclusion

Introduction

iANPR SDK is a set of development tools for car plate number recognition. The main goal is to provide automated recognition of car numbers based on the OpenCV computer vision library. The library features include image processing. The main language of the library is C/C++.

Version 1.8 compiled with versions of OpenCV 3.4.0, if necessary (at the request of the Client for FULL versions) can be compiled for a different version.

Types of licenses

iANPR FREE This type of license is intended for free use of a library with limited recognition capabilities: the speed of work is artificially significantly slowed down. This type of license can only be used for educational and / or academic purposes. It is not allowed to distribute the software together with this library.

iANPR RUS PRO LIMITED This is a paid license, which provides all the capabilities for recognizing standard and transit Russian plate numbers. It is allowed to use only for own needs within the organization (or only by an individual) that has accepted this license.

iANPR RUS PRO EXTENDED LIMITED This is a paid license, which provides all the capabilities for recognizing standard and transit Russian plate numbers, as well as other types of numbers present in this version. It is allowed to use only for own needs within the organization (or only by an individual) that has accepted this license.

iANPR RUS PRO FULL This is a paid license, which provides all the capabilities to recognize all types of vehicle plate numbers in the Russian Federation that are present in this version. You can use this license for distribution in the composition with your own software product.

iANPR KAZ PRO LIMITED This is a paid license that provides all the capabilities for recognizing private and corporate numbers of Kazakhstan (only rectangular one-line). It is allowed to use only for own needs within the organization (or only by an individual) that has accepted this license.

iANPR KAZ PRO FULL This is a paid license, which provides all the opportunities for recognizing private and corporate numbers of Kazakhstan (only rectangular one-line). You can use this license for distribution in the composition with your own software product.

iANPR TM PRO LIMITED This is a paid license, which provides all the opportunities for recognizing private and business numbers of Turkmenistan (only rectangular one-line ones). It is allowed to use only for own needs within the organization (or only by an individual) that has accepted this license.

iANPR TM PRO FULL This is a paid license, which provides all the opportunities for recognizing private and business numbers of Turkmenistan (only rectangular one-line ones). You can use this license for distribution in the composition with your own software product.

iANPR BY PRO LIMITED This is a paid license, which provides all the possibilities for recognizing the numbers of Belarus. It is allowed to use only for own needs within the organization that has accepted this license.

iANPR BY PRO FULL This is a paid license, which provides all the possibilities for recognizing the numbers of Belarus. You can use this license for distribution in the composition with your own software product.

iANPR PL PRO LIMITED This is a paid license, which provides all the possibilities for recognizing the numbers of Poland. It is allowed to use only for own needs within the organization that has accepted this license.

iANPR PL PRO FULL This is a paid license, which provides all the possibilities for recognizing the numbers of Poland. You can use this license for distribution in the composition with your own software product.

iANPR LV PRO LIMITED This is a paid license, which provides all the possibilities for recognizing the numbers of Latvia. It is allowed to use only for own needs within the organization that has accepted this license.

iANPR LV PRO FULL This is a paid license, which provides all the possibilities for recognizing the numbers of Latvia. You can use this license for distribution in the composition with your own software product.

iANPR LT PRO LIMITED This is a paid license, which provides all the possibilities for recognizing the numbers of Lithuania. It is allowed to use only for own needs within the organization that has accepted this license.

iANPR LT PRO FULL This is a paid license, which provides all the possibilities for recognizing the numbers of Lithuania. You can use this license for distribution in the composition with your own software product.

iANPR EST PRO LIMITED This is a paid license, which provides all the possibilities for recognizing the numbers of Estonia. It is allowed to use only for own needs within the organization that has accepted this license.

iANPR EST PRO FULL This is a paid license, which provides all the possibilities for recognizing the numbers of Estonia. You can use this license for distribution in the composition with your own software product.

iANPR UA PRO LIMITED This is a paid license, which provides all the possibilities for recognizing the numbers of Ukraine. It is allowed to use only for own needs within the organization that has accepted this license.

iANPR UA PRO FULL This is a paid license, which provides all the possibilities for recognizing the numbers of Ukraine. You can use this license for distribution in the composition with your own software product.

iANPR MD PRO LIMITED This is a paid license, which provides all the possibilities for recognizing the numbers of Moldova. It is allowed to use only for own needs within the organization that has accepted this license.

iANPR MD PRO FULL This is a paid license, which provides all the possibilities for recognizing the numbers of Moldova. You can use this license for distribution in the composition with your own software product.

iANPR PRO FULL ALL This is a paid license that provides all the capabilities for recognizing the numbers of the iANPR library for all countries currently supported. You can use this license for distribution in the composition with your own software product.

ARM – this designation characterizes a separate product covered by this license agreement, but other pricing rules.

It is possible to combine licenses to recognize the numbers of different countries. Terms of sharing are available on the website: <http://intbusoft.com/iANPR/>

1. Modules

In version 1.8, the library is divided into 3 modules:

- recognition module;
- interface module;
- streaming module.

In the recognition module, the basic functions for recognizing car numbers remained.

The interface module is designed to facilitate access to the recognition functions and provides various access options.

The stream module is designed to combine recognition results from several frames.

1.1. Recognition module – iANPR

This module (iANPR.h) implements the recognition of one image.

anprPlate

The function of searching for car plates on the OpenCV format image.

```
int anprPlate(  
    IplImage* Image,  
    ANPR_OPTIONS Options,  
    int* AllNumber, CvRect* Rects,  
    char** Texts,  
    void* param = NULL  
);
```

Parameters:

Image - input image in OpenCV format (8-bit 1-channel or 8-bit 3-channel depending on Options.type_number parameters);

Options - settings for the recognition mode in ANPR_OPTIONS structure format;

AllNumber - the number of the found numbers. Before function calling must contain **Texts** lines count;

Rects - a pointer to an array of CvRect structures (this is a structure from the OpenCV library), where the zones of car plates are located;

Texts - a pointer to an array of pointers of a character type, in which text will be returned for each car plate, pointers should point to previously allocated areas of memory;
param - not yet used.

The `anprPlate` function returns 0 if at least one car number is successfully found. 1 - no candidate for the car number was detected, 2 - no car numbers were found. In addition, one of the following errors defined in `iANPRError.h` can be returned:

`IMAGE_EMPTY` (-2) The image is empty;

`ERROR_TYPE_PLATE` (-100) Unsupported type for this configuration. For example, the license `iANPR RUS PRO LIMITED` does not support the `ANPR_RUSSIAN_PUBLIC` type flag. Therefore, its use will return an error.

`ERROR_TYPE_FOR_COLOR` (-101) The image type and the car number type flag in the [ANPR_OPTIONS](#) structure do not match.

Structure `ANPR_OPTIONS`

This structure determines the recognition modes.

```
struct ANPR_OPTIONS
{
    char sign1 = 'i', sign2 = 'a', sign3 = '1';    //
    service information, should not be modified

    int min_plate_size;    // Minimal area of car plate
    int max_plate_size;    // Maximum area of car plate
    int Detect_Mode;    // Detection modes
    int max_text_size;    // Maximum number of
characters of car plate + 1
    int type_number;    // Type of car plate
    int flags;    // Additional options
    void* custom;    // Filled only for type
ANPR_CUSTOM_TYPE, otherwise empty
    char* vers = "1.6";    // Version of iANPR SDK being
used. If not specified (vers = 0), then 1.5 is assumed.
    double alpha = 90.0;    // Rotate around the X axis
    double beta = 90.0;    // Rotate around the Y axis
    double gamma = 90.0;    // Rotate around the Z axis

    int max_threads = 1;    // Number of threads;
}
```

The minimum and maximum areas of car plates limit the search for candidates for car plates. The area of the car plate is determined by the product of the width of the car plate by its height. If it is necessary to specify the car plates of the Russian Federation through the width, the following recalculation can be used:

$$\text{min_plate_size} = \text{min_plate_width} * \text{min_plate_height};$$
$$\text{max_plate_size} = \text{max_plate_width} * \text{max_plate_height};$$

where min_plate_width is the minimum width of the car plate, max_plate_width is the maximum width of the car plate, min_plate_height is the minimum height of the car plate, max_plate_height is the maximum height of the car plate.

Detect_Mode determines the modes of detecting a car plate. You can even use them together. In this version there are 4 detection modes:

- ANPR_DETECTMODE1,
- ANPR_DETECTMODE2,
- ANPR_DETECTMODE3,
- ANPR_DETECTMODE4.

They differ in the settings when searching for car plates and they can be used together (although the performance be slightly reduced).

ANPR_DETECTMODE1 - A method based on the detection of a whole car plate with simple adaptive image processing.

ANPR_DETECTMODE2 - A method based on the detection of a whole car plate with adaptive image processing based on the removal of small jumpers. Includes almost 100% of the car plates detected using ANPR_DETECTMODE1, as well as car plates that ANPR_DETECTMODE1 are not detected. Therefore, ANPR_DETECTMODE1 is not recommended.

ANPR_DETECTMODE3 - A method based on the detection of the whole car plate with block image processing.

ANPR_DETECTMODE4 - A method based on the detection of parts of a car plate with simple adaptive image processing. It is not recommended to use it separately from other methods, because it gives low values of detected car plates and not always accurate detection. However, it detects those car plates that are not detected by other methods. *In this case, a significant car plates of additional false positives may appear, for example, on posters.*

For qualitative recognition, it is recommended to use combinations of methods ANPR_DETECTMODE2 and ANPR_DETECTMODE3, or ANPR_DETECTMODE2 + ANPR_DETECTMODE3 + ANPR_DETECTMODE4, the last combination of methods can be obtained by one flag ANPR_DETECTCOMPLEXMODE.

Definition in iANPR.h:

```
#define ANPR_DETECTMODE1 0x01
```



```
#define ANPR_DETECTMODE2          0x02
#define ANPR_DETECTMODE3          0x04
#define ANPR_DETECTMODE4          0x08
```

The maximum number of characters in a car plate must match the maximum size specified in the Texts of the anprPlate function. Of course, the maximum number of characters + the end-of-line character (0) is 10, but if you put more buffer size, for example, 20, then this will not be an error.

type_number specifies the type of car numbers to be recognized.

flags defines additional recognition modes. It is necessary to set 0 for the time being, and if it is necessary to output numbers even with a low quality of recognition of individual characters (including those with symbols replaced by the "?" Sign), then set the DEBUG_RECOGNITION_MODE flag to 1. The NO_LOW_RELIABILITY flag is used to reduce false positives, some numbers with low reliability can be discarded.

The flag RETURN_TYPE_NUMBER allows you to output after the recognized car number through the colon its type. For example X111XX11: 0.

The IR_LIGHTING_CAMERA flag indicates that an infrared camera is used for shooting, that is, the background is white in the image, and the numbers are black.

In version 1.8, the following return types are defined:

Russian car numbers

Car number type	Code	Description
TYPE_RUSSIAN_BASE	0	Base car numbers of Russia
TYPE_RUSSIAN_TRANSIT	1	Transit car numbers of Russia
TYPE_RUSSIAN_TRAILER	2	Trailer car numbers of Russia
TYPE_RUSSIAN_PUBLIC	3	Public transport car numbers of Russia
TYPE_RUSSIAN_POLICE	4	Russian police car numbers
TYPE_RUSSIAN_ARMY	5	Military car numbers of Russia
TYPE_RUSSIAN_SQUARE_BASE	6	Tractor or motorcycle numbers of Russia
TYPE_RUSSIAN_DIPLOMAT	7	Diplomat car numbers of Russia
TYPE_RUSSIAN_BASE_SQUARE	8	Base square plate number of Russia since 2019

Car numbers of Kazakhstan

Car number type	Code	Description
TYPE_KAZ_PRIVATE1993	21	Private car numbers of the 1993 standard
TYPE_KAZ_ORGANIZATION1993	22	Car numbers of the organizations of the 1993 standard
TYPE_KAZ_PRIVATE2012	23	Private car numbers of the 2012 standard
TYPE_KAZ_ORGANIZATION2012	24	Car numbers of organizations of the 2012 standard

Car numbers of the Republic of Belarus

Car number type	Code	Description
-----------------	------	-------------

TYPE_BY_2004_BASE	30	Basic car numbers of the 2004 standard
TYPE_BY_TRANSIT	31	Transit car numbers of standard STB 914-99 (type 12 and 12a as revised in 2011)
TYPE_BY_2004_TRAILER	32	Trailer car numbers of the 2004 standard
TYPE_BY_PUBLIC	33	Public transport car numbers
TYPE_BY_POLICE	34	Police car numbers
TYPE_BY_ARMY	35	Military car numbers
TYPE_BY_SQUARE_BASE	36	Rear Base Two-Line car numbers of the 2004 standard
TYPE_BY_2004_TRUCK	37	Truck and bus car numbers of the 2004 standard
TYPE_BY_1992_TRUCK	38	Truck and bus car numbers of the 1992 standard

Car numbers of Poland

Car number type	Code	Description
TYPE_PL_BASE	40	Base car numbers of the 2000 and 2006 standard

Car numbers of Latvia

Car number type	Code	Description
TYPE_LV_BASE	50	Base car numbers for the 1992 and 2004 standard

Car numbers of Lithuania

Car number type	Code	Description
TYPE_LT_BASE	60	Base car numbers of the 2004 standard

Car numbers of Estonia

Car number type	Code	Description
TYPE_EST_BASE	70	Base car numbers of the 2004 standard

Car numbers of Ukraine

Car number type	Code	Description
TYPE_UA_BASE	80	Base car numbers of the 2015 standard (passenger cars, trailers, buses)
TYPE_UA_TRANSIT	81	Car numbers for single trips on cars, trailers and buses of the 2015 standard
TYPE_UA_DIPLOMAT	82	Car numbers of diplomats and technical personnel of the 2013 standard

Car numbers of Moldova

Car number type	Code	Description
TYPE_MD_2011_BASE	90	Car numbers of legal entities of the 2011-2015 standard
TYPE_MD_2011_TRAILER	91	Trailer and semi-trailer car numbers of the 2011-2015 standard
TYPE_MD_2011_SQUARE_BASE	92	Motorcycle numbers 2011-2015 standard

It must be remembered that the buffer size for the text are larger than the number with the return type.

Types of recognizable car numbers

Types of recognition of Russian numbers

Car number type	Code	Image type	Description
ANPR_RUSSIAN_BASE	0	8bit, 1 channel	Basic rectangular
ANPR_RUSSIAN_BASE2	1	8bit, 1 channel	Basic and transit
ANPR_RUSSIAN_EXTENDED	2	8bit, 1 channel	Base, transit and trailer
ANPR_RUSSIAN_PUBLIC	3	8bit, 1 channel	Public transport only
ANPR_RUSSIAN_POLICE	5	8bit, 1 channel	Police car numbers only
ANPR_RUSSIAN_ARMY	6	8bit, 1 channel	Military car numbers only
ANPR_RUSSIAN_EXTENDED2	4	8bit, 3 channel	ANPR_RUSSIAN_EXTENDED + ANPR_RUSSIAN_PUBLIC
ANPR_RUSSIAN_FULL	7	8bit, 3 channel	ANPR_RUSSIAN_EXTENDED2 + ANPR_RUSSIAN_POLICE + ANPR_RUSSIAN_ARMY
ANPR_RUSSIAN_SQUARE_BASE	8	8bit, 1 channel	Square car numbers only
ANPR_RUSSIAN_FULL_WITH_SQUARE	9	8bit, 3 channel	ANPR_RUSSIAN_FULL + ANPR_RUSSIAN_SQUARE_EXTENDED
ANPR_RUSSIAN_DIPLOMAT	10	8bit, 1 channel	Diplomatic plates of Russia
ANPR_RUSSIAN_SQUARE_EXTENDED	11	8bit, 1 channel	All square plates available

8bit, 1 channel – image in gradations of gray; 8bit, 3 channel – color image.

Base car numbers [GOST R 50577-93]:



Рисунок А.1 — Регистрационный знак типа 1 с двухзначным кодом региона регистрации

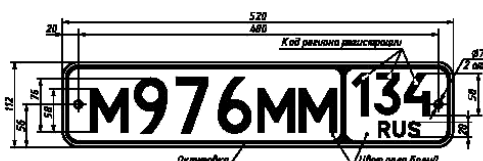


Рисунок А.2 — Регистрационный знак типа 1 с трехзначным кодом региона регистрации

Transit russian car numbers (MM976M34).

Primer: Sea Hare



Police car numbers are blue car numbers of M111122 format.

Square car numbers of motorcycles and tractors type:

MM22

Number type	Code	Image type	Description
-------------	------	------------	-------------

Car number type	Code	Image type	Description
ANPR_KAZ_1993_PRIVATE	100	8bit, 1 channel	Private car numbers of the 1993 Standard
ANPR_KAZ_1993_ORGANIZATION	101	8bit, 1 channel	Car numbers of organizations of the 1993 standard
ANPR_KAZ_2012_PRIVATE	102	8bit, 1 channel	Private car numbers of the 2012 standard
ANPR_KAZ_2012_ORGANIZATION	103	8bit, 1 channel	Car number of organizations of the 2012 standard
ANPR_KAZ_BASE	104	8bit, 1 channel	Private and organization car numbers of the 1993 and 2012 standards

Number type	Code	Image type	Description
-------------	------	------------	-------------

Car number type	Code	Image type	Description
ANPR_TM_2009	201	8bit, 1 channel	Private car numbers of the 2009 Standard
ANPR_TM_PRIVATE_BEFORE_2009	202	8bit, 1 channel	Private car numbers of the standard until 2009
ANPR_TM_BASE	203	8bit, 1 channel	All private car numbers

Number type	Code	Image type	Description
-------------	------	------------	-------------

Car number type	Code	Image type	Description
ANPR_BY_TRUCK	300	8bit, 1 channel	All cargo car numbers
ANPR_BY_2004_TRUCK	301	8bit, 1 channel	Cargo car numbers of the 2004 standard
ANPR_BY_1992_TRUCK	302	8bit, 1 channel	Cargo car numbers of the 1992 standard

ANPR_BY_2004_TRAILER	303	8bit, 1 channel	Rear car numbers for trailers and semi-trailers since 2004
ANPR_BY_2004_BASE	304	8bit, 1 channel	Passenger cars starting from 2004
ANPR_BY_SQUARE_BASE	305	8bit, 1 channel	Rear two-line car numbers, starting from 2004
ANPR_BY_2004_BASE2	306	8bit, 1 channel	ANPR_BY_2004_TRUCK + ANPR_BY_2004_BASE + ANPR_BY_2004_TRAILER
ANPR_BY_TRANSIT	307	8bit, 1 channel	Transit car numbers of standard STB 914-99 (type 12 and 12a as revised in 2011)
ANPR_BY_PUBLIC	308	8bit, 1 channel	Public transport
ANPR_BY_POLICE	309	8bit, 1 channel	Police car numbers
ANPR_BY_FULL	310	8bit, 3 channel	ANPR_BY_2004_BASE2 + ANPR_BY_TRANSIT + ANPR_BY_PUBLIC + ANPR_BY_POLICE
ANPR_BY_FULL_WITH_SQUARE	311	8bit, 3 channel	ANPR_BY_FULL + ANPR_BY_SQUARE_BASE
ANPR_BY_2004_BASE3	312	8bit, 1 channel	ANPR_BY_2004_BASE2 + ANPR_BY_TRANSIT

Types of recognition of car numbers in Poland

Car number type	Code	Image type	Description
ANPR_PL_BASE	400	8bit, 1 channel	Standard car numbers of Poland ANPR_PL_BASE_7 + ANPR_PL_BASE_8
ANPR_PL_BASE_7	401	8bit, 1 channel	Standard car numbers in Poland (7 characters)
ANPR_PL_BASE_8	402	8bit, 1 channel	Standard numbers in Poland (8 characters)

Types of recognition of car numbers in Latvia

Car number type	Code	Image type	Description
ANPR_LV_BASE	500	8bit, 1 channel	Car numbers of the 1992 and 2004 standards

Types of recognition of car numbers in Lithuania

Car number type	Code	Image type	Description
ANPR_LT_BASE	600	8bit, 1 channel	Standard car numbers since 2004

Types of recognition of car numbers in Estonia

Car number type	Code	Image type	Description
ANPR_EST_BASE	700	8bit, 1 channel	Car numbers of the 2004 standard

Types of recognition of car numbers in Ukraine

Car number type	Code	Image type	Description
ANPR_UA_BASE	800	8bit, 1 channel	Standard car numbers of 2015 (passenger cars, trailers, buses)
ANPR_UA_TRANSIT	801	8bit, 1 channel	Car numbers for single trips on cars, trailers and buses of the 2015 standard
ANPR_UA_DIPLOMAT	802	8bit, 1 channel	Car numbers of diplomats and technical personnel, 2013
ANPR_UA_BASE2	803	8bit, 1 channel	ANPR_UA_BASE + ANPR_UA_DIPLOMAT
ANPR_UA_BASE3	804	8bit, 1 channel	ANPR_UA_BASE2 + ANPR_UA_TRANSIT

Types of recognition of car numbers in Moldova

Car number type	Code	Image type	Description
ANPR_MD_2011_BASE	900	8bit, 1 channel	Car numbers legal entities 2011-2015 + taxi car numbers
ANPR_MD_2011_TRAILER	901	8bit, 1 channel	Trailer and semi-trailer car numbers 2011-2015
ANPR_MD_2011_SQUARE_BASE	902	8bit, 1 channel	Motorcycle numbers 2011-2015
ANPR_MD_2011_BASE2	903	8bit, 1 channel	ANPR_MD_2011_BASE + ANPR_MD_2011_TRAILER
ANPR_MD_2011_FULL_WITH_SQUARE	911	8bit, 3 channel	ANPR_MD_2011_BASE2 + ANPR_MD_2011_SQUARE_BASE

Type ANPR_CUSTOM_TYPE

This type is a configurable user type and has a value of -1. In this case, the custom field in the structure must be filled. For all other types of recognition, the field is empty. The image on the input can be fed as full-color or gray.

The configurable type is defined in iANPRCustom.h as follows:

```
const int max_ianpr_custom_in_types = 20;
struct iANPRCustomElement
{
    int all_types;
    float probability; // Probability of the country
    int types[max_ianpr_custom_in_types]; // Types are taken in
    iANPR
```

```

        float probability_types[max_ianpr_custom_in_types]; // The
probability of each type, you can not fill
    };
    struct iANPRCustom
    {
        int all_countries;
        iANPRCustomElement* Elements;
        int flags; // By default is 0
    };

```

In the current version, the probability of each type does not work. The probability of a country is not really a probability, but a signification. The obtained probability of the number is multiplied by the importance value. Let's take, for example, the Orenburg region, where car numbers from Kazakhstan come across quite often. Determine the level of significance for Russia 1.2, and for Kazakhstan 1. Then if the car number from Russia was identified, then its probability will be multiplied by 1.2. If at the same time the car number from Kazakhstan is determined, then its probability will be multiplied by 1. I.e. car number from Russia is more significant than from Kazakhstan. We can consider this as a form of a priori probability.

If you write a program in C/C++, you can fill the structure yourself, for example, like this:

```

iANPRCustom* custom = NULL;
custom = new iANPRCustom;
custom->all_countries = 2;
custom->Elements = new iANPRCustomElement[2];
custom->Elements[0].all_types = 5;
custom->Elements[0].probability = 1.2f;
custom->Elements[0].types[0] = CUSTOM_RUSSIAN_BASE_EXTENDED;
custom->Elements[0].types[1] = CUSTOM_RUSSIAN_PUBLIC;
custom->Elements[0].types[2] = CUSTOM_RUSSIAN_POLICE;
custom->Elements[0].types[3] = CUSTOM_RUSSIAN_ARMY;
custom->Elements[0].types[4] = CUSTOM_RUSSIAN_SQUARE;
custom->Elements[1].all_types = 1;
custom->Elements[1].probability = 1.0f;
custom->Elements[1].types[0] = CUSTOM_KAZ_2012_PRIVATE;

```

However, in this case, remember that you must delete the selected elements (memory) by yourself. For other programming languages, you can use the creation and deletion functions from iANPRCustom.h:

```

void* CreateiANPRCustom(char* xml_buffer, int buffer_size);
void DeleteiANPRCustom(void* xml);

```

An example of an xml file that is identical to the code above:

```

<?xml version="1.0"?>
  <countries value="2">
    <country all_types="5" probability="1.2" comment="First
country - Russia">
      <type value="2"/>
      <type value="3"/>
    </country>
  </countries>

```

```

        <type value="5"/>
        <type value="6"/>
        <type value="8"/>
    </country>
    <country all_types="1" probability="1" comment="Second
country - Kazakhstan">
        <type value="102"/>
    </country>
</countries>

```

The flags in the iANPRCustom structure are intended for additional features. At the moment only one FLAG_CUSTOM_MULTI_RESULT flag is supported, which allows to return not one most probable car number, but the probable car numbers found (not more than 3). Moreover, the first car number is most probable. The returned car numbers are in the same text line, so it must be made larger (at least 40 bytes), the car numbers will be returned in the following form:

Car number 1 | Car number 2 | Car number 3

To register a flag in xml, you need to change the line with the number of countries like this:

```
<countries value="2" multi="1">
```

Rotation of the input image (alpha, beta, gamma)

iANPR SDK assumes: if the car number on the image is rotated, then the rotation angle is insignificant (more details in the [requirements](#) section of the recognition algorithm). To correct a significant angle, you can use the parameters *alpha*, *beta*, *gamma* from [ANPR_OPTIONS](#). *Alpha* sets the rotation around the X axis, *beta* sets the rotation around the Y axis, *gamma* sets the rotation around the Z axis. The value 90.0 means no rotation. To determine the rotation angles along the axes, use the utility [persptrans](#), which can be found in the iANPR SDK.

anprPlateRect

The function of searching for car numbers on the region of the image. Image has the OpenCV format.

```

int anprPlateRect(
    IplImage* Image,
    CvRect Rect,
    ANPR_OPTIONS Options,
    int* AllNumber,
    CvRect* Rects,
    char** Texts,

```



```
void* param = NULL  
);
```

The parameters are the same as the [anprPlate](#) function, the additional Rect parameter specifies the region to be processed.

The returned values are the same, but an error is added:
ERROR_RECT (-1) is an invalid region.

LicenseValue

Activation function of the licensed version of iANPR. Calling this function is required only for the licensed version of the library and only once before the first recognition. If you use the LicenseValue function with a demo version of iANPR, it will not affect the operation of the program in any way.

```
void LicenseValue(  
    char* lic  
);
```

Parameters:

lic – the array containing the license key.

The function does not return values.

1.2. Interface module – iANPRinterface

The interface module extends the connectivity to the library. Definitions of functions are presented in iANPRinterface.h.

anprPlateMemory

The function is intended for recognition of a graphic file of formats BMP, JPEG, PNG, TIFF, which is in memory. For example, a BMP file is read from the hard disk into memory, and a pointer to it is passed to the function.

```
int anprPlateMemory(  
    char* in_buffer,  
    int size_buffer,  
    ANPR_OPTIONS Options,
```

```

    int* AllNumber,
    CvRect* Rects,
    char** Texts
);

```

Parameters:

in_buffer – the pointer to the input image;

size_buffer – the size of the image buffer;

The remaining parameters are similar to the [anprPlate](#) function.

The return values are the same as in [anprPlate](#).

anprPlateMemoryRect

The function assignment is similar to [anprPlateMemory](#), just like in [anprPlateRect](#), the search area is added.

```

int anprPlateMemoryRect(
    char* in_buffer,
    int size_buffer,
    CvRect Rect,
    ANPR_OPTIONS Options,
    int* AllNumber,
    CvRect* Rects,
    char** Texts
);

```

The assignment of parameters and return values is similar to [anprPlateMemory](#) and [anprPlateRect](#).

anprPlateMat

The function is similar to [anprPlate](#), except that the first parameter is an image in the format cv::Mat of the C++ image interface in OpenCV.

```

int anprPlateMat(
    cv::Mat Image,
    ANPR_OPTIONS Options,
    int* AllNumber,
    CvRect* Rects,
    char** Texts
);

```

The assignment of parameters and return values is similar to [anprPlate](#).

anprPlateMatRect

The function assignment is similar to [anprPlateMat](#), just like in [anprPlateRect](#), the search area is added.

```
int anprPlateMatRect(  
    cv::Mat Image,  
    CvRect Rect,  
    ANPR_OPTIONS Options,  
    int* AllNumber,  
    CvRect* Rects,  
    char** Texts  
);
```

The assignment of parameters and return values is similar to [anprPlate](#).

anprPlateXML

A function similar to [anprPlate](#), except that the numbers returned are returned in XML format.

```
int anprPlateXML(  
    IplImage* Image,  
    ANPR_OPTIONS Options,  
    char* xml_buffer,  
    int* size_xml_buffer  
);
```

Parameters:

Image – the input image in OpenCV format (8-bit 1-channel or 8-bit 3-channel depending on Options.type_number parameters);

Options – the settings for the recognition mode in [ANPR_OPTIONS](#) structure format;

xml_buffer - the pointer to the buffer allocated in the memory before calling the function, which will return an XML string of the following format:

```
<?xml version="1.0" encoding="windows-1251"?>  
<action_result version='1.0'>  
    <allnumbers value='1'>  
        <number value='M976MM134' coord='243,256,177,53'>  
        </number>  
    </allnumbers>  
</action_result>
```

allnumbers shows the number of car plates found. And for each found car plate, the *number* tag is returned, whose *value* is the text of the car plate, and coord is its coordinates (X, Y, width, height).

size_xml_buffer - the size of the allocated buffer, after the function call this variable will contain the size of the recorded XML string.

Return values are the same as in [anprPlate](#), a possible error is added:

ERROR_SIZE_XML_BUF (-3) XML buffer size is insufficient.

**anprPlateRectXML, anprPlateMemoryXML,
anprPlateMemoryRectXML, anprPlateMatXML,
anprPlateMatRectXML**

Variants of the previous functions with parameters returning via XML.

```
int anprPlateRectXML(  
    IplImage* Image,  
    CvRect Rect,  
    ANPR_OPTIONS Options,  
    char* xml_buffer,  
    int* size_xml_buffer  
);  
  
int anprPlateMemoryXML(  
    char* in_buffer,  
    int size_buffer,  
    ANPR_OPTIONS Options,  
    char* xml_buffer,  
    int* size_xml_buffer  
);  
  
int anprPlateMemoryRectXML(  
    char* in_buffer,  
    int size_buffer,  
    CvRect Rect,  
    ANPR_OPTIONS Options,  
    char* xml_buffer,  
    int* size_xml_buffer  
);  
  
int anprPlateMatXML(  
    cv::Mat Image,  
    ANPR_OPTIONS Options,  
    char* xml_buffer,
```

```

        int* size_xml_buffer
    );

int anprPlateMatRectXML(
    cv::Mat Image,
    CvRect Rect,
    ANPR_OPTIONS Options,
    char* xml_buffer,
    int* size_xml_buffer
);

```

Parameters and return values are similar to the functions described above.

1.3. Streaming module - iANPRcapture

The streaming module is designed to increase the reliability of recognition of car numbers on the video stream by combining recognition results from several frames.

The functions of this module are defined in iANPRcapture.h. **ATTENTION!** A multi-type (several numbers for ANPR_CUSTOM_TYPE) can not be used at the moment.

CreateiANPRCapture

Creating an iANPR-stream.

```

iANPRCapture CreateiANPRCapture(
    int max_frames,
    ANPR_OPTIONS Options,
    CvRect Rect
);

```

Parameters:

max_frames - the number of frames from which the result is combined;

Options - recognition settings (filled structure [ANPR_OPTIONS](#));

Rect - recognition area in CvRect format.

Returns the selected iANPRCapture object. If it fails, it returns NULL.

ReleaseiANPRCapture

Release memory from the iANPRCapture object.

```
void ReleaseiANPRCapture(  
    iANPRCapture *Capture  
);
```

Parameters:

Capture – is a pointer to an iANPRCapture object.

AddFrameToiANPRCapture

The function adds the current frame to the iANPRCapture stream and returns the recognized car numbers.

```
int AddFrameToiANPRCapture(  
    iANPRCapture Capture,  
    IplImage* Image,  
    int* AllNumber,  
    CvRect* Rects,  
    char** Texts  
);
```

Parameters:

Capture – the iANPRCapture object.

Image – the input image.

The remaining parameters are similar to [anprPlate](#).

The return values are the same as [anprPlate](#).

AddFrameToiANPRCaptureMat

The function adds the current frame (cv::Mat) to the iANPRCapture stream and returns the recognized car numbers..

```
int AddFrameToiANPRCaptureMat(  
    iANPRCapture Capture,  
    cv::Mat Mat,  
    int* AllNumber,  
    CvRect* Rects,  
    char** Texts  
);
```

The parameters are similar to [AddFrameToiANPRCapture](#) except that the current frame is transmitted in the cv::Mat structure.

The return values are the same as [anprPlate](#).

CreateMemoryForiANPRCapture

The function allocates memory for additional streaming recognition functionality. It is necessary to call only if the function [GetNumbersInMemory](#) is assumed later.

```
int CreateMemoryForiANPRCapture(  
    iANPRCapture Capture,  
    int min_frames_with_plate,  
    int frames_without_plate,  
    int max_plates_in_mem  
);
```

Parameters:

Capture – the iANPRCapture object.

min_frames_with_plate - the number of frames between the first and last recognized car plate, after which you can assume that this is really a car plate.

frames_without_plate - the number of frames without the previously recognized car plate, after which you can output the result.

max_plates_in_mem - maximum number of car plates in memory.

The function returns 0 on success, otherwise error.

GetNumbersInMemory

Returns the summed result of finding a car number from memory. It returns only after *frames_without_plate* is passed, which set in [CreateMemoryForiANPRCapture](#).

```
int GetNumbersInMemory(  
    iANPRCapture Capture,  
    int* AllNumber,  
    char** Texts,  
    int Size_Texts,  
    CvPoint* Points,  
    int* all_point  
);
```

Parameters:

Capture – the iANPRCapture object.

AllNumber – the number of recognized car plates is returned to the last frame from memory (initially contains the buffer size (number of car plates) in **Texts** array).

Texts – the contents of the car plate.

Size_Texts – the size of each element (string) of the Texts array.

Points – a pointer to the CvPoint array allocated for the trajectory before the call (if NULL, then the trajectory does not return).

all_points – here the size of the **Points** array is transferred, the number of points found is returned, and if the array is smaller (1000 elements are recommended) than the points actually found, only the number of points for which there is space will be returned.

The function returns 0 on success, otherwise error code. If two car plates are returned simultaneously in the frame, which is unlikely, because of the summation and delay, the trajectory will return only for the first.

CreateLineIntersection

The function creates a line to fix the intersection. In fact, the line consists of two lines - and the intersection is fixed only when car number intersects both lines (done to avoid false positives).

```
int CreateLineIntersection(  
    iANPRCapture Capture,  
    CvPoint p1a,  
    CvPoint p2a,  
    CvPoint p1b,  
    CvPoint p2b  
);
```

Parameters:

Capture – is the iANPRCapture object.

p1a и **p2a** – the points characterizing the top line (the line can't be vertical - the maximum of the delta x can be 3 times smaller than the delta y, otherwise ERROR_SLOPE_LINE).

p1b и **p2b** – the points characterizing the bottom line.

The function returns 0 on success, otherwise error. The lines must be parallel to each other. Otherwise, ERROR_NO_PARALLEL_LINES..

LicenseCapture

Activation function of the licensed version of iANPRcapture. Calling this function is required only for the licensed version of the library and only once before the first recognition. If you use the LicenseCapture function with the demo version of iANPRcapture, it will not affect the operation of the program in any way.

```
void LicenseCapture(  
    char* lic  
);
```


Parameters:

lic – an array containing the license key.

The function does not return values.

2. Installation and use

There are no special requirements for the installation.

2.1. Windows

To use the iANPR SDK on your computer, you need to install:
Microsoft Visual C++ 2015 Redistributable

<https://www.microsoft.com/ru-ru/download/details.aspx?id=48145>

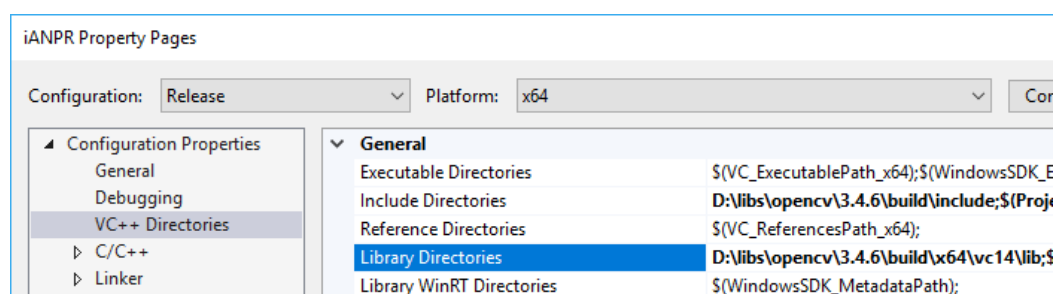
Download OpenCV 3.4.0

https://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.4.0/opencv-3.4.0-vc14_vc15.exe/download

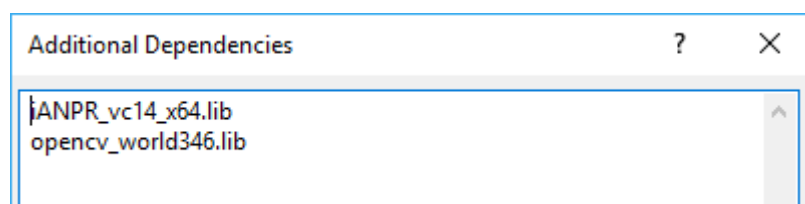
Although the required OpenCV libraries come with the iANPR SDK, you probably will need the header files to develop your programs. iANPR used libraries compiled by vc14.

Next, you connect in the form of normal dynamic libraries.

If you implement a project in C/C++ on Visual Studio, then write the path to .h and .lib for OpenCV and the location of the iANPR:



Add plug-in libraries (in the properties of the linker):



After that, make sure that all dlls from the x86 or x64 folder are either in the folder with your executable file, or in the folder specified in the PATH variable.

2.2. Linux

Used OS – Debian 9.9.0 desktop i386

1. Guided by

http://docs.opencv.org/doc/tutorials/introduction/linux_install/linux_install.html

install additional software for OpenCV.

2. Download the OpenCV 3.4.6 for linux from here.

<http://sourceforge.net/projects/opencvlibrary/>

And compile it.

3. Download and unpack the iANPR SDK.

4. It is necessary to make the library visible for programs. To do this, you can write the path to libianpr_86.so or just copy this file to the same place as the OpenCV libraries:

/usr/local/lib/

5. Update paths to libraries

ldconfig

6. In the example folder for Linux, open the makefile and fix lianpr_64 to lianpr_86 (as we compile in the 32-bit version of Linux).

7. In the same folder, we call

make

You must create an executable file.

8. Check the work of the program.

3. C/C++ examples for Windows

3.1. Image

The source code of the examples is constantly being improved. The description below shows the most significant parts of the code. However, the full source code of the examples may differ. The full source code for the example image is located in the samples folder distributed with the iANPR SDK.

```
#include "opencv2/highgui/highgui_c.h"
#include "../Include/iANPR.h"
#include <stdio.h>
#include <Windows.h>

int main( int argc, char** argv)
{
    IplImage* Img = 0;
    IplImage* grayImg = 0;

    // filter input
    if (argc < 2)
    {
        printf ("Too few arguments. For help print %s /?", argv [0]);
        return -1;
    }
    else if (!strcmp (argv [1], "help") || !strcmp (argv [1], "-help") || !strcmp
(argv [1], "--help") || !strcmp (argv [1], "/?"))
    {
        printHelp (argv [0]);
        return 0;
    }
    else if (argc < 3)
    {
        printf ("Too few arguments. For help print %s /?", argv [0]);
        return -2;
    }
    else // argc >= 3
        Img = cvLoadImage( argv[2], CV_LOAD_IMAGE_COLOR );

    if (!Img)
    {
        printf( "Can't load file!\n");
        return -4;
    }

    CvRect Rects[100];
    int all = 100;
    char** res = new char*[all];
    for(int j=0;j<all;j++) res[j] = new char[20];
    ANPR_OPTIONS a;
    a.Detect_Mode = ANPR_DETECTCOMPLEXMODE;
```

```

a.min_plate_size = 500;
a.max_plate_size = 50000;
a.max_text_size = 20;
a.type_number = atoi (argv [1]);
a.flags = 0;
a.max_threads = 1;

bool isFullType = false;
for (size_t i = 0; i < anprFullTypesCount; i++)
if (anprFullTypes[i] == a.type_number)
    isFullType = true;

// LicenseValue is only required for paid versions
// And only once, before the first recognition.
char* key = new char[8001]; memset(key, 0, 8001);
FILE* f = fopen("lic.key", "rb");
if (f != NULL)
{
    fread((void*)key, 8000, 1, f);
    fclose(f);
}
else
    puts("WARNING! File lic.key not found. This may crash program if you use
license version of iANPR SDK dlls");

LicenseValue(key);
delete [] key; key = 0;

int i = -9999;
if (isFullType)
    i = anprPlate( Img, a, &all, Rects, res );
else
{
    grayImg = cvCreateImage (cvGetSize (Img), 8, 1);
    cvCvtColor (Img, grayImg, CV_BGR2GRAY);
    i = anprPlate( grayImg, a, &all, Rects, res );
}

if ( i == 0 )
    for( int j = 0; j < all; j++ )
    {
        printf( "%s\n", res[j] );
    }
else
    printf( "Error:%d\n", i );

for(int j=0;j<100;j++) delete [] res[j];
delete [] res;
cvReleaseImage ( &Img );
cvReleaseImage ( &grayImg );

return 0;
}

```

The example loads an image using the cvLoadImage function. Further we allocate memory for storing 100 car numbers (we assume that no more than 100 car numbers will be found, of course, you can install less).

Next we fill the structure [ANPR_OPTIONS](#), set the mode of detecting car numbers ANPR_DETECTCOMPLEXMODE. The range of area (in pixels in square) of car numbers, 1 stream for recognition is established.

The value 20 corresponds to the above buffer for storing car numbers. The type of number detection is set to basic, which means that if the trailer car numbers, for example, is in the frame, they can't be recognized correctly.

The license key is read from the file `lic.key` using the `fread` function, and then the key is passed to the `LicenseValue` function. Calling this function is required only for the licensed version of the iANPR SDK and only before the first recognition.

Next we call the [anprPlate](#) function from iANPR SDK. After that, the car numbers found in the image are output to the console.

At the end, the previously allocated memory is released.

Example of using:

```
image.exe 0 ..\images\image.bmp
```

Information will be displayed on the console, for redirecting to the file:

```
image.exe 0 ..\images\image.bmp > res.txt
```

3.2. Image_new

This example shows how to use various interfaces to work with the library. Unlike the previous example, an additional header file `iANPRinterface.h` is connected.

The `Memory` function in the example shows the ability to read JPEG, BMP, PNG, TIFF files from memory. You can read the file into memory, and then pass the pointer to [anprPlateMemory](#).

The `WithMat` function provides access to the C++ interface based on `Mat`. The image is recognized via the [anprPlateMatRect](#) function.

The functions `XMLWork` and `XMLWork2` provide examples of working with the functions [anprPlateRectXML](#) and [anprPlateMatRectXML](#), respectively. The result is displayed as an XML string in the console.

3.3. Image_omp

In the version of iANPR 1.6, built-in parallelization appeared. This example can be used to compare the results of the work. In this example, parallelization is achieved by splitting the input image into separate parts.

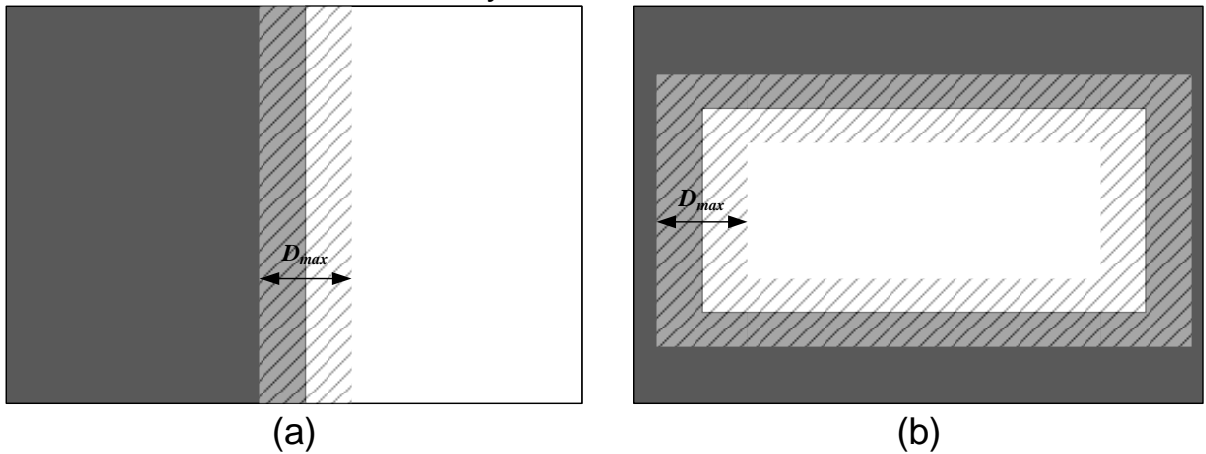
When working with large images, for example, 1920x1080, the recognition time on a conventional PC may not be sufficient for real-time operation. To solve this problem, you can split the image into parts and analyze them in parallel. Here, however, it should be remembered that if car number is located on the intersection of parts, it will not be recognized.

Therefore, it is necessary to introduce some redundancy, using intersecting parts. And the level of intersection is determined by the maximum possible size of the object recognition.

Suppose that the object of recognition is in the form of a circle, and its maximum diameter in the image is D_{max} . Then the width of the intersection of segments should be greater than D_{max} . In this case, the area of the analyzed area (in different segments) will be calculated using the following formula:

$$S = L * D_{max},$$

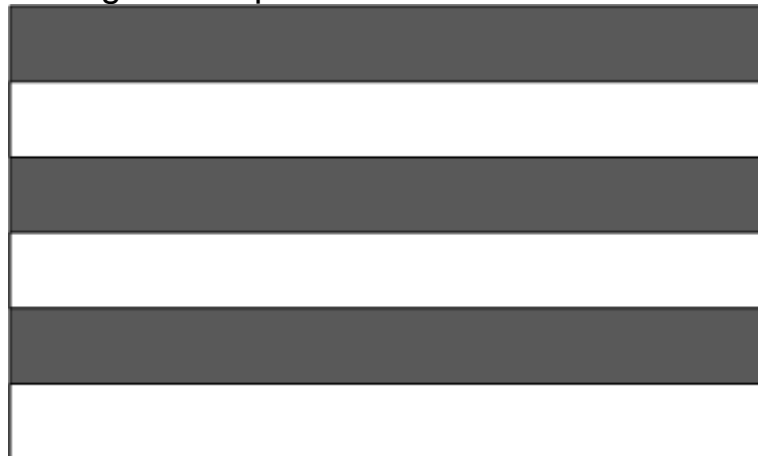
where L is the length of the boundaries between all segments. It is clear that in this case S will depend not only on the number of segments, but also on their shape. The following figure shows two examples of the arrangement of segments. One segment is gray and the other is white. The intersection area is indicated by dashes.



Decomposition with intersecting boundaries: with a small area of intersection (a), with a significant intersection area (b)

The disadvantage is that when the image is fully analyzed, Width * Height will not be analyzed, but Width * Height + S.

Suppose that we know the number of processor cores, for example, 6. We divide the image into 6 parts as follows:



It is natural to remember the area of intersection of parts. $1080/6 = 180$

Let's take the height of 80 pixels as the maximum height of the number. Then the parts up and down will increase by 40 pixels, except for the top and bottom, where the increase is only one way.

In the example, parallelization occurred on the basis of the OpenMP library.

For each thread, we created our own image, after which a parallel loop was started:

```
#pragma omp parallel
{
#pragma omp for
    for( i = 0; i < max; i++ )
    {
        CvRect Rects[100];
        int all = 100;
        char** res = new char*[all];
        for(int j=0;j<all;j++) res[j] = new char[20];
        ANPR_OPTIONS a;
        a.Detect_Mode = ANPR_DETECTCOMPLEXMODE;
        a.min_plate_size = 500;
        a.max_plate_size = 50000;
        a.max_text_size = 20;
        a.type_number = ANPR_RUSSIAN_BASE;
        a.flags = 0;

        int i2 = anprPlate( Images[i], a, &all, Rects, res );
        if ( i2 == 0 )
            for( int j = 0; j < all; j++ ) {
                {
                    printf( "%s\n", res[j] );
                }
            }

        for(int j=0;j<100;j++) delete [] res[j];
        delete [] res;
    }
}
```

First, let's compare how much redundancy is, compiling the program with OpenMP turned off. Successful recognition was in both cases. Recognition time on AMD-FX (tm)-6100 Six-cores 3.3GHz (on one core), 8GB of RAM, Windows 7 64:

Serial block 0.297c

Parallel block 0.369s

The redundancy of calculations is approximately 1.24 times. In so much slowed down the processing.

After enabling OpenMP, the parallel block worked for 0.1s. The productivity gain is 2.9 times.

Why such a small increase? Answer: 1) redundancy; 2) the calculations in the blocks are uneven - where the number was found, and where not, so the operating time is the time of recognition of the block with the maximum information.

But even such a performance increase - almost 3 times. And it allows for 10 seconds to process 10 frames, which for real time can be enough with these settings.

3.4. Capture

An example of Capture works with a Web camera, but if you specify a video file as the command line argument, it will handle it. The recognition information is output directly to the frame as follows:



The feature of this example of stream car number recognition is that the result of recognizing not every frame is produced, but it is checked whether there is the same found car number in the previous frame. If the car number appeared, then the recognition result is output.

This approach allows you to discard a significant part of false positives that occur in a separate frame.

3.5. Capture_(iANPRcapture)

This example shows how to work with the [iANRcapture](#) module.

This module allows to achieve higher recognition accuracy than in the Capture example, because it checks not a coincidence of car numbers in both frames, but calculates the overall reliability of the recognized symbols in the frame with the depth (number of frames) specified by the programmer:

```
i_capture = CreateiANPRCapture( 10, a, cvRect( 0, 0, frame->width, frame->height ) );
```

Each frame is added to the stream object

```
i1 = AddFrameToiANPRCapture( i_capture, object, &all, Rects, res );
```

Where the old frame is forced out and checked - was there a similar car number on the previous frames. If the car number found in the current frame is similar to the car numbers in one or more previous ones, the recognition results are summed up.

3.6. (iANPRcapture_motion)

This example is an improvement on the previous one. Allows you to calculate the trajectory of the movement of the car plate and detect the intersection of the line, i.e. it is possible to realize the functional of detecting the entrance-exit of the car.

The difference of this example is as follows.

Initially, additional memory is created for additional functionality (you do not need to delete it later, it will be cleaned together with the removal of the object).

```
CreateMemoryForiANPRCapture( i_capture, 10, 15, 100 );
```

The maximum number of car plates in memory is 100. If 15 frames were not the car plate recognized earlier, then the result is output. A car plate is considered recognized only if its initial detection was earlier than 10 frames in the final one.

Then the intersection line is created:

```
Lines[0].x = int( frame->width * 0.1f ); Lines[0].y = int( frame->height * 0.6f );  
Lines[1].x = int( frame->width * 0.3f ); Lines[1].y = int( frame->height * 0.6f );  
Lines[2].x = int( frame->width * 0.1f ); Lines[2].y = int( frame->height * 0.7f );  
Lines[3].x = int( frame->width * 0.3f ); Lines[3].y = int( frame->height * 0.7f );  
CreateLineIntersection( i_capture, Lines[0], Lines[1], Lines[2], Lines[3] );
```

To obtain such a refined result, each time (in each frame), after calling the AddFrameToiANPRCapture function, call the function returning the trajectory:

```
GetNumbersInMemory( i_capture, &all, res , 20, Points, &all_points );
```

If the value of all_points is greater than 0, then the car plate is found. And the display shows the trajectory of the car plate and the car plate itself:

```
for(int j = 0; j < all_points; j++ )
```

```

{
    cvCircle( frame, Points[j], 5, CV_RGB(0,0,255), 3 );
    if ( j > 0 ) cvLine( frame, Points[j], Points[j-1], CV_RGB(0,0,255), 2 );
}

CvFont font;
float aa=0.001f*frame->width;
cvInitFont( &font, CV_FONT_HERSHEY_SIMPLEX, aa,
            aa,0,1, 8 );

CvSize size;
int b;
cvGetTextSize( res[0], &font, &size, &b );
cvRectangle( frame, cvPoint(0, 0 ), cvPoint( size.width + 2,
        50 ), CV_RGB( 255, 255, 255 ), CV_FILLED );

CvPoint pp2,pp1;
pp2.x=0;
pp2.y=40;
pp1.x=1;
pp1.y=41;
cvPutText( frame, res[0], pp1, &font, CV_RGB(0,0,0) );
cvPutText( frame, res[0], pp2, &font, CV_RGB(0,255,0) );
cvResize( frame, image );
cvShowImage( "frame", image);

```

3.7. Persptrans

The utility `persptrans` is designed to eliminate the rotation of the car number on the original image. The rotation angle is specified in the [ANPR_OPTIONS](#) structure using the `alpha`, `beta`, and `gamma` parameters. The *alpha* angle specifies the rotation around the X axis, the angle *beta* - around the Y axis, the *gamma* angle - around the Z axis. The current *alpha*, *beta* and *gamma* parameters are output to the standard output stream (console). The obtained values can be used in the [ANPR_OPTIONS](#) structure, which is transferred to one of the recognition functions, for example, in [anprPlate](#).

The following keys are used to control the program:

- w, s - rotation around the X axis;
- a, d - rotation around the Y axis;
- q, e - rotation around the Z axis;
- r - turn off/on the recognition;
- g - show/hide the grid;
- v, b - increase/decrease the number of grid cells;
- esc (escape) - quit.

Press the control keys in the "Transformed" window. Rotate the image until you reach the desired result in the recognition.

Note: Try to make sure that the edges of license plate (or at least bottom edge) on the transformed image were as much as possible parallel to the borders of the window.

Note: Sometimes a smaller rotation is better than the bigger one.

Example of use:

```
persptrans.exe image.bmp 0
```

In this usage example, image.bmp will be loaded, the base Russian car numbers (ANPR_RUSSIAN_BASE, that is, type 0) will be recognized.

4. Examples in other programming languages for Windows

4.1. C#

In C#, it is recommended to use the XML interface to work with the library. In the library this is an example of a platereader.

The function from the library is connected as follows:

```
[StructLayout(LayoutKind.Sequential, Pack = 0)]
public struct ANPR_OPTIONS
{
    public byte sign1, sign2, sign3;
    public int min_plate_size;
    public int max_plate_size;
    public int Detect_Mode;
    public int max_text_size;
    public int type_number;
    public int flags;
    public IntPtr custom;

    public IntPtr vers;

    public double alpha;
    public double beta;
    public double gamma;

    public int max_threads;
};
#if WIN32
[DllImport("iANPRInterface_vc12_x86.dll", CallingConvention =
CallingConvention.StdCall)]
unsafe public static extern int anprPlateMemoryXML(byte[] in_buffer, int
size_buffer, ANPR_OPTIONS Options,
    StringBuilder xml_buffer, int[] size_xml_buffer);
[DllImport("iANPR_vc12_x86.dll", CallingConvention = CallingConvention.StdCall)]
unsafe public static extern void LicenseValue(sbyte[] key);
#elif WIN64
[DllImport("iANPRInterface_vc12_x64.dll", CallingConvention =
CallingConvention.StdCall)]
unsafe public static extern int anprPlateMemoryXML(byte[] in_buffer, int
size_buffer, ANPR_OPTIONS Options,
    StringBuilder xml_buffer, int[] size_xml_buffer);
[DllImport("iANPR_vc12_x64.dll", CallingConvention = CallingConvention.StdCall)]
unsafe public static extern void LicenseValue(sbyte[] key);
#endif
```

In the project settings, you must enable unsafe code and specify WIN32 and WIN64 conditional compilation symbols for the x86 and x64 build configurations, respectively.

In the example, the file is read completely into memory and passed to the library function:

```
ANPR_OPTIONS anpr;
anpr.Detect_Mode = 14;
anpr.min_plate_size = 500;
anpr.max_plate_size = 25000;
anpr.max_text_size = 20;
anpr.type_number = 0;
anpr.flags = 1;

anpr.sign1 = (byte)'i'; anpr.sign2 = (byte)'a'; anpr.sign3 = (byte)'1';

anpr.vers = Marshal.AllocHGlobal(4);
Marshal.WriteByte(anpr.vers, (byte)'1');
Marshal.WriteByte(anpr.vers + 1, (byte)'.');
Marshal.WriteByte(anpr.vers + 2, (byte)'6');
Marshal.WriteByte(anpr.vers + 3, 0);

anpr.alpha = 90;
anpr.beta = 90;
anpr.gamma = 90;

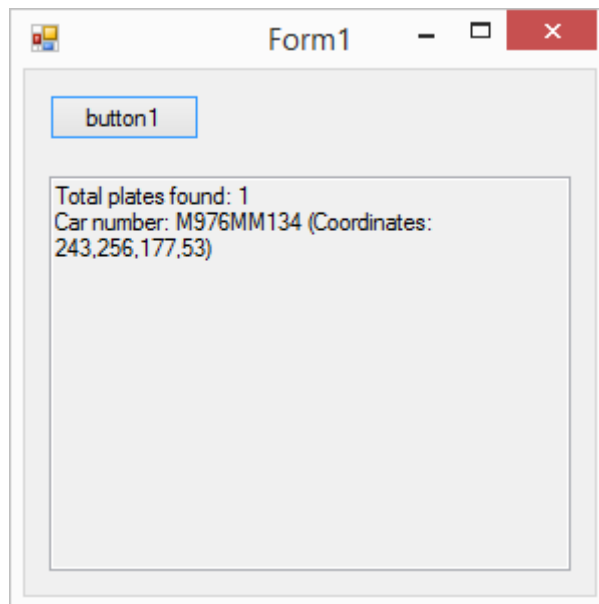
anpr.max_threads = 1;

StringBuilder buffer_builder = new StringBuilder(10000);
int[] size_builder = new int[1];
size_builder[0] = 10000;
int result = anprPlateMemoryXML(buffer, size, anpr, buffer_builder, size_builder);
```

To find out the recognition result, you need to look at the returned XML:

```
StringBuilder output = new StringBuilder();
using (XmlReader reader = XmlReader.Create(new StringReader(buffer_builder.ToString())))
{
    reader.ReadToFollowing("allnumbers");
    reader.MoveToFirstAttribute();
    string numbers = reader.Value;
    output.AppendLine("Plates found: " + numbers);
    int all = Convert.ToInt32(numbers);
    for (int i = 0; i < all; i++)
    {
        reader.ReadToFollowing("number");
        reader.MoveToFirstAttribute();
        string num = reader.Value;
        reader.MoveToNextAttribute();
        string num_coord = reader.Value;
        output.AppendLine("Homep: " + num + " (Coordinates: " + num_coord + ")");
    }
}
```

As a result, the car numbers and their coordinates are displayed in the form:



4.1.1. Example iANPRcapture_motion in C# for iANPR SDK

The program `iANPRcapture_motion_CShrp` is designed to demonstrate the capabilities of the iANPR SDK in calculating the trajectory of the car number and detecting the intersection of the car number of predefined lines, i.e. to demonstrate the possibility of implementing the functionality of vehicle entry/exit detection using the iANPR SDK. This program is written in C# and is analogous to the `iANPRcapture_motion` program written in C++. These and other usage examples are distributed as part of the iANPR SDK.

The current version of `iANPRcapture_motion_CShrp` is used like this:
`iANPRcapture_motion_CShrp.exe 7 D:/video.avi`

Where 7 is the type of the recognized number (Russian numbers),
`D:/video.avi` is the full name of the video file.

4.2. Delphi

Of course, in Delphi you can also use XML to return results, but here is an example of how to call functions from the [iANPR demo](#) without XML using Delphi 7. To call functions from the paid version of iANPR, before using the recognition functions for the first time, load the license key using functions [LicenseValue](#) or [LicenseCapture](#) to activate `iANPR.dll` or `iANPRcapture.dll`, respectively.

Definition of types:

```
type
  ANPR_OPTIONS = Record
    min_plate_size:integer;
    max_plate_size:integer;
    Detect_Mode:integer;
    max_text_size:integer;
    type_number:integer;
    flags:integer;
  end;
```

```
type
  CvRect = Record
    x:integer;
    y:integer;
    width:integer;
    height:integer;
  end;
```

```
type
  PRect = ^CvRect;
```

Connection the function:

```
function anprPlateMemoryRect( in_buffer: PChar; size_buffer: integer; Rect: CvRect;
Options: ANPR_OPTIONS; AllNumber: PInt; Rects: PRect; Texts: PPChar ): integer;
stdcall; external 'iANPRinterface_vc12_x86.dll'
  name 'anprPlateMemoryRect';
```

Reading from a file and calling the function:

```
with TFileStream.create(File_, fmOpenReadWrite) do
  try
    GetMem(p, Size);
    read(p^, Size);
    s := Size;
  finally
    free;
  end;
  all := 100;
  anpr.min_plate_size := 500;
  anpr.max_plate_size := 25000;
  anpr.Detect_Mode := 6; // ANPR_DETECTMODE2 | ANPR_DETECTMODE3;
  anpr.max_text_size := 20;
  anpr.type_number := 0; // ANPR_RUSSIAN_BASE
  anpr.flags := 0;
  pr := @rect;
  GetMem( ptext, all * sizeof(pchar));
  for i := 0 to all-1 do
    GetMem( ptext[i], 20 * sizeof( char ) );
    RectArea.x := 0; RectArea.y := 0;
    RectArea.width := 640; RectArea.height := 480;
    r := anprPlateMemoryRect( p, s, RectArea, anpr, @all, pr, @ptext[0] );
  end;
  FreeMem(p);
```


5. Recommendations for use

REQUIREMENTS RECOGNITION ALGORITHM

Vehicle identification number must be placed in the frame in its entirety.

Vertical angle of the video camera is not more than 40°.

The inclination angle depth - not more than 30°.

Images must be clear and not blurry.

The character size for reliable recognition must be at least 14 pixels in height.

The distance to the car camera is determined by the focal length set on the camera and must meet the requirements for the height of the characters and the clarity of the image. This can be 3 meters and 7 meters, depending on the camera used and its settings.

COMPARISON OF PERFORMANCE FOR DIFFERENT VERSIONS

Computer: AMD-FX (tm) -6100 Six-cores 3.3GHz (on one core), 8GB of RAM, Windows 7 64 in ANPR_RUSSIAN_BASE mode

The average image processing is 640x480: 0.025s (PRO), 0.8c (FREE)

The average image processing is 1920x1080: 0.26s (PRO), 9c (FREE)

RECOMMENDATIONS FOR RECOGNITION PARAMETERS

1. For parking entry automation systems on the list of white numbers.

Detect_Mode = ANPR_DETECTCOMPLEXMODE;

flags = DEBUG_RECOGNITION_MODE;

In order to observe the maximum number of car plates, combine the results and then screen out the list of white numbers. Car plate with one mistakenly recognized symbol or even unrecognized, you can, on the basis of the coincidence of the remaining symbols with the number from the white list (you compare yourself) be attributed to the correctly recognized one.

2. For systems of reliable number recognition in the absence of a whitelist.

Detect_Mode = ANPR_DETECTMODE2 | ANPR_DETECTMODE3;

flags = NO_LOW_RELIABILITY;

For the minimum number of false positives.

6. How to use the demo version of the iANPR SDK

The demo version is not intended for recognition in real time, but only allows you to estimate the recognition functions of car numbers. The recognition technology is implemented in the library iANPR_vc14_x86.dll is significantly slower (approximately 25-35 times depending on the recognition mode and processor) to limit usage. You can check the work of the SDK using the ready-made examples. For example, image.exe.

```
image 0 image.bmp > image.txt
```

After working out in image.txt, the results of the recognition algorithm work (the same is achieved by the bat-file run_read_image_bmp.bat). If you want to recognize a group of files in a directory, then you can use the bat-file run_read_in_dir.bat. For example, like this:

```
run_read_in_dir.bat c:\im
```

Recognition results will be displayed in the console.

To test the recognition in the stream, which should increase the reliability of recognition by analyzing not one frame but a sequence, you need to use the capture example. By default, the example works with a camera connected to the computer, but here it should be remembered that as the speed is significantly slowed down, the reliability will even go down, but not rise. So if you want to test the real reliability of the recognition, then write the video to the file first, and then call the example with the parameter, for example:

```
capture.exe 0 100media\AMBA2826.MOV
```

Conclusion

The library is constantly developing and improving, including in terms of recognition quality.

Recognition of car numbers from other countries will be added.

Additional modules with auxiliary functions are planned.

When errors occur, incorrect recognition, etc., please contact support@intbusoft.com, specifying the algorithm settings that you are using and attaching an image with which you are having problems.