User Manual - JIDOSHA

Automatic License Plate Recognition Software Library

Version 1.5.18

Last update: 2015/07

---

1 Overview

1.1 General Conditions

The data and information contained in this document may not be altered without written permission by GAUSSIAN Inteligência Computacional Ltda. No part of this document may be reproduced or transmitted for whatever purpose, whether by electronic or physical means.

1.2 Software license

The software and this document are protected by author rights. On installing the software, you are agreeing with the conditions of the license contract.

---

2 Introduction

This document, User Manual - JIDOSHA, details the Application Programming Interface of the automatic license plate recognition library called JIDOSHA and the conditions for its correct usage.

2.1 Objective

The main feature of the software library JIDOSHA consists in recognizing vehicle license plates from images. Its main application is in electronic traffic surveillance, a scenario for which JIDOSHA was specifically created and in which it achieves excellent performance. It is also possible to use the library in any kind of application for controlling and managing vehicles.

Due to its high accuracy, JIDOSHA is the ideal tool for those who need to obtain license plate information in an automatic fashion, without external intervention, through image analysis methods.

2.2 Usage Conditions

The software library JIDOSHA was created to work along with a USB hardkey (security key) which accompanies the library. In other words, for correct functioning the hardkey must be connected to a USB port in the computer in which the software license is being used.

There are two hardkey versions: demonstration and full. The demonstration version has an expiry date, while the full one does not. When the expiry date is reached, the library will automatically start returning empty plates. If your demonstration has expired and you would like to purchase a license or extend the demonstration period, please contact GAUSSIAN Inteligência Computacional (contato@gaussian.com.br).

This version of JIDOSHA is compatible with Windows under the programming platforms Delphi, C++ Builder 6, Visual C++, .NET 2.0 or above, and Java; and with Linux under C (gcc), C++ (g++) and Java.

2.3 Installation

2.3.1 Windows

To install the software library JIDOSHA, simply decompress the supplied SDK (Software Development Kit) file, which should contain DLLs and sample source code, and copy the DLLs to your project folder. If a password is asked upon trying to decompress the file, the default password is "jidosha".

To test the library and the hardkey, plug the hardkey (Windows will automatically install a generic driver when the hardkey is first connected to each USB port), open a command prompt window (cmd.exe) and, from inside the folder where the SDK was decompressed, run "jidoshaSample.exe 0 Foto_teste.jpg". In some older versions of the library the executable sample was called "jidoshapc.exe".

If all is well, this program should return, along with other information, the word "autorizado" (authorized) and the license plate present in the supplied test image. If the word "autorizado" does not appear, try removing the hardkey and plugging it again. If case that does not solve the problem, your hardkey might have expired, or it might have some other problem. In that case, please contact GAUSSIAN Inteligência Computacional.

2.3.2 Linux

For its correct functioning, the permissions for the USB hardkey must be changed. Add the following line:

```
ATTRS{idVendor}=="0403", ATTRS{idProduct}=="c580", MODE="0666"
```

to the end of the following file corresponding to your Linux distribution:

```
Centos 5.2/5.4:          /etc/udev/rules.d/50-udev.rules
Centos 6.0 onwards:      /lib/udev/rules.d/50-udev-default.rules
Ubuntu 7.10:             /etc/udev/rules.d/40-permissions.rules
Ubuntu 8.04/8.10:        /etc/udev/rules.d/40-basic-permissions.rules
Ubuntu 9.04 onwards:     /lib/udev/rules.d/50-udev-default.rules
openSUSE 11.2 onwards:   /lib/udev/rules.d/50-udev-default.rules
```

In Centos and openSUSE, it is also necessary to add the user which will run the OCR to the "lock" group, since JIDOSHA creates a lock file in `/var/lock`.

If you are using Debian, add the following lines:

```
SUBSYSTEM=="usb_device", MODE="0666"
SUBSYSTEM=="usb", ENV{DEVTYPE}=="usb_device", MODE="0666"
```

to the end of the following file:

```
Debian 6.0 onwards:      /lib/udev/rules.d/91-permissions.rules
```

For instructions on how to enable the hardkey for other Linux distributions, please contact GAUSSIAN Inteligência Computacional.

After changing the USB permissions, plug in the hardkey and run `./jidoshaSample 0 Foto_teste.jpg` from the SDK directory, which should output the library version, the hardkey serial number, the hardkey status (authorized or not authorized) and, finally, the license plate present in the test image.

---

3 API JIDOSHA (C/C++)

The API (Application Programming Interface) of the JIDOSHA library is written in C, which allows it to be used from practically any programming environment. The SDK includes wrapper libraries to simplify use of JIDOSHA from .NET (C# and VB.NET), Java and Delphi. These wrappers simply wrap the calls to the library functions, performing any necessary conversion of parameters and results.

The entire C API can be included with a single header file, jidoshaCore.h, whose contents are display below. A detailed description is also shown.

Beginning of file jidoshaCore.h

```
#define JIDOSHA_TIPO_PLACA_CARRO 1 /* recognize only non-motorcycle plates */
#define JIDOSHA_TIPO_PLACA_MOTO  2 /* recognize only motorcycle plates */
#define JIDOSHA_TIPO_PLACA_AMBOS 3 /* recognize all plates */

enum jidoshaError {
    JIDOSHA_SUCCESS = 0,
    JIDOSHA_ERROR_HARDKEY_NOT_FOUND,
    JIDOSHA_ERROR_HARDKEY_NOT_AUTHORIZED,
    JIDOSHA_ERROR_FILE_NOT_FOUND,
    JIDOSHA_ERROR_INVALID_IMAGE,
    JIDOSHA_ERROR_INVALID_IMAGE_TYPE,
    JIDOSHA_ERROR_INVALID_PROPERTY,
    JIDOSHA_ERROR_OTHER = 999,
};

/* OCR parameters */
typedef struct JidoshaConfig
{
    int tipoPlaca;  /* indicates the type of plate the OCR should recognize
                       use JIDOSHA_TIPO_PLACA_CARRO,
                           JIDOSHA_TIPO_PLACA_MOTO,
                       or  JIDOSHA_TIPO_PLACA_AMBOS */
    int timeout;    /* timeout in milliseconds */
} JidoshaConfig;

/* OCR result */
typedef struct Reconhecimento
{
    char placa[8]; /* null-terminated, 7 character plate, or an empty string
                      if the plate was not found */
    double probabilities[7]; /* values from 0.0 to 1.0 indicating the
                                reliability of each recognized character */
    int xText;      /* left coordinate of the plate text rectangle */
    int yText;      /* top coordinate of the plate text rectangle */
    int widthText;  /* width of the plate text rectangle */
    int heightText; /* height of the plate text rectangle */
    int textColor;  /* text color, 0 - dark, 1 - light */
    int isMotorcycle; /* 0 - non-motorcycle plate, 1 - motorcycle plate */

} Reconhecimento;

/* run OCR on a buffer containing a coded image (JPG, BMP etc)
   returns an empty plate if no hardkey was found or if it is unauthorized */
int lePlacaFromMemory(const unsigned char* stream, int n, JidoshaConfig* config, Reconhecimento* rec);

/* run OCR on an image file
   returns an empty plate if no hardkey was found or if it is unauthorized */
int lePlaca(const char* filename, JidoshaConfig* config, Reconhecimento* rec);

/* library version */
int getVersion(int* major, int* minor, int* release);

/* hardkey serial number */
int getHardkeySerial(unsigned long* serial);
```

```
/* hardkey state
   state == 0   -> unauthorized
   state == 1   -> authorized
   retorno == 0 -> hardkey found
   retorno == 1 -> hardkey not found */
JIDOSHACORE_API int getHardkeyState(int* state);
```

End of file jidoshaCore.h

3.1 struct JidoshaConfig

3.1.1 Description

This structure is used to configure the library's behavior when running license plate recognition.

3.1.2 Members

`int tipoPlaca`: indicates the type of plate which the OCR must search. It must be among the following values:

JIDOSHA_TIPO_PLACA_CARRO: only car license plates are searched, where "car" means "non-motorcycle", that is, it includes cars, trucks, buses etc.

JIDOSHA_TIPO_PLACA_MOTO: only motorcycle license plates will be searched

JIDOSHA_TIPO_PLACA_AMBOS: both motorcycle and non-motorcycle license plates will be searched.

`int timeout`: indicate the maximum time that the plate recognition should take, in milliseconds. A value of zero indicates no timeout. A non-zero timeout is useful in keeping a low average processing time. This value should be determined base on the image resolution and the CPU used.

3.2 struct Reconhecimento

3.2.1 Description

This structure is used to store results from license plate recognition, including: the plate characters, the reliability of each character, and the coordinates of the plate in the image.

3.2.2 Members

`char placa[8]`: null-terminated, 7 character plate, or an empty string if the plate was not found.

`double probabilities[7]`: values from 0.0 to 1.0 indicating the reliability of each recognized character.

`int xText`: left coordinate of the plate text rectangle.

`int yText`: top coordinate of the plate text rectangle.

`int widthText`: width of the plate text rectangle.

`int heightText`: height of the plate text rectangle

`int textColor`: text color, 0 - dark, 1 - light.

`int isMotorcycle`: 0 – non-motorcycle plate, 1 - motorcycle plate.

3.3 lePlaca

3.3.1 Function Prototype

```
int lePlaca(const char* filename, JidoshaConfig* config, Reconhecimento* rec);
```

3.3.2 Description

Recognizes the license plate and stores it in a `Reconhecimento` object. The image (JPG, BMP etc.) should be passed as a filepath pointing to an image file. If no plate is found, or if the hardkey is unauthorized or could not be found, the `Reconhecimento` object will contain an empty string as the plate.

3.3.3 Return

Error code: 0 (zero) in case of success, otherwise a non-zero number.

3.4 lePlacaFromMemory

3.4.1 Function Prototype

```
int lePlacaFromMemory(const unsigned char* stream, int n, JidoshaConfig* config, Reconhecimento*rec);
```

3.4.2 Description

Recognizes the license plate and stores it in a `Reconhecimento` object. The image (JPG, BMP etc.) should be passed as a byte array, where the number of bytes should be indicated by parameter `n`. If no plate is found, or if the hardkey is unauthorized or could not be found, the `Reconhecimento` object will contain an empty string as the plate.

3.4.3 Return

Error code: 0 (zero) in case of success, otherwise a non-zero number.

3.5 getVersion

3.5.1 Function Prototype

```
int getVersion(int* major, int* minor, int* release);
```

3.5.2 Description

Used to get the library version, in the format major.minor.release.

3.5.3 Return

Always returns 0 (zero).

3.6 getHardkeySerial

3.6.1 Function Prototype

```
int getHardkeySerial(unsigned long* serial);
```

3.6.2 Description

Used to get the hardkey serial number.

3.6.3 Return

Returns 0 in case of success, 1 if the hardkey was not found.

3.7 getHardkeyState

3.7.1 Function Prototype

```
int getHardkeyState(int* state);
```

3.7.2 Description

Used to get the hardkey state. If `state` is equal to 0, the hardkey is unauhorized; if state is equal to 1, the hardkey is authorized.

3.7.3 Return

Return 0 in case of success, 1 if the hardkey was not found.

---

4 JIDOSHA API (C# / VB.NET)

The library's .NET API contains three overloaded functions, which make it easy to apply license plate recognition to an image from any of three sources: an array of bytes containing a coded image (JPG, BMP etc.), and object of type `Image`, or a filepath. All of these functions need an object of type`JidoshaConfig`, which is used to configure the behavior of the library.

4.1 reconhecePlaca 1

4.1.1 Function Prototype

```
Reconhecimento reconhecePlaca(byte[] array, JidoshaConfig config)
```

4.1.2 Description

Returns a `Reconhecimento` object which represents the result of the license plate recognition process. The image (JPG, BMP etc.) should be passed as a byte array.

4.1.3 Return

`Reconhecimento` object containing the license plate text string, an array of doubles containing the reliability of each character, the coordinates of the text rectangle, the text color (dark or light), and whether the license plate is that of a motorcycle. If no plate is found, or if the hardkey is unauthorized or could not be found, the `Reconhecimento` object will contain an empty string as the plate.

4.2 reconhecePlaca 2

4.2.1 Function Prototype

```
Reconhecimento reconhecePlaca(Image image, JidoshaConfig config)
```

4.2.2 Description

Returns a `Reconhecimento` object which represents the result of the license plate recognition process. The image should be passed as an `Image` object.

4.2.3 Return

`Reconhecimento` object containing the license plate text string, an array of doubles containing the reliability of each character, the coordinates of the text rectangle, the text color (dark or light), and whether the license plate is that of a motorcycle. If no plate is found, or if the hardkey is unauthorized or could not be found, the `Reconhecimento` object will contain an empty string as the plate.

4.3 reconhecePlaca 3

4.3.1 Function Prototype

```
Reconhecimento reconhecePlaca(string filename, JidoshaConfig config)
```

4.3.2 Description

Returns a `Reconhecimento` object which represents the result of the license plate recognition process. The image should be passed as a filepath pointing to the image file.

4.3.3 Return

`Reconhecimento` object containing the license plate text string, an array of doubles containing the reliability of each character, the coordinates of the text rectangle, the text color (dark or light), and whether the license plate is that of a motorcycle. If no plate is found, or if the hardkey is unauthorized or could not be found, the `Reconhecimento` object will contain an empty string as the plate.

4.4 getVersion

4.4.1 Function Prototype

```
String getVersionString()
```

4.4.2 Description

Used to get the library version, in the format major.minor.release.

4.4.3 Return

Returns a string formatted with the library version in the format major.minor.release.

4.5 getHardkeySerial

4.5.1 Function Prototype

```
int getHardkeySerial()
```

4.5.2 Description

Used to get the hardkey serial number.

4.5.3 Return

Returns the hardkey serial number.

4.6 getHardkeyState

4.6.1 Function Prototype

```
int getHardkeyState()
```

4.6.2 Description

Used to get the hardkey state. If the return value is equal to 0, the hardkey is unauhorized; if the return value is

equal to 1, the hardkey is authorized.

4.6.3 Return

Returns the hardkey state, according to the description above.

---

5 JIDOSHA API (Delphi)

5.1 reconhecePlaca

5.1.1 Function Prototype

```
function reconhecePlaca(filename: String; config: JidoshaConfig) : Reconhecimento;
```

5.1.2 Description

Returns a `Reconhecimento` object which represents the result of the license plate recognition process. The image should be passed as a filepath pointing to the image file.

5.1.3 Return

`Reconhecimento` object containing the license plate text string, an array of doubles containing the reliability of each character, the coordinates of the text rectangle, the text color (dark or light), and whether the license plate is that of a motorcycle. If no plate is found, or if the hardkey is unauthorized or could not be found, the `Reconhecimento` object will contain an empty string as the plate.

5.2 reconhecePlacaFromMemory

5.2.1 Function Prototype

```
function reconhecePlacaFromMemory(byteArray: array of byte; config: JidoshaConfig) : Reconhecimento;
```

5.2.2 Description

Returns a `Reconhecimento` object which represents the result of the license plate recognition process. The image (JPG, BMP etc.) should be passed as a byte array.

5.2.3 Return

`Reconhecimento` object containing the license plate text string, an array of doubles containing the reliability of each character, the coordinates of the text rectangle, the text color (dark or light), and whether the license plate is that of a motorcycle. If no plate is found, or if the hardkey is unauthorized or could not be found, the `Reconhecimento` object will contain an empty string as the plate.

---

6 JIDOSHA API (Java)

6.1 reconhecePlaca

6.1.1 Function Prototype

```
public static native Reconhecimento reconhecePlaca(String filename, JidoshaConfig config);
```

6.1.2 Description

Returns a `Reconhecimento` object which represents the result of the license plate recognition process. The image should be passed as a filepath pointing to the image file.

6.1.3 Return

`Reconhecimento` object containing the license plate text string, an array of doubles containing the reliability of each character, the coordinates of the text rectangle, the text color (dark or light), and whether the license plate is that of a motorcycle. If no plate is found, or if the hardkey is unauthorized or could not be found, the `Reconhecimento` object will contain an empty string as the plate.

6.2 reconhecePlacaFromMemory

6.2.1 Function Prototype

```
public static native Reconhecimento reconhecePlacaFromMemory(byte[] buf, JidoshaConfig config);
```

6.2.2 Description

Returns a `Reconhecimento` object which represents the result of the license plate recognition process. The image (JPG, BMP etc.) should be passed as a byte array.

6.2.3 Return

`Reconhecimento` object containing the license plate text string, an array of doubles containing the reliability of each character, the coordinates of the text rectangle, the text color (dark or light), and whether the license plate is that of a motorcycle. If no plate is found, or if the hardkey is unauthorized or could not be found, the `Reconhecimento` object will contain an empty string as the plate.

---

7 Usage examples

7.1 C/C++ Example

```c
#include <stdio.h>
#include "jidoshaCore.h"

int main(int argc, char* argv[])
{
    Reconhecimento rec;
    JidoshaConfig config;
    config.tipoPlaca = JIDOSHA_TIPO_PLACA_AMBOS;
    config.timeout = 1000;
    lePlaca(argv[1], &config, &rec);
    printf("plate: %s\n", rec.placa);
    return 0;
}
```

7.2 C# Example

```c#
using System;
using System.Drawing;
using jidoshaNET;

namespace JidoshaSample
{
    class JidoshaSample
    {
        static void Main(string[] args)
        {
            string filename = args[0];
            Console.WriteLine(filename);
            JidoshaConfig config = new JidoshaConfig();
            config.tipoPlaca = TipoPlaca.AMBOS;
            config.timeout = 1000;
            Reconhecimento rec = Jidosha.reconhecePlaca(filename, config);
            Console.WriteLine("plate: " + rec.placa);
        }
    }
}
```

7.3 VB.NET Example

```vb
Imports jidoshaNET

Module Module1

    Sub Main()
        Dim args() As String = Environment.GetCommandLineArgs()
        Dim filename As String = args(1)
        Dim config As JidoshaConfig = New JidoshaConfig()
        config.tipoPlaca = TipoPlaca.AMBOS
        config.timeout = 1000
        Dim rec As Reconhecimento = Jidosha.reconhecePlaca(filename, config)
        Console.WriteLine("plate: " + rec.placa)
    End Sub

End Module
```

7.4 Delphi Example

```delphi
program JidoshaDelphiSample;

{$APPTYPE CONSOLE}
```

```
uses
  SysUtils,
  jidoshaDelphi in 'jidoshaDelphi.pas';

var
    filename: String;
    rec: Reconhecimento;
    config: JidoshaConfig;
begin
    if ParamCount < 1
    then begin
      Writeln('usage: jidoshaDelphiSample.exe image.jpg');
      Exit;
    end;

    filename := ParamStr(1);
    Writeln(filename);

    config.tipoPlaca := JIDOSHA_TIPO_PLACA_AMBOS;
    config.timeout := 1000;
    rec := reconhecePlaca(filename, config);
    Writeln('plate: ', rec.placa);
end.
```

7.5 Java Example

```
import br.com.gaussian.jidosha.Jidosha;
import br.com.gaussian.jidosha.JidoshaConfig;
import br.com.gaussian.jidosha.Reconhecimento;

class JidoshaSample {
    public static void main(String args[]) throws java.io.IOException {
        JidoshaConfig config = new JidoshaConfig(JidoshaConfig.JIDOSHA_TIPO_PLACA_AMBOS, 0);
        for (int i=0; i < args.length; i++) {
            System.out.println(args[i]);
            Reconhecimento rec = Jidosha.reconhecePlaca(args[i], config);
            System.out.println("plate: " + rec.placa);
        }
    }
}
```
