

车牌识别 **SDK** 开发包使用指南

上海名图软件有限公司

版本号: 1.2.4.1

Tel: 021-61259082

www.cubicimage.com

上海名图软件有限公司

一 SDK开发包说明.....	3
1 简介.....	3
1.1. 文档用途.....	3
1.2. 背景.....	3
2 技术概要.....	3
2.1.系统需求.....	3
2.1.1. 硬件平台.....	3
2.1.2. 操作系统.....	3
2.2.模块说明.....	4
2.3.DLL调用方式.....	4
2.4. OCX调用方式.....	4
二 MulCubicPlate.Dll 动态链接库接口说明.....	6
2.1. 用户需要了解的结构体定义.....	6
2.1.1 车牌结构.....	6
2.1.2 四边形结构.....	6
2.1.3 跟踪定位车牌结构.....	7
2.2. DLL函数.....	7
三. PlateConfigure.ini 车牌识别参数设置说明.....	13
3.1. 环境光线条件, LightingMode.....	13
3.2. 可信度门限 ConThresh.....	13
3.3. MinPlateWidth MaxPlateWidth.....	13
3.4. 省份可信度门限ProvinceConfThresh.....	14
3.5. 默认省份DefaultProvince.....	14
3.6 车牌字符类型定义.....	14
四. CISImage.Dll 动态链接库接口说明.....	15
4.1. 用户需要了解的结构体定义.....	15
4.2. DLL函数.....	15
五 CISVideo.ocx组件接口说明.....	20
六 CISAVI.ocx组件接口说明.....	22
七.CubicCoil.Dll 动态链接库接口说明.....	24
7.1. 用户需要了解的结构体定义.....	24
7.2. DLL函数.....	24
八. 应用程序测试识别效果.....	26
九 技术支持.....	27

一 SDK开发包说明

1 简介

1.1. 文档用途

本文档为上海名图信名图软件司（CubicImage Software）SDK 开发包的集成指南，主要包括车牌识别模块（MulCubicPlate.dll），图片处理模块(CISImage.dll)，视频采集线圈检测模块（CISVideo.Ocx、CISAVI.Ocx、CubicCoil.dll）等功能模块。本文档的目标对象为应用程序开发人员。透过阅读本文档，相关的开发人员可以了解并掌握各核心模块使用方法及详细的开发接口。

1.2. 背景

CIS - CubicImage Software 名图软件，是一家致力于计算机视觉和图像处理科技产品研发和销售的软件公司。她于2005年初创立于上海浦东软件园。其核心技术的研发历史可追溯至1998年，当其创立者们开始进入这个引人入胜的领域时。多年来，名图软件掌握了一系列核心技术，主要涉及计算机视觉、图像处理、模式识别和虚拟现实。本开发包所提供的核心模块经由系统集成商嵌入工程应用管理系统中，本公司可为集成商提供应用完整的解决方案和详细的技术支持，这样集成商可以轻易的把其嵌入开发系统而不必深入了解车牌识别技术的细节，从而把主要精力投入于商务和系统整体解决方案。

2 技术概要

2.1. 系统需求

2.1.1. 硬件平台

- 。英特尔奔3 800MHZ或更高

- 。512MB DDR内存

2.1.2. 操作系统

- 。Microsoft Windows NT/XP

- 。Linux/Unix

2. 2. 模块说明

- (1) MulCubicPlateSdk(车牌识别模块 SDK):提供两种方式,即 OCX 和 DLL 两种方式
- (2) CubicImageSdk(图象处理模块 SDK):提供两种方式,即 OCX 和 DLL 两种方式
- (3) CubicVideoSdk(视频处理模块 SDK):提供两种方式,即 OCX 和 DLL 两种方式
- (4) 为了方便用户将以上各个模块集成进入自己的应用系统,提供各个模块的调用实例,用户可参考 Sample 中的原代码,进行系统集成。

2. 3. DLL调用方式

1. C++:以 VC.NET 2003 为例,将 MulCubicPlate.dll(车牌识别模块)、CISImage.dll(图象处理模块(可选))、PlateConfigure.ini 拷入您的应用系统工程中,然后将.h 和 lib 文件复制并添加到工程中,在项目->项目属性页->链接器->常规,添加 lib 文件的路径,在项目->项目属性页->链接器->输入中添加 CubicPlate.lib、CISImage.lib 项、编译工程。
2. C#:以 VC#.NET2003 编译环境,静态调用为例,将 MulCubicPlate.dll(车牌识别模块)、CISImage.dll(图象处理模块(可选))、PlateConfigure.ini 拷入您的应用系统工程中,静态申明函数接口,编译工程.函数接口申明参考 sdk\sample\基于图像文件的识别实例\C#例子。
3. VB:以 VB.NET2003 编译环境,静态调用为例,将 MulCubicPlate.dll(车牌识别模块)、CISImage.dll(图象处理模块(可选))、PlateConfigure.ini 拷入您的应用系统工程中,静态申明函数接口,编译工程.函数接口申明参考 sdk\sample\基于图像文件的识别实例\VB 例子。
4. Dephi:以 depi7,静态调用为例,将 MulCubicPlate.dll(车牌识别模块)、CISImage.dll(图象处理模块(可选))、PlateConfigure.ini 拷入您的应用系统工程中,静态申明函数接口,编译工程.函数接口申明参考 sdk\sample\基于图像文件的识别实例\dephi 例子。

2. 4. OCX调用方式

1. C++:以 VC.NET 2003 为例,将 MulCubicPlate.ocx、CISImage.ocx(可选)、CISVideo.ocx(可选)、PlateConfigure.ini 拷入您的应用系统工程中,然后将 *.i.c、.h 文件复制并添加进工程.编译工程。
2. 以 VC#.NET 2003 为例,将 MulCubicPlate.ocx、CISImage.ocx(可选)、CISVideo.ocx(可选)、PlateConfigure.ini 拷入您的应用系统工程,然后项目->添加引用->将 ocx 添加进入工程,添加引用,编译工程。具体函数的调用参考 SDK\sample\基于视频实时识别实例\Video(C#)
3. VB:同上,参考 SDK\sample\基于视频实时识别实例\Video(vb.net)。
4. Dephi:以 depi7 为例,将 MulCubicPlate.ocx、CISImage.ocx(可选)、CISVideo.ocx(可选)、PlateConfigure.ini 拷入您的应用系统工程,然后 project->import type library->add(将 OCX 添加进 library),Create unit,引用单元,具体参考 SDK\sample\基于视频实时识别实例\Video(dephi)。

CISVideo.ocx 注：本捕获类，要求视频采集卡支持微软的 directshow 组件.

CISAVI.OCX 注：本捕获类，要求视频文件装好解码器,并要求支持 directshow 组件

上海名图软件有限公司

二 MulCubicPlate.Dll 动态链接库接口说明

2.1. 用户需要了解的结构体定义

2.1.1 车牌结构

/*车牌结构：车牌号，车牌颜色，车牌字符位置等等*/

//识别函数出口参数结构

typedef struct CISPlateStruct

```
{  
    CHAR   PlateChar[20];           /*对应7个车牌字符,注意汉字是宽字符*/  
    INT     PlateColor;              /* 0 – 蓝牌, 1 – 黄牌, 2 –白牌, 3 – 黑牌*/  
    FLOAT   PlateCharConf[7];       /*对应7个车牌的可信度*/  
    INT     PlateRect[4];            /*对应表示车牌位置左上角x,左上角y,右下角x,右下角y*/  
    INT     PlateCharPos[7][2];     /*对应7个字符中的每个字符的x,y坐标*/  
    INT     Contrast;                /* 车牌对比度 */  
}CISPlateStruct;
```

2.1.2 四边形结构

/* 四边形结构：其四个顶点以任意一个为起点按顺序排列（逆时针、顺时针均可） */

//任意四边形，可通过设置该四边形（CISSetImageROI）来限制识别区域，与避免识别一些伪车牌

typedef struct CISQuadRangle

```
{  
    int x1;  
    int y1;  
    int x2;  
    int y2;  
    int x3;  
    int y3;  
    int x4;  
    int y4;  
}CISQuadRangle;
```

2.1.3 跟踪定位车牌结构

```
typedef struct CISPlateTrackingStruct
{
    int PlateColor;           /* 0 - 蓝牌或白牌, 1 - 黄牌或黑牌 */
    float PlateLocatingScore; /* 车牌跟踪的可信度 */
    int PlateRect[4];         /* 对应表示车牌位置左上角x, 左上角y, 右下角x, 右下角y */
    int nCentX;               /* 车牌中心x 座标 */
    int nCentY;               /* 车牌中心y 座标 */
    int Contrast;             /* 车牌前后景的对比度 */
    int Reserved[10];
} CISPlateTrackingStruct;
```

2.2. DLL函数

1) 函数 CISDllInit

原型 `BOOL __stdcall CISDllInit ()`

说明 初始化函数, 车牌识别动态链接库MulCubicPlate.dll使用前需要初始化, 初始参数需要从一个配置文件读取, 如 PlateConfig.ini。该初始化应在其被集成的应用程序初始化部分完成, 比如VC中的OnInitDialog, delphi的FormCreate, C#的InitializeComponent函数中。

范例 `BOOL bres = CISDllInit ();`

2) 函数 CISDllRelease

原型 `void __stdcall CISDllRelease ()`

说明 释放所有占用的内存资源, 该函数应在在其被集成的应用程序结束时完成, 如VC的OnCancel, delphi中的FormDestroy, C#中的Dispose函数中。

范例 `CISDllRelease ();`

3) 函数 CISProcessFrame

原型 `INT __stdcall CISProcessFrame (char* pImageData,
int iImgWidth,
int iImgHeight,
int iImgDepth,
int iImgChannel,
CISPlateStruct* plateStruct
)`

说明 处理图像数据,提取车牌信息,多车牌识别SDK将各个参数送入到DLL中,DLL把车牌号,车牌颜色,车牌可信度,车牌位置,车牌的x,y坐标等信息存入到CISPlateStruct结构中。

输入参数 Char *pImageData 图象数据指针,指图象或者视频的一帧在内存中的数据区地址

Int iImgWidth 图象宽度

Int iImgHeight 图象高度

Int iImgDepth 图象深度(指表示像素单色所用的比特数,通常为8bits)

Int iImageChannel 图象通道数(黑白图象为1,彩色图象为3)

输出参数 INT 返回参数 标示图像中的车牌的个数

plateStruct 存储车牌的结构

范例 INT iPlateNum;

CISPlateStruct pPlate[20];

CISImageStruct *pImage = CISLoadImage("E:\\001.jpg");

iPlateNum = CISProcessFrame (pImage->data,pImage->width,
pImage->height,8,3,pPlate);

if(iPlateNum > 0)

{

For(int i = 0; i < iPlateNum; i++)

{

/*提取车牌号码*/

Char *pPlateNum = pPlate[i]. PlateChar;

/*提取车牌颜色*/

int nColor = pPlate[i]. PlateColor;

}

}

/*为防止内存上涨,CISLoadImage成功以后要释放图象内存*/

If(pImage != NULL)

{

CISReleaseImage(&pImage);

}

4) 函数 CISSetLightingMode

原型 void __stdcall CISSetLightingMode (int iLightingMode)

说明 设置工作光照模式

输入参数 int iLightingMode 光照模式

光照模式列表: 0-100 , 设置可以参考CISPlateStruct中的Contrast的稍微小点。

输出参数 无

范例 CISSetLightingMode(5),详细参数配置请看下一章。

5) 函数 CissSetPlateSizeMode

原型 void __stdcall CissSetPlateSizeMode (int iPlateSearchMode)

说明 设置车牌尺寸模式

输入参数 int iPlateSearchMode 车牌搜索尺寸模式

车牌尺寸模式列表:

0 — 大尺寸车牌搜索模式

1 — 中尺寸车牌搜索模式

2 — 小尺寸车牌搜索模式

输出参数: 无

范例 CissSetPlateSizeMode(1);

6) 函数 CissSetFieldMode

原型 void __stdcall CissSetFieldMode (int iFieldMode)

说明 设置工作图像帧 / 场模式

输入参数 int iFieldMode 工作图像帧 / 场模式

工作图像帧 / 场模式列表:

0 — 全场模式 (或帧模式)

1 — 半场模式

输出参数 无

范例 CissSetFieldMode (0);

7) 函数 CissSetImageROI

原型 void __stdcall CissSetImageROI(int iX,int iY,int iWidth,int iHeight)

说明 设置车牌搜索矩形区域

输入参数 Int iX 矩形左上角X坐标

Int iY 矩形左上角Y坐标

Int iWidth 矩形宽度

Int iHeight 矩形高度

输出参数 无

范例 CissSetImageROI(200,200,400,400);

8) 函数 CissResetImageROI

原型 void __stdcall CissResetImageROI(void)

说明 设置车牌搜索区域为整个图象区域

范例 CissResetImageROI();

9) 函数 CissSetQuadRangleROI

原型 void __stdcall CissSetQuadRangleROI(int nQuadRangleNum,

CISQuadRangle *pQuadRangle);

说明 设置车牌搜索任意四边形区域

输入参数 nQuadRangleNum 设置的四边形区域数目

pQuadRangle 指向一个存储四边形结构的数组

输出参数 无

范例 CISQuadRangle quad1 = {116, 654, 386, 672, 386, 1050, 46, 1024};
CISQuadRangle quad2 = {598, 752, 886, 776, 878, 916, 614, 904};
CISQuadRangle quads[2] = {quad1, quad2};
int nQuadNum = 2;
CISSetQuadRangleROI(nQuadNum, quads);

10) 函数 CISClearQuadRangleROI

原型 void __stdcall CISClearQuadRangleROI(void);

说明 取消四边形搜索ROI设置

范例 CISClearQuadRangleROI();

11) 函数 CISPlateTracking

原型 int __stdcall CISPlateTracking(char* pImageData,

int iImgWidth,

int iImgHeight,

int iImgDepth,

int iImgChannel,

CISPlateTrackingStruct *pPlateTrackingStruct)

说明 搜索定位车牌,多车牌识别SDK将各个参数送入到DLL中,DLL把车牌位置,车牌颜色,可信度,车牌中心坐标等信息存入到CISPlateTrackingStruct结构中.

输入参数 Char *pImageData 图象数据指针,指图象或者视频的一帧在内存中的数据区地址

Int iImgWidth 图象宽度

Int iImgHeight 图象高度

Int iImgDepth 图象深度(指表示像素单色所用的比特数,通常为8bits)

Int iImageChannel 图象通道数(黑白图象为1,彩色图象为3)

输出参数 INT 返回参数 标示图像中的车牌的个数

pPlateTrackingStruct 存储车牌信息的结构

范例 INT iPlateNum;

CISPlateTrackingStruct pPlate[20];

CISImageStruct *pImage = CISLoadImage("E:\\001.jpg");

iPlateNum = CISPlateTracking (pImage->data,pImage->width,
pImage->height,8,3,pPlate);

```
if(iPlateNum > 0)
{
    For(int i = 0; i < iPlateNum; i++)
    {
        /*提取车牌颜色*/
        int nColor = pPlate[i]. PlateColor;
        /*提取车牌中心点*/
        int nCenX = pPlate[i]. nCentX;
        int nCentY = pPlate[i]. nCentY;
    }
}

/*为防止内存上涨,CISLoadImage成功以后要释放图象内存*/
If(pImage != NULL)
{
    CISReleaseImage(&pImage);
}
```

12) 函数 CISPlateTrackingToRecognition

原型 int __stdcall CISPlateTrackingToRecognition (char* pImageData,
int iImgWidth,
int iImgHeight,
int iImgDepth,
int iImgChannel,
int nTrackedPlateNumber,
CISPlateTrackingStruct *pPlateTrackingStruct,
CISPlateStruct* plateStruct)

说明 根据搜索定位到的车牌位置信息, 进一步识别出车牌字符等信息 将最终结果传递至CISPlateStruct结构

输入参数 char *pImageData 图象数据指针,指图象在内存中的数据区地址

int iImgWidth 图象宽度

int iImgHeight 图象高度

int iImgDepth 图象深度(指表示像素单色所用的比特数,通常为8bits)

int iImgChannel 图象通道数(黑白图象为1,彩色图象为3)

int nTrackedPlateNumber 搜索定位得到的车牌数目

CISPlateTrackingStruct *pPlateTrackingStruct 搜索定位得到的车牌结构数组

输出参数 INT 返回参数 标示图像中的车牌的个数

plateStruct存储车牌信息的结构

范例:

```
INT iPlateNum;
CISPlateStruct pPlate[20];
.....CISPlateTracking 获取 nTrackedPlateNumber 和 pPlateTrackingStruct
iPlateNum = CISPlateTrackingToRecognition ( pImageData,
                                           iImgWidth,
                                           iImgHeight,
                                           iImgDepth,
                                           iImgChannel,
                                           nTrackedPlateNumber,
                                           pPlateTrackingStruct,
                                           plateStruct);

if(iPlateNum > 0)
{
    for(int i = 0; i < iPlateNum; i++)
    {
        /*提取车牌号码*/
        char *pPlateNum = pPlate[i]. PlateChar;
        /*提取车牌颜色*/
        int nColor = pPlate[i]. PlateColor;
    }
}
```

13) 函数 CISSetPlateTrackingConfThresh

原型 void __stdcall CISSetPlateTrackingConfThresh(float fConfThresh)

说明 设置车牌跟踪定位可信度门限 (0~1) 值越小 可信度越低 跟踪到的车牌越多 值越大 可信度越高 跟踪越严格

输入参数 float fConfThresh 可信度门限值 (0~1)

输出参数 无

范例 CISSetPlateTrackingConfThresh (0.45) ;

三. PlateConfigure.ini 车牌识别参数设置说明

针对目前各个工程商的工程配置环境的不同和光线的强弱,我们通过配置不同的识别参数来达到各个工程商的要求。

3.1. 环境光线条件, LightingMode

此参数可以由 CISPlateStruct 中的 Contrast 的值计算获取,如:取同一组图片陆续获取一些 Contrast 求取其平均值,设置 LightingMode 稍微低于此平均值。

此参数一般白天为 20 上下,晚上为 10 上下,参数越高,识别算法越简单,识别速度越快,但同时识别效果会降低。反之,参数越低,识别算法越复杂,识别速度越慢,但识别效果会越好。工程商按照自己的实际使用效果来确定此值。该参数可以由 CISSetLightingMode 来动态设置。

3.2. 整牌可信度门限 ConThresh

- (1)车牌识别会存在着假目标,比如对栏杆识别等。
- (2)提高可信度可以减少误识别率,但是同时也可能会降低识别率。
- (3)降低可信度可能会增加误识别率,同时也增加识别率。
- (4)一般可信度一般都是 0.4-0.5 左右,取值在 0-1 之间。

3.3 车牌定位跟踪可信度门限 PlateTrackingConfThresh

- 1)车牌跟踪定位中会存在着假目标,比如对栏杆识别等。
- (2)提高可信度可以减少误定位率,但是同时也可能会降低定位率。
- (3)降低可信度可能会增加误定位率,同时也增加定位率。
- (4)一般可信度一般都是 0.4-0.5 左右,取值在 0-1 之间。

3.4. MinPlateWidth MaxPlateWidth

车牌搜索的最小最大像素限制,如果出现将太大或太小的目标 如车顶或栅栏等 误认为车牌时,可适当调整此值

3.5. 省份可信度门限ProvinceConfThresh

汉字可信度过滤,范围为 0~1, 一般设置为 0.55, 如果识别结果省份可信度小于此值, 则在多个或一个默认省份中, 选取最可能的省份。如果识别结果省份可信度大于此值, 则在多个通用省份中选取最可能的省份

3.6 通用省份GeneralProvince

一般设置多个, 识别结果省份可信度, 高于可信度门限时, 选取其中最可能的省份。

如 GeneralProvince = 京津冀晋蒙辽吉黑沪苏浙皖闽赣鲁豫鄂湘粤川陕甘新军空海北
沈兰济南广成

3.7 默认省份 DefaultProvince

可设置一个 比如本省省份, 或设置多个, 比如邻省省份。识别结果省份可信度小于此值, 则在多个或一个默认省份中, 选取最可能的省份

如 DefaultProvince = 京津冀晋

3.8 车牌字符类型定义

对应车牌 7 个字符, C:汉字 E:英文 D:数字 M:英文或数字。软件将按照用户所设置的类型返回车牌号码, 如用户设置 CEMMMMD, 返回的车牌号码是沪 AB5678。

四. CISImage.Dll 动态链接库接口说明

4. 1. 用户需要了解的结构体定义

图像内存结构

```
typedef struct CISImageStruct
{
    int width;//图像宽度
    int height;//图像高度
    int depth;//图象深度(指表示像素单色所用的比特数,通常为8bits)
    int channels;//图象通道数(黑白图象为1,彩色图象为3)
    char* data;//图象数据指针,指图象或者视频的一帧在内存中的数据区地址
}CISImageStruct;
```

矩形区域结构

```
typedef struct
{
    int x;        //图象区域起始 X 坐标
    int y;        //图象区域起始 Y 坐标
    int width;    //图象区域宽度
    int height;   //图象区域高度
}CISImageRect;
```

图象大小结构

```
typedef struct
{
    int width;
    int height;
}
CISImageSize;
```

4. 2. DLL函数

1) 函数 CISImageDllRelease

原型 void __stdcall CISImageDllRelease();

说明 释放所有占用的内存资源

范例 CISImageDllRelease();

2) 函数 CISLoadImage

原型 CISImageStruct* __stdcall CISLoadImage(LPSTR imageFileName)

说明 读取图像到内存

输入参数 imageFileName 图像文件路径名

输出参数 CISImageStruct* 指向 CISImageStruct 结构,存储图像信息

范例 CISImageStruct *pImage = CISLoadImage("E:\\001.jpg");

3) 函数 CISReleaseImage

原型 void __stdcall CISReleaseImage(CISImageStruct** ppImage);

说明 释放图像内存

输入参数 图像内存二级指针

输出参数 无

范例 CISImageStruct *pImage = CISLoadImage("E:\\001.jpg");

.....

CISReleaseImage(&pImage);

4) 函数 CISSaveImage

原型 BOOL __stdcall CISSaveImage(LPSTR imageFileName, char* imageData, int width, int height, int depth, int channels);

说明 保存内存图像到文件

输入参数 imageFileName 要保存到的目标文件名

imageData 图象数据指针,指图象或者视频的一帧在内存中的数据区地址

width 图像宽度

height 图像高度

depth 图象深度(指表示像素单色所用的比特数,通常为 8bits)

channels 图象通道数(黑白图象为 1,彩色图象为 3)

输出参数 若保存成功则返回 TRUE, 否则返回 FALSE.

范例 BOOL bres = CISSaveImage("E:\\002.jpg", pImage->data, pImage->width, pImage->height, 8, 3);

5) 函数 CISSetImageQuality

原型 void __stdcall CISSetImageQuality(float fQuality)

说明 设置图像存储质量参数

输入参数 fQuality (0~1) 图像存储质量参数, 由该系数决定 CISSaveImage 的压缩比 值越高, 图片质量越好, 图片所占硬盘越大

输出参数 无

范例 `CISSetImageQuality (0.45) ;`
`BOOL bres = CISSaveImage("E:\\002.jpg",pImage->data,pImage->width,`
`pImage->height,8,3);`

6) 函数 CISFlipImage

原型 `void __stdcall CISFlipImage(char *imageData,int width,int height,`
`int depth,int channels);`

说明 垂直翻转图像

输入参数 同 CISSaveImage 函数

输出参数 无

范例 `CISFlipImage (pImage->data,pImage->width,pImage->height,8,3)`

7) 函数 CISGetPixPointer

原型 `void __stdcall CISGetPixPointer(char *imageData, int width, int height, int depth,`
`int channels, int x, int y, char **pxy);`

说明 得到图像坐标第 x 行第 y 列的像素点 RGB 值

输入参数 imageData , width, height, depth, channels 同上

x,y 第 x 行第 y 列

输出参数 pxy 图像第 x 行 y 列像素 RGB 值指针

范例 `CISImageStruct *pImage = CISLoadImage("1.jpg");`
`int iRow = 1000;`
`int iCol = 800;`
`char *pPixData = NULL;`
`CISGetPixPointer(pImage->data, pImage->width,`
`pImage->height,8,3,iRow,iCol,&pPixData);`
.....

8) 函数 CISCreateImage

原型 `CISImageStruct* __stdcall CISCreateImage(int width, int height,`
`int depth, int channels)`

说明 创建一个图像空间

输入参数 width, height, depth, channels 同上

输出参数 CISImageStruct* 指向 CISImageStruct 结构, 存储图像信息

范例 `CISImageStruct* pImage = CISCreateImage(1024,728,8,3);`

9) 函数 CISCopyImage

原型 void __stdcall CISCopyImage(CISImageStruct* psourceImage, int sx, int sy,
int swidth,int sheight,CISImageStruct* pdestImage,
int dx, int dy, int dwidth,int dheight);

说明 源图像拷贝到目的图像中去, 需要拷贝的源图像区域和目的图像区域分别由
sx,sy...dx,dy,... 指定

输入参数 psourceImage pdestImage分别指向源图像和目的图像

sx sy swidth sheight 需要拷贝的源图像矩形区域左上角x,y坐标, 宽度, 高度

dx dy dwidth dheight 拷贝到目的图像后所占的矩形区域左上角x,y坐标, 宽
度, 高度。

输出参数 无

范例 CISCopyImage(psourceImage,0,0,100,100,pdestImage,100,100,400,400);

10) 函数 CISPutTextToImage

原型 void __stdcall CISPutTextToImage(CISImageStruct* pImage, char* pText, int x,
int y, int color,float scale,int thickness)

说明 将文字叠加到图像上

输入参数 pImage 指向所叠加的图像

pText 叠加字符

x,y叠加位置

color叠加文字颜色定义: 0 - 红色, 1 - 绿色, 2 - 蓝色

scale叠加文字尺寸伸缩因子, 1.0表示无伸缩

thickness叠加字符笔画宽度

范例 CISPutTextToImage(pImage,"CubicImage Software co.,Ltd",100,100,0,1.0,2);

11) 函数 CISDrawPlateRect

原型 void __stdcall CISDrawPlateRect (CISImageStruct *pImage,CISImageRect
thisImageRect, LONG color,int thickness);

说明 在图象上画框

输入参数 pImage 指向所叠加的图像

thisImageRect 框绘制的区域

color 叠加框颜色定义: 0 - 红色, 1 - 绿色, 2 - 蓝色

thickness叠加框笔画宽度

范例 CISImageRect thisImageRect;

thisImageRect.x = 10;

thisImageRect.y = 20;

thisImageRect.width = 100;

thisImageRect.height = 100;

CISDrawPlateRect (pImage, thisImageRect,0,2);

12) 函数 CISReSizeImage

原型 void __stdcall CISReSizeImage (CISImageStruct *pSourceImage,
CISImageStruct *pdestImage);

说明 图象伸缩函数

输入参数 pSourceImage 源图象数据指针

pDestImage 缩放后图象数据指针

范例 CISReSizeImage (pSourceImage, pDestImage);

上海名图软件有限公司

五 CISVideo.ocx组件接口说明

注:本开发包采集卡需要有支持微软的 directshow 的驱动

1 连接摄像机函数

函数 createCapture

原型 VC6.0: createCapture (long deviceID,VARIANT_BOOL bFlipImageHorizontally,
VARIANT_BOOL bSetCameraParam long width,long height)

C#: void createCapture (System.Int32 deviceID ,
System.Boolean bFlipImageHorizontally ,
System.Boolean bSetCameraParam ,
System.Int32 width ,
System.Int32 height)

Dephi: function createCapture(deviceID: Integer;
bFlipImageHorizontally: WordBool;
bSetCameraParam: WordBool;
width: Integer;
height: Integer): HRESULT;

说明 连接摄像机,使摄像机的数据链路通畅

输入参数 deviceID 表示需要连通的摄像机的台数,一般默认为一台.

bFlipImageHorizontally:摄像机图像是否需要翻转,false为不需要,默认为false

bSetCameraParam:是否需要设置摄像机图像的尺寸大小,true为需要,在后面参数的
width,height中设置参数的大小值,false为不需要,在
后面参数的
width,height中设置为0.函数将弹出DX本身的对话框,供用户设置,
默认为false.

width:摄像机图像的宽度.设置同上

height:摄像机图像的高度.设置同上.

范例 HRESULT hr = m_pCISVideoProc->createCapture(1,false,true,720,288);

2 开启摄像机函数

函数 startCapture

原型 VC6.0: HRESULT startCapture ();

C#: void startCapture ();

Dephi: function startCapture: HRESULT;

说明 将已经连接摄像机开启

范例 hr = m_pCISVideoProc->startCapture();

3 捕获一帧函数

函数 Grab

原型 VC6.0: HRESULT Grab (struct CISImageStruct * imageStruct);

C#:void startCapture ();

Dephi: function startCapture: HResult;

说明 在视频流中捕获一帧图像数据

输入参数 无

输出参数 在视频中捕获到视频数据存储在 CISImageStruct 结构中

范例 HRESULT hr = m_pCISVideoProc->Grab(m_imageStruct);

4 销毁视频流函数

函数 destroyCapture

原型 VC6.0: HRESULT destroyCapture();

C#: void destroyCapture()

Dephi: function destroyCapture: HResult;

说明 将建立的视频流销毁

范例 hr = m_pCISVideoProc->destroyCapture();

上海名图软件有限公司

六 CISA.VI.ocx组件接口说明

注:本开发包采集需要安装微软的 DirectX9.0C

1 连接摄像机函数

函数 createCapture

原型 VC6.0: createCapture (_bstr_t aviFileString,VARIANT_BOOL
bFlipImageHorizontally, VARIANT_BOOL bSetCameraParam)

C#: createCapture (System.String aviFileString ,
System.Boolean bFlipImageHorizontally ,
System.Boolean bCaptureCreated)

Dephi: function createCapture(const aviFileString: WideString;
bFlipImageHorizontally: WordBool;
out bCaptureCreated: WordBool): HRESULT;

说明 连接视频文件,使视频文件的数据链路通畅.

输入参数 aviFileString 表示视频文件的路径.

bFlipImageHorizontally: 视频文件图像是否需要翻转,false 为不需要,默认为 false

bCaptureCreated: 1.true 表示数据连接成功, 可以进行视频捕获

2. False 表示数据连接失败, 请检查视频文件格式是否有正确的
的视频解码器

输出参数 无

范例 HRESULT hr = m_pCISAVIProc->createCapture(bstrText,false,&bCreate);

2 开启视频函数:

函数 startCapture

原型 VC6.0: HRESULT startCapture ();

C#: void startCapture ();

Dephi: function startCapture: HRESULT;

说明 将已经连接的视频文件开启

范例 hr = m_pCISAVIProc->startCapture();

3 捕获一帧函数

函数 Grab

原型 VC6.0: HRESULT Grab (struct CISImageStruct * imageStruct);

C#:void startCapture ();

Dephi: function startCapture: HRESULT;

说明 在视频流中捕获一帧图像数据

输入参数 无

输出参数 在视频中捕获到视频数据存储在 CISImageStruct 结构中

范例 `HRESULT hr = m_pCISAVIProc->Grab(m_imageStruct);`

4 销毁视频流函数

函数 `destroyCapture`

原型 VC6.0: `HRESULT destroyCapture();`

C#: `void destroyCapture()`

Dephi: `function destroyCapture: HRESULT;`

说明 将建立的视频流销毁

范例 `hr = m_pCISAVIProc->destroyCapture();`

上海名图软件有限公司

七. CubicCoil.Dll 动态链接库接口说明

7.1. 用户需要了解的结构体定义

视频检测区域结构

/* 四边形结构：其四个顶点以任意一个为起点按顺序排列（逆时针、顺时针均可） */

```
typedef struct CISQuadrangle
```

```
{
```

```
    int x1;
```

```
    int y1;
```

```
    int x2;
```

```
    int y2;
```

```
    int x3;
```

```
    int y3;
```

```
    int x4;
```

```
    int y4;
```

```
}CISQuadrangle;
```

7.2. DLL函数

1)函数 CISCoilInit

原型 `BOOL __stdcall CISCoilInit()`

说明 初始化Dll，视频检测动态链接库CubicRing.dll使用前需要初始化，该初始化应在其被集成的应用程序初始化部分完成,比如VC中的OnInitDialog(),C#

InitializeComponent(),dephi的FormCreate中。

范例 `BOOL bres = CISCoilInit();`

2) 函数 CISCoilRelease

原型 `void __stdcall CISCoilRelease()`

说明 释放所有占用的内存资源，该函数应在在其被集成的应用程序结束时完成。比如VC的OnCancel,dephi的FormDestroy中

范例 `CISCoilRelease();`

3) 函数 CISSetSurvAreas

原型 `void __stdcall CISSetSurvAreas(int nAreaNum, CISQuadrangle *pQuadrangle
float* pDisturbThresh);`

说明 设置视频检测区域个数,区域范围,以及区域的可信度

输入参数 nAreaNum 视频检测区域个数

pSurvAreas 区域范围

pDisturbThresh 区域的可信度

输出参数 无

范例 int nAreaNum = 1;

Float pDisturbThresh = 0.3;

CISQuadrangle SourceRoi;

SourceRoi.x1 = 266;

SourceRoi.y1 = 123;

SourceRoi.x2 = 299;

SourceRoi.y2 = 126;

SourceRoi.x3 = 287;

SourceRoi.y3 = 1273;

SourceRoi.x4 = 237;

SourceRoi.y4 = 182;

CISSetSurvAreas(nAreaNum,&SourceRoi,&pDisturbThresh);

4) 函数 CISGetSurvResults

原型 void __stdcall CISGetSurvResults(char* pImageData, int nImgWidth, int
nImgHeight, int nImgDepth, int nImgChannel,
bool* bDisturbArray)

说明 获取设置的视频区域的检测结果

输入参数 pImageData 图像的原始 RGB 数据头指针

nImgWidth 图像的宽度

nImgHeight 图像的高度

nImgDepth 图像的深度, 一般为 8

nImgChannel 图像的通道数, 彩色为 3, 黑白为 1

输出参数 bDisturbArray 视频区域的检测结果, 与设定区域的 pSurvAreas 相对应, true 为有扰动 false 为无扰动。

范例 bool bDis=false;

CISGetSurvResults(pImage->data,pImage->width,pImage->height,8,3,&bDis);

八. 应用程序测试识别效果

- 1) 安装加密狗驱动, 加密狗驱动\eliteiv_drivers\eliteiv_drivers\EliteIV_Drivers
InstWiz3.exe
- 2) 打开应用程序, 开发资料--名图\车牌识别演示 demo\演示 demo\ AutoPlate.exe
- 3) 演示如下



识别演示程序示意图

- 4) 若无识别结果请检查:

- 1 加密狗是否已经插好
- 2.加密狗驱动是否被正确安装
- 3.PlateConfigure.ini 是否配置正确
- 4.车牌是否太脏,或者污损严重.

- 5) 若有识别结果不准确情况出现, 可通过设置光照条件参数调节, 光照条件参数越小, 识别算法越精密, 识别速度降低, 识别效果增加。

- 6) 若有把栏杆, 添加字符误认为车牌情况出现, 可通过设置识别区域设置过滤伪车牌

注: 该区域为识别函数搜索定位车牌区域, 故只需把车牌所在区域位置包含在该范围内, 把栏杆, 其他字符放在该范围之外, 便可达到过滤伪车牌的效果

九 技术支持

有什么疑问请联系我们：上海名图软件有限公司技术部

QQ: 82526516 742736415

MSN:cubicimage@hotmail.com

电话:021-61259082-602

地址：上海市闵行区东川路紫竹科学园区 2 号楼 2 层

上海名图软件有限公司