基于OpenCV的车流量检测方法及实现作品说明书

1 系统概述

本设计作品为开发和实现了一个基于OpenCV的车流量检测方法。该算法可以嵌入交通摄像头作为视频交通事件检测的基础功能之一。程序采用C++编译完成，使用者需要提前配置OpenCV开源库的开发环境，并对其有基础的了解与掌握。

算法实现了车流量统计，主要涉及技术包括前景提取算法、车流量统计算法以及车辆粘连解决方案等。车流量统计的实现思路，首先通过EM算法改进的高斯混合模型提取前景，再对前景进行阈值化、二值化、开运算等图像处理，接着通过设置虚拟检测线圈得到ROI区域内的运动车辆所占像素百分比，最后通过设置经验阈值进行比较来判断是否存在车辆通过。算法的车流量统计准确率达到94%，处理1分钟视频数据平均只需要21秒，可以满足交通流量统计的实时性要求。

2 OpenCV的配置

本文以WIN7-Visual Studio 2010-OpenCV2.4.4为例，展示如何配置OpenCV的开发环境。步骤具体如下：

（1）安装visual studio 2010；

（2）解压OpenCV2.4.4；

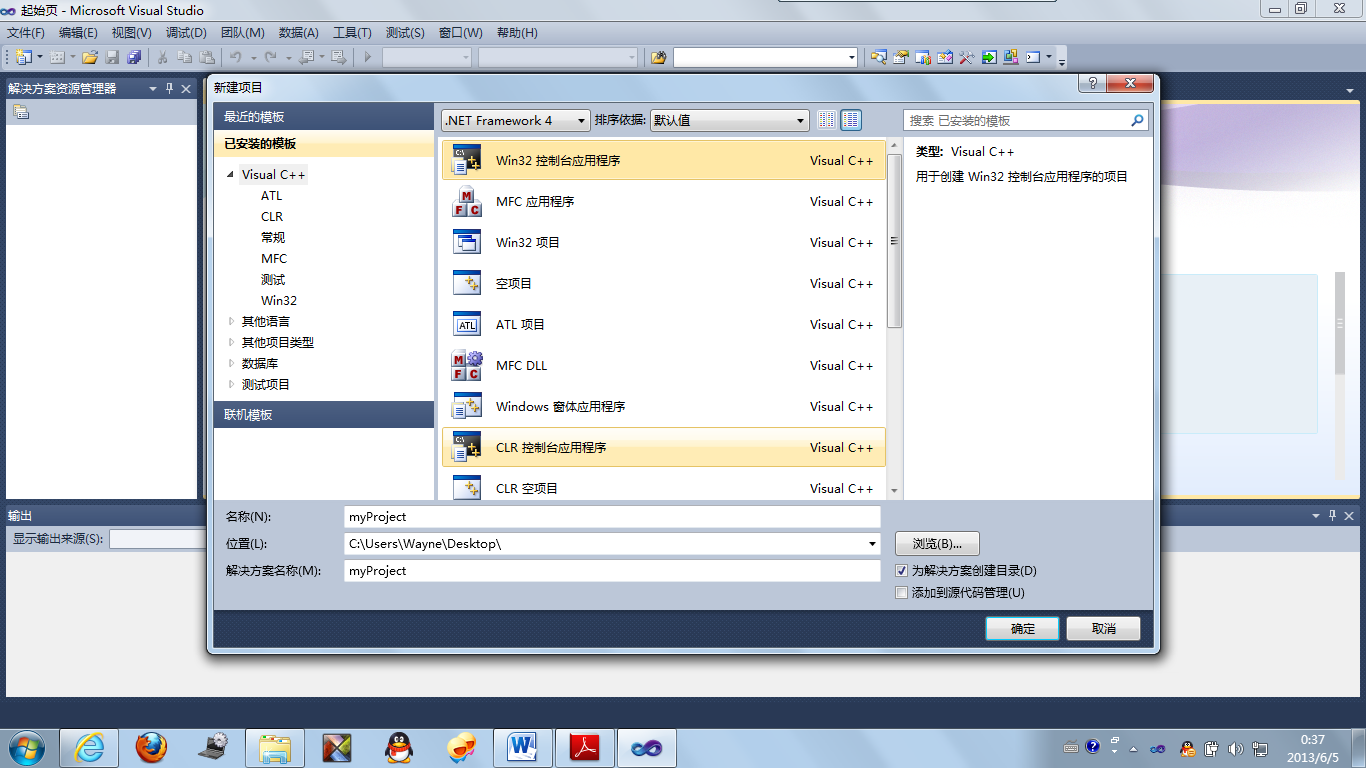
（3）设置环境变量

E:\OpenCV\opencv\build\x86\vc10\bin

E:\OpenCV\opencv\build

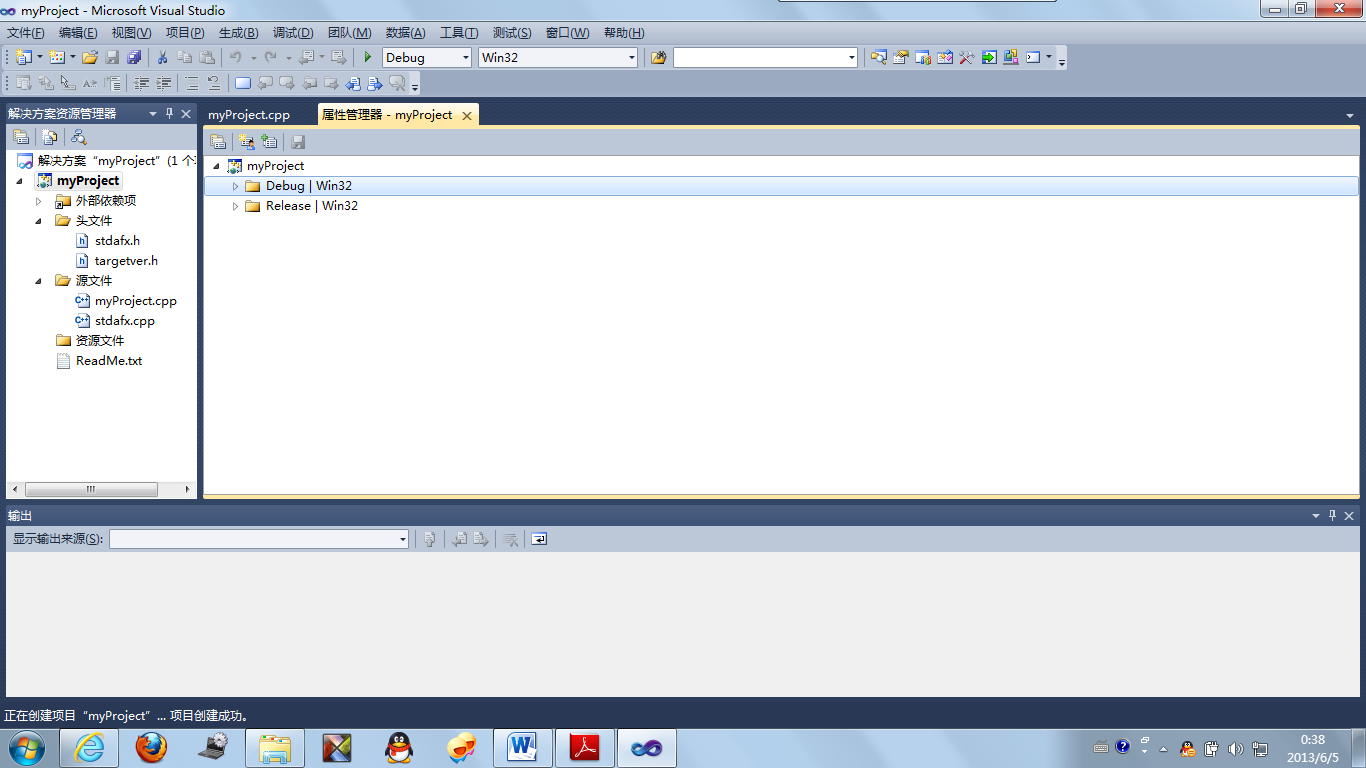
（4）配置include路径以及lib路径

打开visual studio 2010，新建一个项目，



**图2-1 新建项目页面**

点“视图-属性管理器”，



**图2-2 视频-属性管理器页面**

双击Debug|Win32，选择vc++目录，我们要设置这里的“包含目录”和“库目录”，点开包含目录的“编辑”，输入这三条路径：

E:\OpenCV\opencv\build\include\opencv2

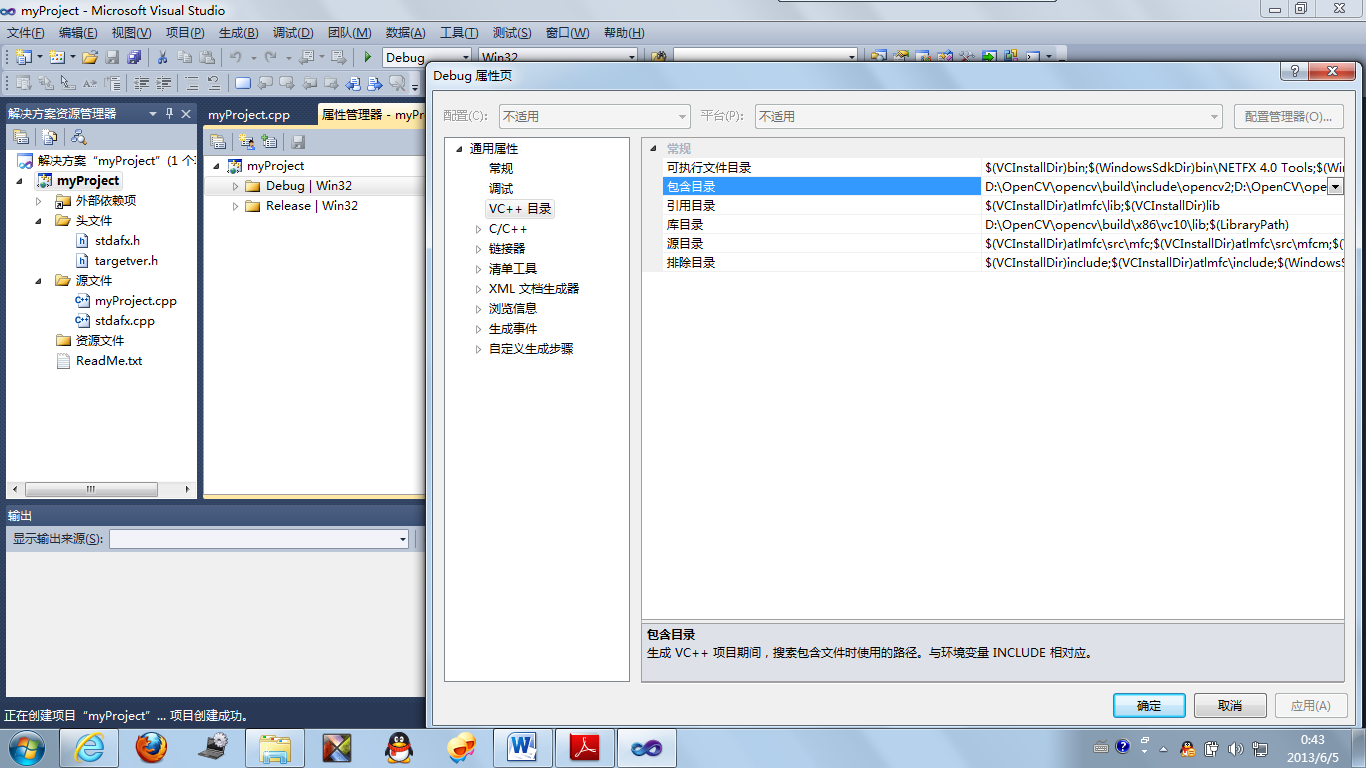
E:\OpenCV\opencv\build\include\opencv

E:\OpenCV\opencv\build\include

点开库目录的“编辑”，输入：

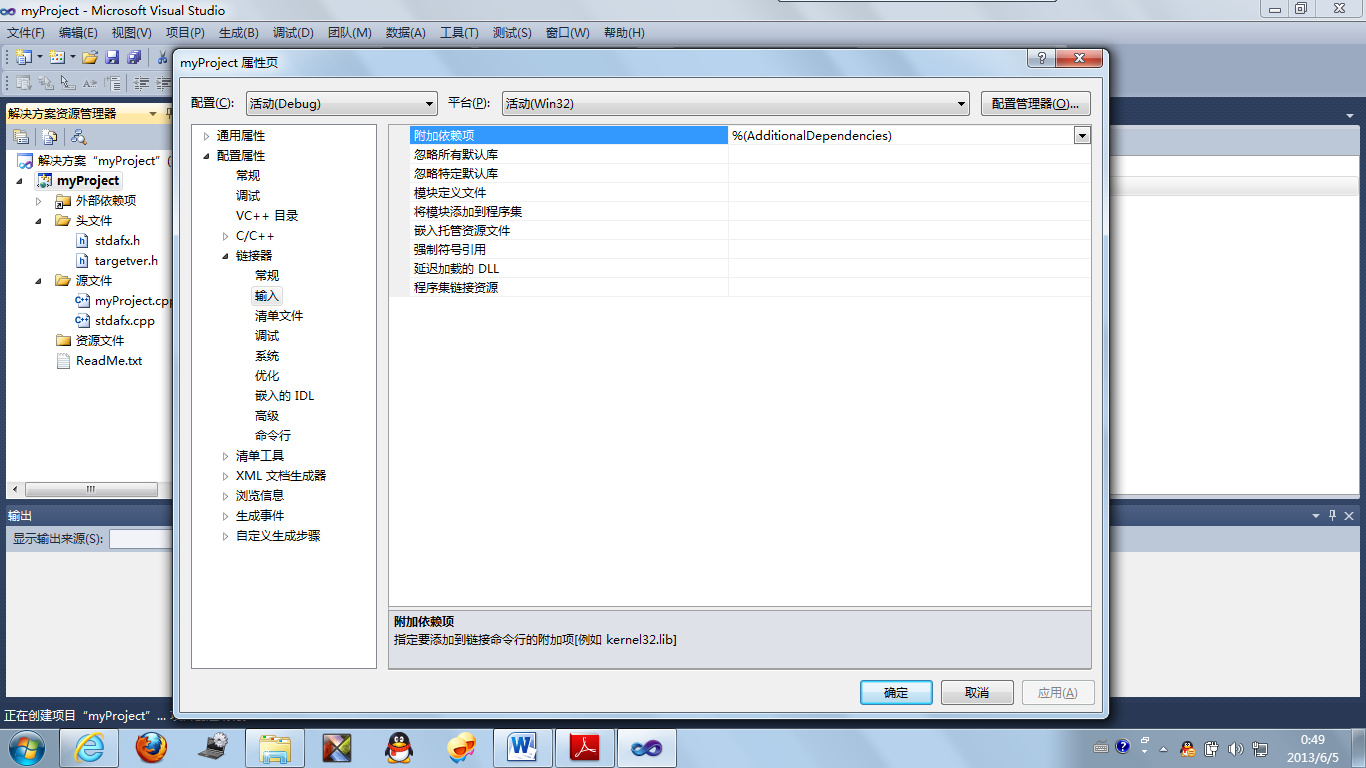
E:\OpenCV\opencv\build\x86\vc10\lib

输入完成后，结果为：



**图2-3 编辑路径页面**

打开刚才新建的工程myProject，右击项目，选择“属性”，



**图2-4 属性页面**

（1）在“活动debug”状态下，链接器->输入，点击“附加依赖项”的“编辑”，输入：

opencv\_calib3d244d.lib

opencv\_contrib244d.lib

opencv\_core244d.lib

opencv\_features2d244d.lib

opencv\_flann244d.lib

opencv\_gpu244d.lib

opencv\_highgui244d.lib

opencv\_imgproc244d.lib

opencv\_legacy244d.lib

opencv\_ml244d.lib

opencv\_objdetect244d.lib

opencv\_ts244d.lib

opencv\_video244d.lib

（2）在“Release”状态下，点击“附加依赖项”的“编辑”，输入：

opencv\_calib3d244.lib

opencv\_contrib244.lib

opencv\_core244.lib

opencv\_features2d244.lib

opencv\_flann244.lib

opencv\_gpu244.lib

opencv\_highgui244.lib

opencv\_imgproc244.lib

opencv\_legacy244.lib

opencv\_ml244.lib

opencv\_objdetect244.lib

opencv\_ts244.lib

opencv\_video244.lib

3 使用说明

3．1 参数设置

算法的参数主要是根据HK\_D\_C.avi进行调节设置的，可在PARAMETER.txt中进行修改设置，其中各字段含义如表3-1所示。

**表3-1 算法参数一览表**

|  |  |
| --- | --- |
| 字段名 | 含义 |
| ROAD1\_COIL\_P1 | 各个车道虚拟检测线圈的坐标位置，其中P1为线圈左上角点，P2为线圈右下角点 |
| ROAD1\_COIL\_P2 |
| ROAD2\_COIL\_P1 |
| ROAD2\_COIL\_P2 |
| IMPURITY\_SIZE | 噪音的尺寸阈值 |
| TIMESTAMP\_SIZE | 时间戳尺寸阈值 |
| APPROXPOLY\_PARAMETER | 多边形逼近精度 |
| BGUPDATE\_FREQUENCY | 背景模型更新频率 |
| BGUPDATE\_INTERVAL | 背景模型更新间隔 |
| COIL\_THRESHOLD | 虚拟检测线圈检测阈值 |

本程序仿真实验共测试了HK\_D\_C.avi、TEST.avi以及TEST2.mp4三个交通视频（carCount/carCount/video），可以在main.cpp进行相应设置（第4行）。当进行不同视频测试的时候，要导入不同的已经训练好的EM聚类结果（carCount/carCount/txt），在BackGroundModel.CPP进行相应设置（第193行）；同时要导入不同的参数（carCount/carCount/txt），参数主要包括虚拟线圈位置、最大噪声尺寸、形态学处理次数、检测线圈判定阈值等，在Parameter.cpp进行设置（第13行）。

以下是视频与参数、EM聚类结果的对应：

1）HK\_D\_C.avi - EM\_RESULT.txt - PARAMETER.txt

2）TEST.avi - EM\_RESULT2.txt - PARAMETER2.txt

3）TEST.mp4 - EM\_RESULT3.txt - PARAMETER3.txt

3．2 类说明

程序主要包括VideoInfo、Parameter、CarCount、BackGroundModel、ForeGroundModel、CarCount等6个类，其中各个类的功能以及可调用的方法在其头文件里都要相应描述。简单描述程序的逻辑：

（1）Main函数完成参数的传入和各个类的创建，之后每读取视频一帧，就传入CarCount类；

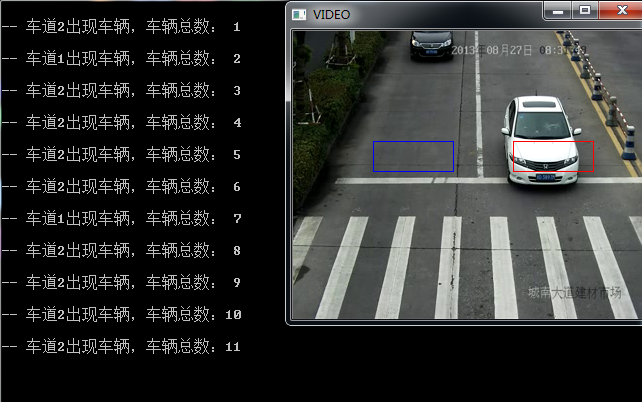
（2）CarCount类会调用BackGroundModel类进行背景更新，更新方法在main函数中设置（我设置了利用高斯混合模型进行更新），其实CarCount类调用的是ForeGroundModel，再由其调用BackGroundModel类；

（3）ForeGroundModel在获取高斯前景后，进行一些列的处理，先后包括：闭处理、利用轮廓特征去除噪声点、闭处理（自定义核，竖型）、利用轮廓特征去除时间戳、多边形逼近、填充“镂空”（这个不是很重要），之后将前景图片返回给CarCount类；

（4）CarCount首先获取前景图片的各个目标检测区域，计算其白色像素点百分比，并将其输入判断模型（模型保存了连续几帧的白色像素点百分比），由模型判断车道是否出现车辆，并进行相应的绘制；

（5）其中在模型判断过程中，还添加了粘连检测，主要解决了HK\_D\_C.avi中两辆车粘连的问题，方法比较难说，可以参考PPT中的描述。

4 程序界面



**图 4-1 程序界面**

5 OpenCV简单使用入门

OpenCV配置完成后，你可以在你的编译器中使用OpenCV的相应类以及函数。下面简单介绍一些OpenCV简单的图像操作的例子。帮助使用者入门。在使用时，只要把头文件包括进来就可以直接使用了。例如：#include “cv.h”， #include “highgui.h”（具体每个库中的内容，可参阅openCV手册）。

（1） 打开图像

//打开图像文件，成功返回IplImage\*图像指针，否则返回NULL

IplImage\* WINAPI imgFileOpen()

{

      IplImage\* pImg = 0; //声明IplImage指针

      CString fileName = "";

      //窗口对话窗体的文件属性

    static char szFilter[] = "图像文件格式(\*.bmp) |\*.bmp|JPG file format (\*.jpg)|\*.jpg|TIF file format(\*.tif) |\*.tif|";

      //打开文件对话窗口

      CFileDialog OpenDlg( TRUE, NULL, NULL/\*LastFilePath\*/, OFN\_HIDEREADONLY | OFN\_EXPLORER, szFilter, NULL );

      OpenDlg.m\_ofn.lpstrTitle = "Open BMP,JPG and TIF Files";

      //从文件对话窗口中打开图像

      if(OpenDlg.DoModal()==IDOK)

            fileName = OpenDlg.GetPathName(); //获得文件名

      else

            return NULL;

      //图像文件成功打开

      pImg = cvLoadImage(fileName, 0);

      //返回

      return pImg;

}

（2） 如果把上面打开的图像显示出来，则可以这样调用：

    IplImage\* pOpenImg = imgFileOpen(); //打开图像

    Cstring srcWinLabel = “source image”;//窗体title

      if(pOpenImg)//如果成功打开则显示

      {

         cvNamedWindow(srcWinLabel, 1); //创建窗体

         cvShowImage(srcWinLabel, pOpenImg); //显示图像

         cvWaitKey(0);      //等待按键

         cvReleaseImage(&pOpenImg);

         cvDestroyWindow( srcWinLabel );//销毁窗口

      }

（3） 一些常用操作如角点检测：

BOOL WINAPI FindCorner(IplImage\* srcImg1, double qualityLevel, double minDistance)

{

         //定义循环变得与检测角点数量

         int  i, cornerCount  = max\_corners;

         //定义IplImage指针图像

         IplImage\* eigImage1  = 0;

         IplImage\* tempImage1 = 0;

         IplImage\* smoothImg1 = 0;

         IplImage\* grayImg1   = 0;

         //初始化，分配空间

         //浮点格式

         eigImage1  = cvCreateImage(cvGetSize(srcImg1), IPL\_DEPTH\_32F, 1);

         tempImage1 = cvCreateImage(cvGetSize(srcImg1), IPL\_DEPTH\_32F, 1);

         //灰度单通道格式

         smoothImg1 = cvCreateImage(cvGetSize(srcImg1), IPL\_DEPTH\_8U, 1);

         grayImg1   = cvCreateImage(cvGetSize(srcImg1), IPL\_DEPTH\_8U, 1);

         //格式转换：转换成灰度图像

         if(srcImg1->nChannels !=1)

         {

                  cvCvtColor(srcImg1, grayImg1, CV\_BGR2GRAY);

         }

         else

         {

                  cvCopy(srcImg1, grayImg1);

         }

        //启用高斯滤波，平滑图像

 //   cvSmooth(grayImg1,smoothImg1,CV\_GAUSSIAN, 9, 9);

    cvCopy(grayImg1, smoothImg1); //不平滑时把图像COPY到smoothImg1，以便后面处理

         //角点

         cvGoodFeaturesToTrack(smoothImg1, eigImage1, tempImage1, corners,

                                    &cornerCount, qualityLevel, minDistance, 0);

    //子像素极角点

         cvFindCornerSubPix(smoothImg1, corners, cornerCount, cvSize(5,5), cvSize(-1,-1),

                                 cvTermCriteria(CV\_TERMCRIT\_ITER|CV\_TERMCRIT\_EPS, 100, 0.1) );

    // draw circles at each corner location in the gray image and

    // print out a list the corners  画出所有角点

    if(cornerCount > 0)

    {

        for (i=0; i<cornerCount; i++)

        {

            cvCircle(srcImg1, cvPoint((int)(corners[i].x), (int)(corners[i].y)), 6,

                CV\_RGB(255,0,0), 2, CV\_AA, 0);

        }

    }

         //创建窗体，以便显示标记出角点的图像

         cvNamedWindow( "corners of image", 1);

         //显示标记图像

    cvShowImage( "corners of image", srcImg1 );

         //等待按键

         cvWaitKey(0);

         //销毁窗体

         cvDestroyWindow("corners of image");

         //释放空间

         cvReleaseImage(&eigImage1);

         cvReleaseImage(&tempImage1);

         cvReleaseImage(&smoothImg1);

         cvReleaseImage(&grayImg1);//showImg1

         //返回

         return TRUE;

}

 BOOL WINAPI FindCorner(IplImage\* srcImg1, double qualityLevel, double minDistance)

{

         //定义循环变得与检测角点数量

         int  i, cornerCount  = max\_corners;

         //定义IplImage指针图像

         IplImage\* eigImage1  = 0;

         IplImage\* tempImage1 = 0;

         IplImage\* smoothImg1 = 0;

         IplImage\* grayImg1   = 0;

         //初始化，分配空间

         //浮点格式

         eigImage1  = cvCreateImage(cvGetSize(srcImg1), IPL\_DEPTH\_32F, 1);

         tempImage1 = cvCreateImage(cvGetSize(srcImg1), IPL\_DEPTH\_32F, 1);

         //灰度单通道格式

         smoothImg1 = cvCreateImage(cvGetSize(srcImg1), IPL\_DEPTH\_8U, 1);

         grayImg1   = cvCreateImage(cvGetSize(srcImg1), IPL\_DEPTH\_8U, 1);

         //格式转换：转换成灰度图像

         if(srcImg1->nChannels !=1)

         {

                  cvCvtColor(srcImg1, grayImg1, CV\_BGR2GRAY);

         }

         else

         {

                  cvCopy(srcImg1, grayImg1);

         }

        //启用高斯滤波，平滑图像

 //   cvSmooth(grayImg1,smoothImg1,CV\_GAUSSIAN, 9, 9);

    cvCopy(grayImg1, smoothImg1); //不平滑时把图像COPY到smoothImg1，以便后面处理

         //角点

         cvGoodFeaturesToTrack(smoothImg1, eigImage1, tempImage1, corners,

                                    &cornerCount, qualityLevel, minDistance, 0);

    //子像素极角点

         cvFindCornerSubPix(smoothImg1, corners, cornerCount, cvSize(5,5), cvSize(-1,-1),

                                 cvTermCriteria(CV\_TERMCRIT\_ITER|CV\_TERMCRIT\_EPS, 100, 0.1) );

    // draw circles at each corner location in the gray image and

    // print out a list the corners  画出所有角点

    if(cornerCount > 0)

    {

        for (i=0; i<cornerCount; i++)

        {

            cvCircle(srcImg1, cvPoint((int)(corners[i].x), (int)(corners[i].y)), 6,

                CV\_RGB(255,0,0), 2, CV\_AA, 0);

        }

    }

         //创建窗体，以便显示标记出角点的图像

         cvNamedWindow( "corners of image", 1);

         //显示标记图像

    cvShowImage( "corners of image", srcImg1 );

         //等待按键

         cvWaitKey(0);

         //销毁窗体

         cvDestroyWindow("corners of image");

         //释放空间

         cvReleaseImage(&eigImage1);

         cvReleaseImage(&tempImage1);

         cvReleaseImage(&smoothImg1);

         cvReleaseImage(&grayImg1);//showImg1

         //返回

         return TRUE;

}

（4）Hough 变换检测圆

BOOL WINAPI HoughCircle(IplImage\* img)

{

         if(img == NULL)

                  return FALSE;

    IplImage\* gray = cvCreateImage( cvGetSize(img), 8, 1 );

    CvMemStorage\* storage = cvCreateMemStorage(0);

//转换成灰度图像

if(img->nChannels != 1)

         cvCvtColor( img, gray, CV\_BGR2GRAY );

         else

         cvCopy(img, gray);

    //平滑化

    cvSmooth( gray, gray, CV\_GAUSSIAN, 9, 9 );

    // Hough变换求圆

   // CvSeq\* circles = cvHoughCircles( gray, storage, CV\_HOUGH\_GRADIENT, 2, gray->height/4, 200, 100 );

         CvSeq\* circles = cvHoughCircles( gray, storage, CV\_HOUGH\_GRADIENT, 2, gray->height/4, 250, 55 );

    // 画出识别出的圆

    int i;

    for( i = 0; i < circles->total; i++ ){

                  float\* p = (float\*)cvGetSeqElem( circles, i );

                  cvCircle( img, cvPoint(cvRound(p[0]),cvRound(p[1])), cvRound(p[2]), CV\_RGB(255,0,0), 3, 8, 0 );

         //       cvCircle( img, cvPoint(cvRound(p[0]),cvRound(p[1])), 100, CV\_RGB(255,0,0), 3, 8, 0 );

    }

    // 窗体显示

    cvNamedWindow("Win", 1);

    // 画像表示

    cvShowImage("Win", img);

    // 等待按键

    cvWaitKey(0);

    // 释放

    cvReleaseImage(&img);

    cvReleaseImage(&gray);

    cvDestroyWindow("Win");

    return TRUE;

}

BOOL WINAPI HoughCircle(IplImage\* img)

{

         if(img == NULL)

                  return FALSE;

    IplImage\* gray = cvCreateImage( cvGetSize(img), 8, 1 );

    CvMemStorage\* storage = cvCreateMemStorage(0);

//转换成灰度图像

if(img->nChannels != 1)

         cvCvtColor( img, gray, CV\_BGR2GRAY );

         else

         cvCopy(img, gray);

    //平滑化

    cvSmooth( gray, gray, CV\_GAUSSIAN, 9, 9 );

    // Hough变换求圆

   // CvSeq\* circles = cvHoughCircles( gray, storage, CV\_HOUGH\_GRADIENT, 2, gray->height/4, 200, 100 );

         CvSeq\* circles = cvHoughCircles( gray, storage, CV\_HOUGH\_GRADIENT, 2, gray->height/4, 250, 55 );

    // 画出识别出的圆

    int i;

    for( i = 0; i < circles->total; i++ ){

                  float\* p = (float\*)cvGetSeqElem( circles, i );

                  cvCircle( img, cvPoint(cvRound(p[0]),cvRound(p[1])), cvRound(p[2]), CV\_RGB(255,0,0), 3, 8, 0 );

         //       cvCircle( img, cvPoint(cvRound(p[0]),cvRound(p[1])), 100, CV\_RGB(255,0,0), 3, 8, 0 );

    }

    // 窗体显示

    cvNamedWindow("Win", 1);

    // 画像表示

    cvShowImage("Win", img);

    // 等待按键

    cvWaitKey(0);

    // 释放

    cvReleaseImage(&img);

    cvReleaseImage(&gray);

    cvDestroyWindow("Win");

    return TRUE;

}

（5）再比如轮廓检测

//求轮廓/////////////////////

int levels = 3;           ///

CvSeq\* contours = 0;      ///

IplImage\* imgContour;     ///

/////////////////////////////

int upper  = 80;

int lower  = 10;

CvMemStorage \* storageContours;

void on\_trackbarContour(int pos)

{

         //定义变量

         IplImage\* cnt\_img;

    CvSeq\* \_contours;

    int \_levels;

         //分配空间、初始化

    cnt\_img = cvCreateImage( cvGetSize(imgContour), 8, 3 );

    \_contours = contours;

    \_levels = levels - 3;

    cvZero( cnt\_img );

         //画轮廓

    cvDrawContours( cnt\_img, \_contours, CV\_RGB(255,0,0), CV\_RGB(0,255,0),

                              \_levels, 1, CV\_AA, cvPoint(0,0) );

         //显示轮廓图像

    cvShowImage( "contours", cnt\_img );

         //释放空间

    cvReleaseImage( &cnt\_img );

}

void WINAPI findContour( IplImage\* pImg )

{

         //定义指针变量

         CvMemStorage\* storage;

    IplImage\* img;

         //初始化分配空间

    storage = cvCreateMemStorage(0);

    img = cvCreateImage( cvGetSize(pImg), IPL\_DEPTH\_8U, 1 );

    imgContour = cvCreateImage(cvGetSize(pImg), IPL\_DEPTH\_8U, 3);

         //Copy图像，便于on\_trackbarContour中操作

         cvCopy(pImg, imgContour);

         //转换成灰度图像

if(pImg->nChannels != 1)

         cvCvtColor(pImg, img, CV\_BGR2GRAY);

else

    cvCopy(pImg, img);

         //边缘提取

    cvCanny(img, img, (float)33, (float)33\*3, 3);//0.0, 100, 5);

         //阈值分割

  //cvThreshold( img, img, 150, 255, CV\_THRESH\_BINARY );

         //创建窗体显示要操作的图像

    cvNamedWindow( "image", 1 );

    cvShowImage( "image", img );

         //找轮廓

         //CV\_RETR\_EXTERNAL,CV\_CHAIN\_APPROX\_SIMPLE

    cvFindContours( img, storage, &contours, sizeof(CvContour),

                    CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, cvPoint(0,0) );

         // comment this out if you do not want approximation

    contours = cvApproxPoly( contours, sizeof(CvContour), storage,

                                       CV\_POLY\_APPROX\_DP, 3, 1 );

         //显示轮廓图像

    cvNamedWindow( "contours", 1 );

    cvCreateTrackbar( "levels+3", "contours", &levels, 50, on\_trackbarContour );

    on\_trackbarContour(0);

    cvWaitKey(0);

         //释放空间销毁窗体

         cvReleaseImage( &imgContour);

         cvReleaseImage(&img);

         cvClearSeq(contours);

         cvClearMemStorage( storage );

         cvDestroyWindow("contours");

         cvDestroyWindow("image");

}