

基于 opencv 的 use 摄像头视频采集程序.....	1
基于 opencv 的两个摄像头数据采集.....	3
能激发你用代码做视频的冲动程序.....	6
图像反转（就是把黑的变白，白的变黑）.....	11
图像格式的转换.....	12
从摄像头或者 AVI 文件中得到视频流，对视频流进行边缘检测.....	14
采用 Canny 算子进行边缘检测.....	15
角点检测.....	18
图像的旋转加缩放（效果很拽，用地球做就像谷歌地球似的）.....	21
Log-Polar 极坐标变换.....	22
对图像进行形态学操作（图像的开闭，腐蚀和膨胀运算）.....	25
用不同的核进行图像的二维滤波.....	28
图像域的填充.....	31
寻找轮廓实现视频流的运动目标检测（超推荐一下）.....	35
采用金字塔方法进行图像分割.....	40
图像的亮度变换.....	43
单通道图像的直方图.....	46
计算和显示彩色图像的二维色调-饱和度图像.....	48
图像的直方图均匀化.....	51
用 Hough 变换检测线段.....	53
利用 Hough 变换检测圆（是圆不是椭圆）.....	57
距离变换.....	59
椭圆曲线拟合.....	64
由点集序列或数组创建凸外形.....	68
Delaunay 三角形和 Voronoi 划分的迭代式构造.....	71
利用背景建模检测运动物体（推荐）.....	78
运动模板检测（摄像头）.....	81
显示如何利用 Camshift 算法进行彩色目标的跟踪.....	87

基于 opencv 的 use 摄像头视频采集程序

准备工作：你得把 opencv 库装到电脑上，并把各种头文件，源文件，lib 库都连到 vc 上，然后设置一下系统环境变量，这里这方面就不说了，好像我前面的文章有说过，不懂也可百度一下。

建立一个基于 WIN32 控制台的工程 CameraUSB，在新建一个 c++ 元文件，写代码：

```
#include "cxcore.h"
#include "cvcam.h"
#include "windows.h"
#include "highgui.h"
```

```
void callback(IplImage* image);
int main()
{
    int ncams=cvcamGetCamerasCount( );//返回可以访问的摄像头数目
    HWND MyWin;
        // 设置系统属性
    cvcamSetProperty(0, CVCAM_PROP_ENABLE, CVCAMTRUE); //选择第一个摄像头
    //camera
        cvcamSetProperty(0, CVCAM_PROP_RENDER, CVCAMTRUE); //We' 11
    render stream
        // 在本例中
        // 假设创建一个窗口，并且窗口的 ID 是在变量 MyWin 中定义
        // MyWin 是窗口 HWND 的类型
    MyWin=(HWND) cvGetWindowHandle("CameraUSB window");
    cvcamSetProperty(0, CVCAM_PROP_WINDOW,&MyWin);      // Selects a
window for
        //video rendering
    //回调函数将处理每一帧
    cvcamSetProperty(0, CVCAM_PROP_CALLBACK, callback);
        cvcamInit( );
        cvcamStart( );
        // 现在程序开始工作
    cvWaitKey(0);
        cvcamStop( );
        cvcamExit( );
        return 0;
}

// 在图像中画兰色水平线
void callback(IplImage* image)
{
    IplImage* image1 = image;
    int i, j;

    assert (image);

    for(i=0; i<image1->height; i+=10)
    {
        for(j=(image1->widthStep)*i;
j<(image1->widthStep)*(i+1);
            j+=image1->nChannels)
        {
            image1->imageData[j]      = (char)255;
```

```
    image1->imageData[j+1] = 0;
    image1->imageData[j+2] = 0;
}
}

}
```

嘿嘿,就这么简单就完事了。

不懂可留言问

基于 opencv 的两个摄像头数据采集

实现功能: 同时采集两路 USB 摄像头数据, 并显示, 具有图片保存功能(点击左键保存图片, 并暂停视频; 右键继续视频)。步骤就不说了, 很简单, 直接放代码了:

```
#include <cvcam.h>
#include <cv.h>
#include <highgui.h>
#include "stdio.h"
#include <windows.h>

void StereoCallback(IplImage *frame1, IplImage *frame2);
void onMouse1(int Event, int x, int y, int flags, void *param);
void onMouse2(int Event, int x, int y, int flags, void *param);

IplImage *image1,*image2;

char *strleft[4]={"left1.bmp","left2.bmp","left3.bmp","left4.bmp"};
char
*strright[4]={"right1.bmp","right2.bmp","right3.bmp","right4.bmp"};

void main()
{
    HWND CaptureWindow1=0; //不赋值也行
    HWND CaptureWindow2=0;

//int ncams=cvcamGetCamerasCount(); //获取摄像头的个数, 在这里可有可无
//用对话框的形式来选取摄像头
    int *CameraNumber;
    int nSelected = cvcamSelectCamera(&CameraNumber);
```

```
/* //灰色图像
image1=cvCreateImage(cvSize(320, 240), IPL_DEPTH_8U, 1);
image2=cvCreateImage(cvSize(320, 240), IPL_DEPTH_8U, 1);
*/

//彩色图像
image1=cvCreateImage(cvSize(320, 240), IPL_DEPTH_8U, 3);
image2=cvCreateImage(cvSize(320, 240), IPL_DEPTH_8U, 3);

//初始化两个摄像头
cvNamedWindow("cvcam1 Window", 1);
CaptureWindow1=(HWND) cvGetWindowHandle("cvcam1
Window");
cvcamSetProperty(CameraNumber[0], CVCAM_PROP_ENABLE,
CVCAMTRUE);
cvcamSetProperty(CameraNumber[0], CVCAM_PROP_RENDER,
CVCAMTRUE);
cvcamSetProperty(CameraNumber[0], CVCAM_PROP_WINDOW,
&CaptureWindow1);
// cvSetMouseCallback("cvcam1 Window", onMouse1, 0);

cvNamedWindow("cvcam2 Window", 1);
CaptureWindow2=(HWND) cvGetWindowHandle("cvcam2
Window");
cvcamSetProperty(CameraNumber[1], CVCAM_PROP_ENABLE,
CVCAMTRUE);
cvcamSetProperty(CameraNumber[1], CVCAM_PROP_RENDER,
CVCAMTRUE);
cvcamSetProperty(CameraNumber[1], CVCAM_PROP_WINDOW,
&CaptureWindow2);
// cvSetMouseCallback("cvcam2 Window", onMouse2, 0);

//让两个摄像头同步
cvcamSetProperty(CameraNumber[0], CVCAM_STEREO_CALLBACK, (void*)&S
tereoCallback);

//启动程序
cvcamInit();
cvcamStart();
cvSetMouseCallback("cvcam1 Window", onMouse1, 0);
cvSetMouseCallback("cvcam2 Window", onMouse2, 0);
cvWaitKey(0);

cvcamStop();
free(CameraNumber);
```

```
cvcamExit();
    cvDestroyWindow("cvcam1 Window");
    cvDestroyWindow("cvcam2 Window");
}

void StereoCallback(IplImage* frame1, IplImage *frame2)
{

/*      //把图像转换成灰度图并保存到 image 中
cvCvtColor(frame1, image1, CV_RGB2GRAY);
cvCvtColor(frame2, image2, CV_RGB2GRAY);
*/
//拷贝图像到全局变量 image 中 该函数这样用存在问题
// cvCopy(frame1, image1);
// cvCopy(frame2, image2);
image1=cvCloneImage(frame1);
image2=cvCloneImage(frame2);
//对截取的图像翻转
cvFlip(image1, image1, 0);
cvFlip(image2, image2, 0);

}
void onMouse1(int Event, int x, int y, int flags, void *param)
{

static int num=0;
    if(Event==CV_EVENT_LBUTTONDOWN)
    {
        if(num==4) num=0;//只是固定定义了保存 4 张图片，为了不让程序非法而
设置的复原
        cvcamPause();
        //图像保存
        cvSaveImage(strleft[num], image1);
//        cvSaveImage(strright[num], image2);
        // cvSaveImage("left.bmp", image1);
        // cvSaveImage("right.bmp", image2);

    }
    if(Event==CV_EVENT_RBUTTONDOWN)
    {
        cvcamResume();
        num++;
    }
}
```

```
}

void onMouse2(int Event, int x, int y, int flags, void *param)
{

static int num=0;
    if(Event==CV_EVENT_LBUTTONDOWN)
    {
        if(num==4) num=0;//只是固定定义了保存 4 张图片，为了不让程序非法而
设置的复原
        cvcamPause();
        //图像保存
//        cvSaveImage(strleft[num], image1);
        cvSaveImage(strright[num], image2);
//        cvSaveImage("left.bmp", image1);
//        cvSaveImage("right.bmp", image2);

    }
    if(Event==CV_EVENT_RBUTTONDOWN)
    {
        cvcamResume();
        num++;
    }

}
```

能激发你用代码做视频的冲动程序

这个程序是基于 opencv 的，连接库就不说了，直接建立一个基于 win32 的控制台程序，写代码就 OK 了。

```
/* 程序名: drawing..c
功能: 展示 OpenCV 的图像绘制功能
*/
#include "cv.h"
#include "highgui.h"
#include <stdlib.h>
#include <stdio.h>
#define NUMBER 100
#define DELAY 5
char wndname[] = "Drawing Demo";
```

```
CvScalar random_color(CvRNG* rng) //函数 cvRNG 初始化随机数生成器并返
    回其状态，RNG 随机数生成器
{
    int icolor = cvRandInt(rng); //函数 cvRandInt 返回均匀分布的随
        机 32-bit 无符号整型值并更新 RNG 状态
    return CV_RGB(icolor&255, (icolor>>8)&255, (icolor>>16)&255); //创建一个色彩值
}

int main( int argc, char** argv )
{
    int line_type = CV_AA; // change it to 8 to see non-antialiased
graphics
    int i;
    CvPoint pt1, pt2; //基于二维整形坐标轴的点
    double angle;
    CvSize sz; //矩形框大小，以像素为精度
    CvPoint ptt[6];
    CvPoint* pt[2];
    int arr[2];
    CvFont font;
    CvRNG rng;
    int width = 1000, height = 700;
    int width3 = width*3, height3 = height*3;
    CvSize text_size;
    int ymin = 0;
    // Load the source image
    IplImage* image = cvCreateImage( cvSize(width,height), 8, 3 );
    IplImage* image2;

    // Create a window
    cvNamedWindow(wndname, 1 );
    cvZero( image ); //#define cvZero cvSetZero      void
cvSetZero( CvArr* arr ); arr 要被清空数组
    cvShowImage(wndname, image);

    rng = cvRNG((unsigned)-1);
    pt[0] = &(ptt[0]);
    pt[1] = &(ptt[3]);

    arr[0] = 3;
    arr[1] = 3;

    for (i = 0; i< NUMBER; i++)
    {
```

```
pt1.x=cvRandInt(&rng) % width3 - width;
pt1.y=cvRandInt(&rng) % height3 - height;
pt2.x=cvRandInt(&rng) % width3 - width;
pt2.y=cvRandInt(&rng) % height3 - height;

cvLine( image, pt1, pt2, random_color(&rng),
cvRandInt(&rng)%10, line_type, 0 );//绘制连接两个点的线段
cvShowImage(wndname, image);
cvWaitKey(DELAY);
}

for (i = 0; i< NUMBER; i++)
{
    pt1.x=cvRandInt(&rng) % width3 - width;
    pt1.y=cvRandInt(&rng) % height3 - height;
    pt2.x=cvRandInt(&rng) % width3 - width;
    pt2.y=cvRandInt(&rng) % height3 - height;

    cvRectangle( image, pt1, pt2, random_color(&rng),
cvRandInt(&rng)%10-1, line_type, 0 );//绘制简单、指定粗细或者带填充的
矩形
    cvShowImage(wndname, image);
    cvWaitKey(DELAY);
}

for (i = 0; i< NUMBER; i++)
{
    pt1.x=cvRandInt(&rng) % width3 - width;
    pt1.y=cvRandInt(&rng) % height3 - height;
    sz.width =cvRandInt(&rng)%200;
    sz.height=cvRandInt(&rng)%200;
    angle = (cvRandInt(&rng)%1000)*0.180;

    cvEllipse( image, pt1, sz, angle, angle - 100, angle + 200,
random_color(&rng),
cvRandInt(&rng)%10-1, line_type, 0 );//函数 cvEllipse 用来绘制或者填充
一个简单的椭圆弧或椭圆扇形
    cvShowImage(wndname, image);
    cvWaitKey(DELAY);
}

for (i = 0; i< NUMBER; i++)
{
    pt[0][0].x=cvRandInt(&rng) % width3 - width;
    pt[0][0].y=cvRandInt(&rng) % height3 - height;
```

```
pt[0][1].x=cvRandInt(&rng) % width3 - width;
pt[0][1].y=cvRandInt(&rng) % height3 - height;
pt[0][2].x=cvRandInt(&rng) % width3 - width;
pt[0][2].y=cvRandInt(&rng) % height3 - height;
pt[1][0].x=cvRandInt(&rng) % width3 - width;
pt[1][0].y=cvRandInt(&rng) % height3 - height;
pt[1][1].x=cvRandInt(&rng) % width3 - width;
pt[1][1].y=cvRandInt(&rng) % height3 - height;
pt[1][2].x=cvRandInt(&rng) % width3 - width;
pt[1][2].y=cvRandInt(&rng) % height3 - height;

cvPolyLine( image, pt, arr, 2, 1, random_color(&rng),
cvRandInt(&rng)%10, line_type, 0 );//函数 cvPolyLine 绘制一个简单的或多样的多角曲线
cvShowImage(wndname, image);
cvWaitKey(DELAY);
}

for (i = 0; i< NUMBER; i++)
{
    pt[0][0].x=cvRandInt(&rng) % width3 - width;
    pt[0][0].y=cvRandInt(&rng) % height3 - height;
    pt[0][1].x=cvRandInt(&rng) % width3 - width;
    pt[0][1].y=cvRandInt(&rng) % height3 - height;
    pt[0][2].x=cvRandInt(&rng) % width3 - width;
    pt[0][2].y=cvRandInt(&rng) % height3 - height;
    pt[1][0].x=cvRandInt(&rng) % width3 - width;
    pt[1][0].y=cvRandInt(&rng) % height3 - height;
    pt[1][1].x=cvRandInt(&rng) % width3 - width;
    pt[1][1].y=cvRandInt(&rng) % height3 - height;
    pt[1][2].x=cvRandInt(&rng) % width3 - width;
    pt[1][2].y=cvRandInt(&rng) % height3 - height;

    cvFillPoly( image, pt, arr, 2, random_color(&rng),
line_type, 0 );//函数 cvFillPoly 用于一个单独被多变形轮廓所限定的区域内进行填充
    cvShowImage(wndname, image);
    cvWaitKey(DELAY);
}

for (i = 0; i< NUMBER; i++)
{
    pt1.x=cvRandInt(&rng) % width3 - width;
    pt1.y=cvRandInt(&rng) % height3 - height;
```

```
cvCircle( image, pt1, cvRandInt(&rng)%300,
random_color(&rng),
cvRandInt(&rng)%10-1, line_type,
0 );//函数 cvCircle 绘制或填充一个给定圆心和半径的圆
cvShowImage(wndname, image);
cvWaitKey(DELAY);
}

for (i = 1; i< NUMBER; i++)
{
    pt1.x=cvRandInt(&rng) % width3 - width;
    pt1.y=cvRandInt(&rng) % height3 - height;

    cvInitFont( &font, cvRandInt(&rng) % 8,
(cvRandInt(&rng)%100)*0.05+0.1,
(cvRandInt(&rng)%100)*0.05+0.1,
(cvRandInt(&rng)%5)*0.1, cvRound(cvRandInt(&rng)%10),
line_type );//字体结构初始化。函数 cvRound, cvFloor, cvCeil 用一种舍入
方法将输入浮点数转换成整数。 cvRound 返回和参数最接近的整数值

    cvPutText( image, "Northeast Petroleum University!", pt1,
&font, random_color(&rng));//在图像中加入文本
    cvShowImage(wndname, image);
    cvWaitKey(DELAY);
}

cvInitFont( &font, CV_FONT_HERSHEY_COMPLEX, 3, 3, 0.0, 5,
line_type );

cvGetTextSize( "opencv forever!", &font, &text_size, &ymin );//设置字符串文本的宽度和高度

pt1.x = (width - text_size.width)/2;
pt1.y = (height + text_size.height)/2;
image2 = cvCloneImage(image);

for( i = 0; i < 255; i++ )
{
    cvSubS( image2, cvScalarAll(i), image, 0 );//函数 cvSubS
从原数组的每个元素中减去一个数量
    cvPutText( image, "shentuhongfeng      forever!", pt1,
&font, CV_RGB(255, i, i));
    cvShowImage(wndname, image);
    cvWaitKey(DELAY);
}
```

```
// Wait for a key stroke; the same function arranges events
processing
    cvWaitKey(0);
    cvReleaseImage(&image);
    cvReleaseImage(&image2);
    cvDestroyWindow(wndname);

return 0;
}
```

效果图：太帅了

图像反转（就是把黑的变白，白的变黑）

黑的变白了，白的变黑了

源码：

```
#include<stdio.h>
#include<math.h>
#include<cv.h>
#include<highgui.h>

int main(int argc, char* argv[])
{
    IplImage* img=0;
    int height, width, step, channels;
    UCHAR* data;
    int i, j, k;
    if(argc<2)
    {
        printf("Usage: InvImage<image-file-name>\n\7");
        exit(0);
    }
    img=cvLoadImage(argv[1]);
    if(!img)
    {
        printf("Could not load image file:%s\n", argv[1]);
        exit(0);
    }
    height=img->height;
    width=img->width;
    step=img->widthStep;
    channels=img->nChannels;
```

```
data=(UCHAR*) img->imageData;
printf("Processing a%d*d image with %d
channels\n", height, width, channels);

cvNamedWindow("mainWin", CV_WINDOW_AUTOSIZE);
cvMoveWindow("mainWin", 100, 100);

for(i=0;i<height;i++)
    for(j=0;j<width;j++)
        for(k=0;k<channels;k++)
            data[i*step+j*channels+k]=255-data[i*step+j*channels+k];

cvShowImage("mainWin", img);
cvWaitKey(0);
cvReleaseImage(&img);
return 0;
}
```

图像格式的转换

首先要准备一张图片，和几个 txt 文档，把 txt 文档的扩展名改成一个你要把图片转换成的格式

我用的原始图片是 jpg 的，txt 改成 bmp 的

使用时，运行 cmd cd 转到你的目录- Convert.exe 1.jpg 2.bmp 运行就能把图像 1.jpg 转换成 2.bmp 了

源码如下：

```
/* 程序名：convert.c
功能：图像格式的转换
*/
#include <cv.h>
#include <highgui.h>
#include <stdio.h>
int main( int argc, char** argv )
{
IplImage* src;

// -1: the loaded image will be loaded as is (with number of channels
depends on the file).
if(argc != 3)
{
```

```
printf("CONV: Image format conversion, support\n"
    "JPG, BMP, TIF, PNG, PPM\\n");
    printf("Usage: conv srcImage dstImage\\n");
    return 0;
}
if( ( strstr(argv[1], ".jpg")==NULL
&& strstr(argv[1], ".bmp")==NULL
&& strstr(argv[1], ".tif")==NULL
&& strstr(argv[1], ".png")==NULL
&& strstr(argv[1], ".ppm")==NULL )
    || ( strstr(argv[2], ".jpg")==NULL
&& strstr(argv[2], ".bmp")==NULL
&& strstr(argv[2], ".tif")==NULL
&& strstr(argv[2], ".png")==NULL
&& strstr(argv[2], ".ppm")==NULL ) ) //strstr(a, b)的用法是不是在 a 数组
内查看是否有 b 数组。。。没有则输出 NULL
{
    printf("WARNING: CONV only support JPG, BMP, TIF, PPM, TGA and
PPM\\n");
}
else {
if( (src=cvLoadImage(argv[1], -1))!= 0 ) {
    cvSaveImage( argv[2], src);
    cvReleaseImage(&src);
    printf("\\n Convert successfully. \\n");
}
else
{
    printf("\\n*** Read or write image fails *** \\n");
}
}
return 0;
}
```

发现了个小问题：

原来的 jpg 图像只有 102KB 转换成 bmp 后变成 549KB ， 在运行程序把这个 bmp 转成 jpg 又只有 81KB。这真是汗死我了

从摄像头或者 AVI 文件中得到视频流，对视频流进行边缘检测

```
/*
程序名称: laplace.c
功能: 从摄像头或者 AVI 文件中得到视频流，对视频流进行边缘检测，并输出结果。
*/
#include "cv.h"
#include "highgui.h"
#include <ctype.h>
#include <stdio.h>

int main( int argc, char** argv )
{
    IplImage* laplace = 0;
    IplImage* colorlaplace = 0;
    IplImage* planes[3] = { 0, 0, 0 } ; // 多个图像面
    CvCapture* capture = 0;

    // 下面的语句说明在命令行执行程序时，如果指定 AVI 文件，那么处理
    // 从
    // AVI 文件读取的视频流，如果不指定输入变量，那么处理从摄像头获取
    // 的视频流
    if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 && isdigit(argv[1][0])))
        capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' :
0 );
    else if( argc == 2 )
        capture = cvCaptureFromAVI( argv[1] );

    if( !capture )
    {
        fprintf(stderr, "Could not initialize capturing...\n");
        return -1;
    }

    cvNamedWindow( "Laplacian", 0 );

    // 循环捕捉，直到用户按键跳出循环体
    for(;;)
    {
        IplImage* frame = 0;
        int i;
```

```
frame = cvQueryFrame( capture );
if( !frame )
    break;

if( !laplace )
{
    for( i = 0; i < 3; i++ )
        planes[i] =
cvCreateImage( cvSize(frame->width, frame->height), 8, 1 );
laplace = cvCreateImage( cvSize(frame->width, frame->height),
IPL_DEPTH_16S, 1 );
    colorlaplace =
cvCreateImage( cvSize(frame->width, frame->height), 8, 3 );
}
cvCvtPixToPlane( frame, planes[0], planes[1], planes[2],
0 );
for( i = 0; i < 3; i++ )
{
    cvLaplace( planes[i], laplace, 3 ); // 3:
aperture_size
    cvConvertScaleAbs( laplace, planes[i], 1, 0 ); //
planes[] = ABS(laplace)
}
cvCvtPlaneToPix( planes[0], planes[1], planes[2], 0,
colorlaplace );
colorlaplace->origin = frame->origin;

cvShowImage("Laplacian", colorlaplace );

if( cvWaitKey(10) >= 0 )
    break;
}
cvReleaseCapture( &capture );
cvDestroyWindow("Laplacian");
return 0;
}
```

采用 Canny 算子进行边缘检测

```
#include "cv.h"
#include "highgui.h"
```

```
char wndname[] = "Edge";
char tbarname[] = "Threshold";
int edge_thresh = 1;

IplImage *image = 0, *cedge = 0, *gray = 0, *edge = 0;

// 定义跟踪条的 callback 函数
void on_trackbar(int h)
{
    cvSmooth( gray, edge, CV_BLUR, 3, 3, 0 );
    cvNot( gray, edge );

    // 对灰度图像进行边缘检测
    cvCanny( gray, edge, (float)edge_thresh, (float)edge_thresh*3, 3 );
    cvZero( cedge );
    // copy edge points
    cvCopy( image, cedge, edge );
    // 显示图像
    cvShowImage(wndname, cedge);
}

int main( int argc, char** argv )
{
    char* filename = argc == 2 ? argv[1] : (char*)"fruits.jpg";

    if( (image = cvLoadImage( filename, 1)) == 0 )
        return -1;

    // Create the output image
    cedge = cvCreateImage(cvSize(image->width, image->height),
IPL_DEPTH_8U, 3);

    // 将彩色图像转换为灰度图像
    gray = cvCreateImage(cvSize(image->width, image->height),
IPL_DEPTH_8U, 1);
    edge = cvCreateImage(cvSize(image->width, image->height),
IPL_DEPTH_8U, 1);
    cvCvtColor(image, gray, CV_BGR2GRAY);

    // Create a window
    cvNamedWindow(wndname, 1);

    // create a toolbar
    cvCreateTrackbar(tbarname, wndname, &edge_thresh, 100,
on_trackbar);
```

```
// Show the image
on_trackbar(1);

// Wait for a key stroke; the same function arranges events
processing
cvWaitKey(0);
cvReleaseImage(&image);
cvReleaseImage(&gray);
cvReleaseImage(&edge);
cvDestroyWindow(wndname);

return 0;
}
```

/*****代码中的函数说明

1、cvSmooth，其函数声明为：

```
cvSmooth( const void* srcarr, void* dstarr, int smoothtype, int param1,
int param2, double param3 )
```

cvSmooth 函数的作用是对图象做各种方法的图象平滑。其中，srcarr 为输入图象；dstarr 为输出图象；

param1 为平滑操作的第一个参数；param2 为平滑操作的第二个参数(如果 param2 值为 0，则表示它被设为 param1)；

param3 是对应高斯参数的标准差。

参数 smoothtype 是图象平滑的方法选择，主要的平滑方法有以下五种：

CV_BLUR_NO_SCALE：简单不带尺度变换的模糊，即对每个象素在 $\text{param1} \times \text{param2}$ 领域求和。

CV_BLUR：对每个象素在 $\text{param1} \times \text{param2}$ 领域求和并做尺度变换 $1 / (\text{param1} \times \text{param2})$ 。

CV_GAUSSIAN：对图像进行核大小为 $\text{param1} \times \text{param2}$ 的高斯卷积。

CV_MEDIAN：对图像进行核大小为 $\text{param1} \times \text{param1}$ 的中值滤波（邻域必须是方的）。

CV_BILATERAL：双向滤波，应用双向 3×3 滤波，彩色设置为 param1，空间设置为 param2。

2、void cvNot(const CvArr* src, CvArr* dst);

函数 cvNot() 会将 src 中的每一个元素的每一位取反，然后把结果赋给 dst。

因此，一个值为 0x00 的 8 位图像将被映射到 0xff，而值为 0x83 的图像将被映射到 0x7c。

3、void cvCanny(const CvArr* image, CvArr* edges, double threshold1, double threshold2, int aperture_size=3);

采用 Canny 算法做边缘检测

image

输入图像

edges

输出的边缘图像

threshold1

第一个阈值

threshold2

第二个阈值

aperture_size

Sobel 算子内核大小

4、void cvCopy(const CvArr* src, CvArr* dst, const CvArr* mask=NULL);

在使用这个函数之前，你必须用 cvCreateImage () 一类的函数先开一段内存，

然后传递给 dst。

cvCopy 会把 src 中的数据复制到 dst 的内存中。

5、cvCreateTrackbar

创建 trackbar 并将它添加到指定的窗口。

int cvCreateTrackbar(const char* trackbar_name, const char* window_name,
int* value, int count, CvTrackbarCallback on_change);

trackbar_name

被创建的 trackbar 名字。

window_name

窗口名字，这个窗口将为被创建 trackbar 的父对象。

value

整数指针，它的值将反映滑块的位置。这个变量指定创建时的滑块位置。

count

滑块位置的最大值。最小值一直是 0。

on_change

每次滑块位置被改变的时候，被调用函数的指针。这个函数应该被声明为 void
Foo(int);

如果没有回调函数，这个值可以设为 NULL。

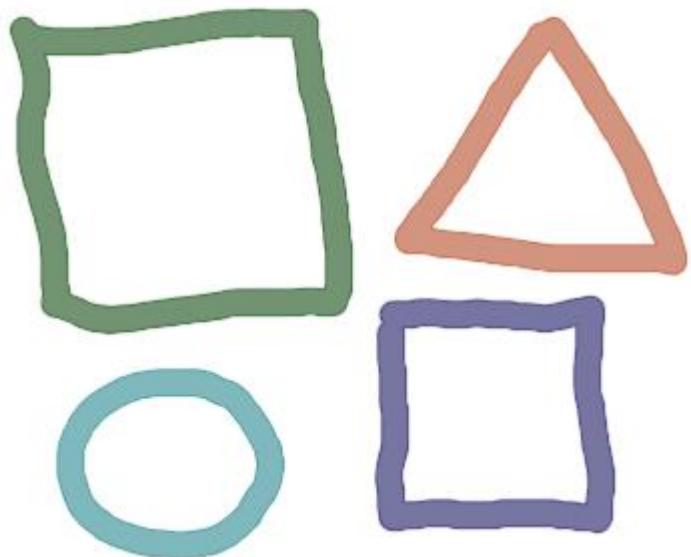
函数 cvCreateTrackbar 用指定的名字和范围来创建 trackbar（滑块或者范围控制），指定与 trackbar 位置同步的变量，

并且指定当 trackbar 位置被改变的时候调用的回调函数。被创建的 trackbar 显示在指定窗口的顶端。

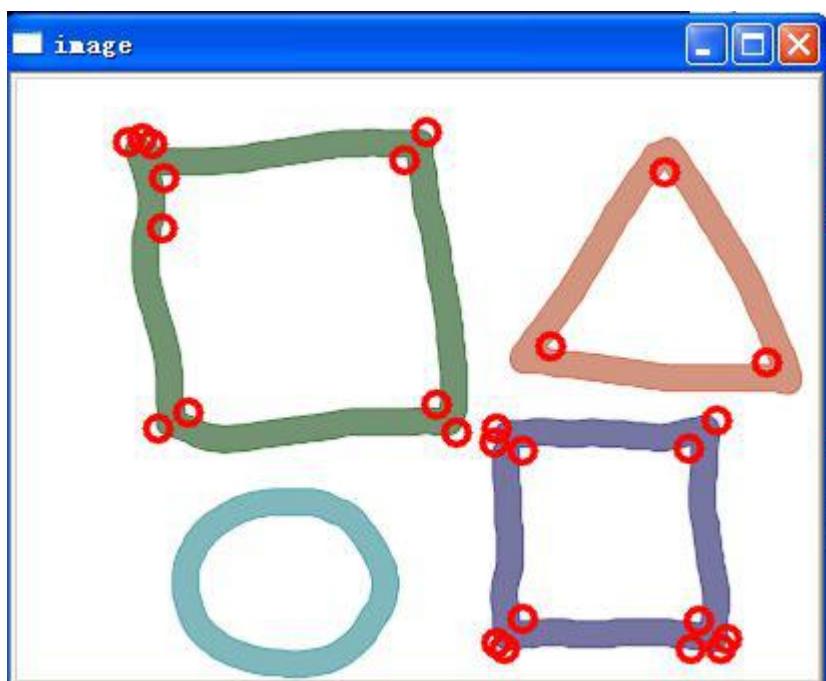
*/

角点检测

原始图：



处理后图:



源代码:

```
#include <stdio.h>
#include "cv.h"
#include "highgui.h"
#define max_corners 100

int main( int argc, char** argv )
{
```

```
int cornerCount=max_corners;
CvPoint2D32f corners[max_corners];
IplImage *srcImage = 0, *grayImage = 0, *corners1 = 0, *corners2
= 0;
int i;
CvScalar color = CV_RGB(255, 0, 0);
char* filename = argc == 2 ? argv[1] : (char*)"pic3.png"; // 注意相对路径

cvNamedWindow("image", 1); // create HighGUI window with name "image"

//Load the image to be processed
srcImage = cvLoadImage(filename, 1);

grayImage = cvCreateImage(cvGetSize(srcImage), IPL_DEPTH_8U, 1);

//copy the source image to copy image after converting the format
cvCvtColor(srcImage, grayImage, CV_BGR2GRAY);

//create empty images of same size as the copied images
corners1= cvCreateImage(cvGetSize(srcImage), IPL_DEPTH_32F, 1);
corners2= cvCreateImage(cvGetSize(srcImage), IPL_DEPTH_32F, 1);

cvGoodFeaturesToTrack (grayImage, corners1,
                      corners2, corners,
                      &cornerCount, 0.05,
                      5,
                      0,
                      3, // block size
                      0, // not use harris
                      0.4 );

printf("num corners found: %d\n", cornerCount);

// draw circles at each corner location in the gray image and
//print out a list the corners
if(cornerCount>0)
{
    for (i=0; i<cornerCount; i++)
    {
        cvCircle(srcImage, cvPoint((int)(corners[i].x),
        (int)(corners[i].y)), 6,
                 color, 2, CV_AA, 0);
    }
}
```

```
    }
```

```
    cvShowImage( "image", srcImage );
```

```
    cvReleaseImage(&srcImage);
```

```
    cvReleaseImage(&grayImage);
```

```
    cvReleaseImage(&corners1);
```

```
    cvReleaseImage(&corners2);
```

```
    cvWaitKey(0); // wait for key. The function has
```

```
    return 0;
```

```
}
```

友情链接一下，这是别人写的：

<http://hi.baidu.com/xiaoduo170/blog/item/2816460175c8330779ec2c64.htm>
1

图像的旋转加缩放(效果很拽,用地球做就像谷歌地球似的)

```
#include "cv.h"
#include "highgui.h"
#include "math.h"
int main( int argc, char** argv )
{
IplImage* src;
/* the first command line parameter must be image file name */
if( argc==2 && (src = cvLoadImage(argv[1], -1))!=0)
{
    IplImage* dst = cvCloneImage( src );
    int delta = 1;
    int angle = 0;
        int opt = 1;      // 1: 旋转加缩放
                           // 0: 仅仅旋转
    double factor;
    cvNamedWindow( "src", 1 );
    cvShowImage( "src", src );

    for(;;)
    {
        float m[6];
                           // Matrix m looks like:
                           //
```

```

    // [ m0 m1 m2 ] ==> [ A11 A12      b1 ]
    // [ m3 m4 m5 ]           [ A21 A22      b2 ]
    //
CvMat M = cvMat( 2, 3, CV_32F, m );
int w = src->width;
int h = src->height;
if(opt) // 旋转加缩放
    factor = (cos(angle*CV_PI/180.) +
1.05)*2;
else // 仅仅旋转
    factor = 1;
m[0] = (float)(factor*cos(-angle*2*CV_PI/180.));
m[1] = (float)(factor*sin(-angle*2*CV_PI/180.));
m[3] = -m[1];
m[4] = m[0];
// 将旋转中心移至图像中间
m[2] = w*0.5f;
m[5] = h*0.5f;
// dst(x, y) = A * src(x, y) + b
cvGetQuadrangleSubPix( src, dst, &M );// 提取象素四边形，使用子象
素精度
cvNamedWindow( "dst", 1 );
cvShowImage( "dst", dst );
if( cvWaitKey(5) == 27 )
    break;
angle =(int) (angle + delta) % 360;
} // for-loop
}
return 0;
}

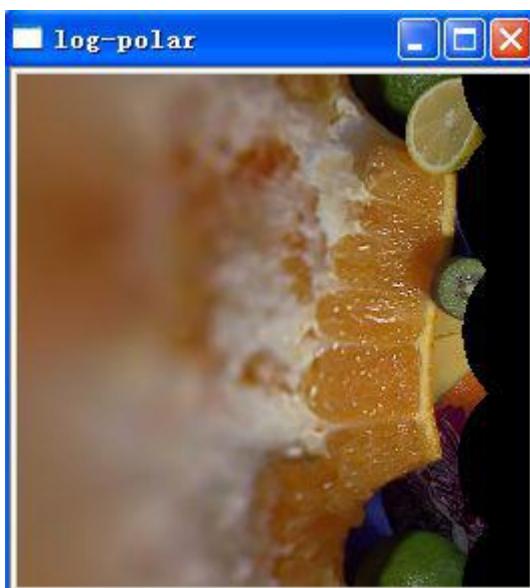
```

Log-Polar 极坐标变换

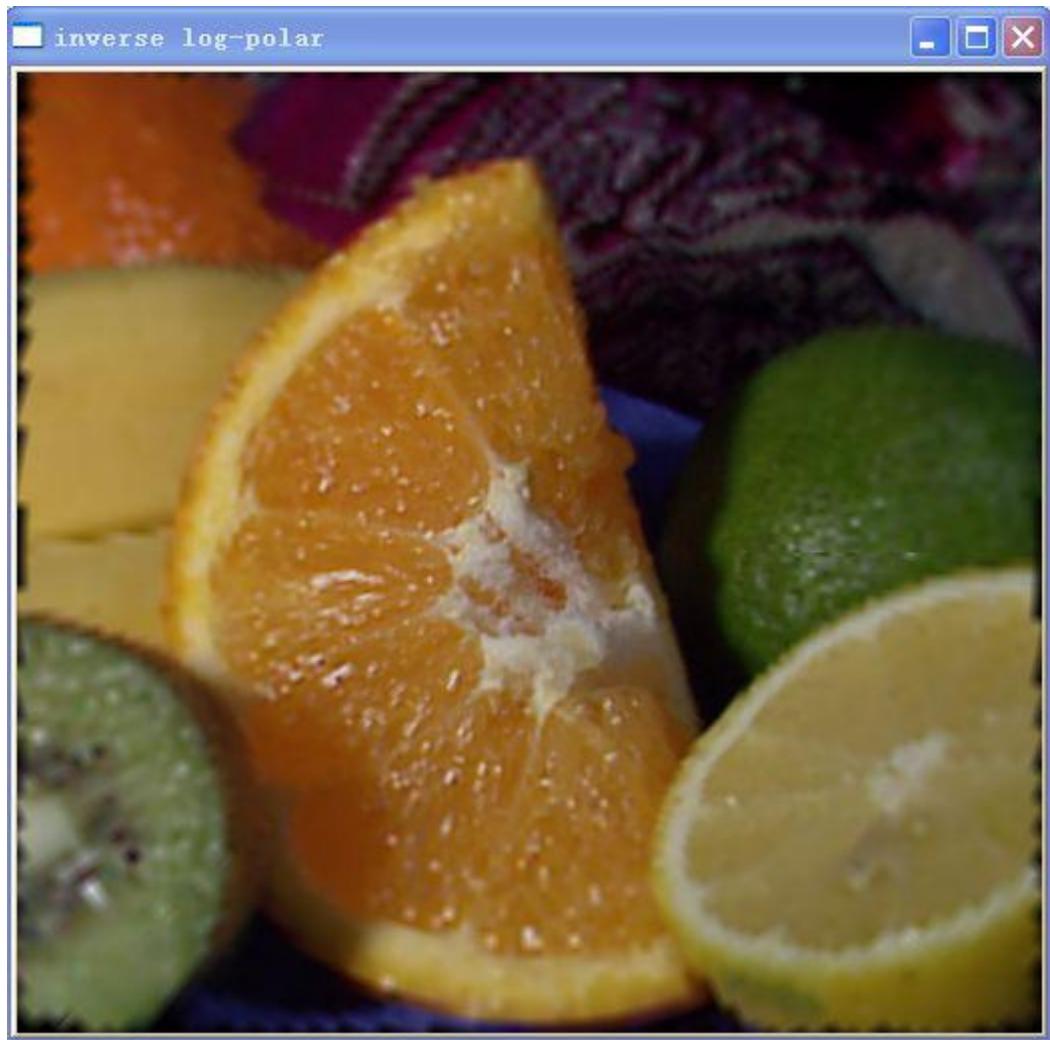
原始图：



效果图：（正变换）



反变换：



正反变换只是函数中一个参数的不同，具体看你所需要的应用。

cvLogPolar 函数可以用来模拟人类的中央视觉 (foveal vision)，并可以用于物体跟踪方面的尺度及旋转不变模板的快速匹配。

源代码：

```
#include <cv.h>
#include <highgui.h>

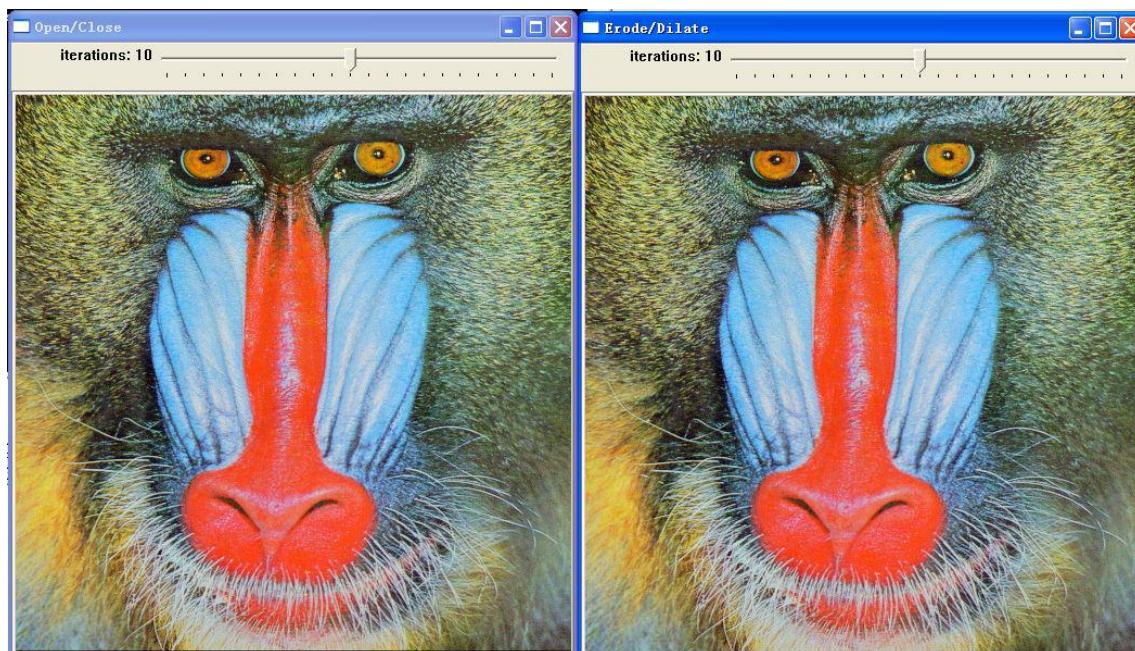
int main(int argc, char** argv)
{
    IplImage* src;

    if( argc == 2 && (src=cvLoadImage(argv[1], 1)) != 0 )
    {
        IplImage* dst = cvCreateImage( cvSize(256, 256), 8, 3 );
        IplImage* src2 = cvCreateImage( cvGetSize(src), 8, 3 );
        cvLogPolar( src, dst,
```

```
cvPoint2D32f(src->width/2, src->height/2), 40,  
CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS );  
    cvLogPolar( dst, src2,  
cvPoint2D32f(src->width/2, src->height/2), 40,  
CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS+CV_WARP_INVERSE_MAP );  
    cvNamedWindow( "log-polar", 1 );  
    cvShowImage( "log-polar", dst );  
    cvNamedWindow( "inverse log-polar", 1 );  
    cvShowImage( "inverse log-polar", src2 );  
    cvWaitKey();  
}  
return 0;  
}
```

对图像进行形态学操作（图像的开闭，腐蚀和膨胀运算）

效果图：（什么东东长这么丑啊，汗）



```
#include <cv.h>  
#include <highgui.h>  
#include <stdlib.h>  
#include <stdio.h>  
  
IplImage* src = 0;  
IplImage* dst = 0;
```

```
IplConvKernel* element = 0;
int element_shape = CV_SHAPE_RECT;

//the address of variable which receives trackbar position update
int max_iters = 10;
int open_close_pos = 0;
int erode_dilate_pos = 0;

// callback function for open/close trackbar
void OpenClose(int pos)
{
    int n = open_close_pos - max_iters;
    int an = n > 0 ? n : -n;
    element = cvCreateStructuringElementEx( an*2+1, an*2+1, an, an,
element_shape, 0 );
    if( n < 0 )
    {
        cvErode(src,dst,element,1);
        cvDilate(dst,dst,element,1);
    }
    else
    {
        cvDilate(src,dst,element,1);
        cvErode(dst,dst,element,1);
    }
    cvReleaseStructuringElement(&element);
    cvShowImage("Open/Close",dst);
}

// callback function for erode/dilate trackbar
void ErodeDilate(int pos)
{
    int n = erode_dilate_pos - max_iters;
    int an = n > 0 ? n : -n;
    element = cvCreateStructuringElementEx( an*2+1, an*2+1, an, an,
element_shape, 0 );
    if( n < 0 )
    {
        cvErode(src,dst,element,1);
    }
    else
    {
        cvDilate(src,dst,element,1);
    }
    cvReleaseStructuringElement(&element);
```

```
        cvShowImage("Erode/Dilate", dst);  
    }  
  
int main( int argc, char** argv )  
{  
    char* filename = argc == 2 ? argv[1] : (char*)"baboon.jpg";  
    if( (src = cvLoadImage(filename, 1)) == 0 )  
        return -1;  
  
    printf( "Hot keys: \n"  
            "\tESC - quit the program\n"  
            "\tr - use rectangle structuring element\n"  
            "\te - use elliptic structuring element\n"  
            "\tc - use cross-shaped structuring element\n"  
            "\tENTER - loop through all the options\n" );  
  
    dst = cvCloneImage(src);  
  
    //create windows for output images  
    cvNamedWindow("Open/Close", 1);  
    cvNamedWindow("Erode/Dilate", 1);  
  
    open_close_pos = erode_dilate_pos = max_iters;  
    cvCreateTrackbar("iterations",  
"Open/Close",&open_close_pos,max_iters*2+1,OpenClose);  
    cvCreateTrackbar("iterations",  
"Erode/Dilate",&erode_dilate_pos,max_iters*2+1,ErodeDilate);  
  
    for(;;)  
    {  
        int c;  
  
        OpenClose(open_close_pos);  
        ErodeDilate(erode_dilate_pos);  
        c = cvWaitKey(0);  
  
        if( (char)c == 27 )  
            break;  
        if( (char)c == 'e' )  
            element_shape = CV_SHAPE_ELLIPSE;  
        else if( (char)c == 'r' )  
            element_shape = CV_SHAPE_RECT;  
        else if( (char)c == 'c' )  
            element_shape = CV_SHAPE_CROSS;
```

```
        else if( (char)c == '\n' )
            element_shape = (element_shape + 1) % 3;
    }

//release images
cvReleaseImage(&src);
cvReleaseImage(&dst);

//destroy windows
cvDestroyWindow("Open/Close");
cvDestroyWindow("Erode/Dilate");

return 0;
}
```

用不同的核进行图像的二维滤波

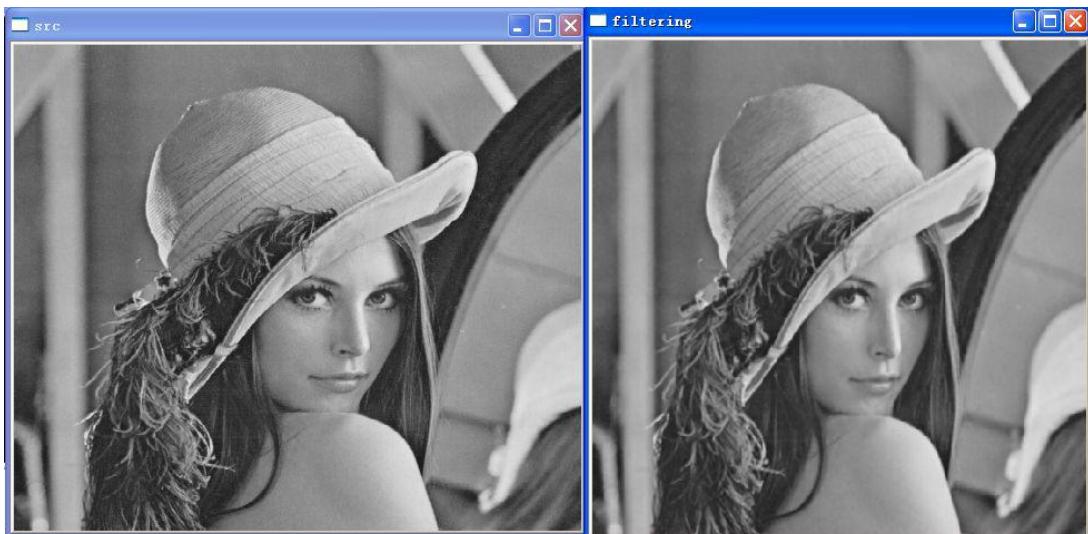
函数 cvSmooth 实现各种方法的图形平滑。

一般来说，图像平滑主要是为了消除噪声。图像的常见噪声主要有加性噪声、乘性噪声和量化噪声等。由于图像的能量主要集在低频部分，而噪声所在频段主要在高频段，因此通常都是采用低通滤波的方法消除噪声。

函数 cvFilter2D 对图像做卷积运算。

对图像进行线性滤波，支持替换方式操作。当核运算部份超出输入图像时，边界外面的像素值等于离它最近的图像像素值。

效果图：



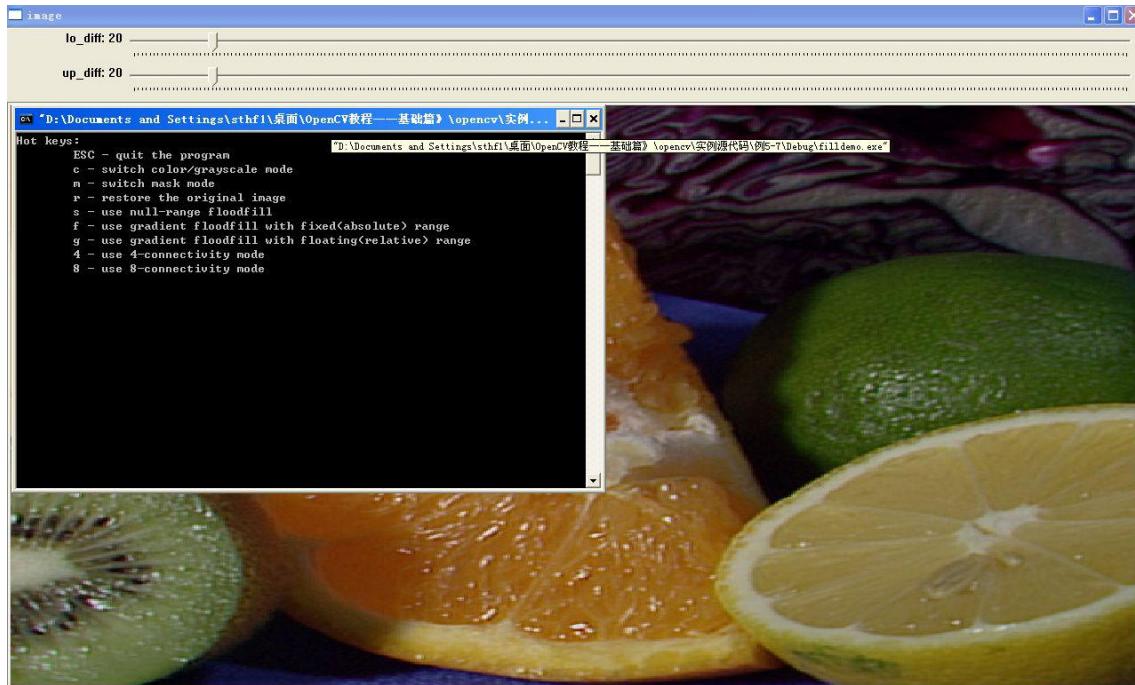
源代码：

```
// Filtering for Image with variaty filtering kernel
//
// CV_PREWITT_3x3_V A gradient filter (vertical Prewitt operator).
//          -1 0 1
//          -1 0 1
//          -1 0 1
// CV_PREWITT_3x3_H A gradient filter (horizontal Prewitt operator).
//          1 1 1
//          0 0 0
//          -1 -1 -1
// CV_SOBEL_3x3_V A gradient filter (vertical Sobel operator).
//          -1 0 1
//          -2 0 2
//          -1 0 1
// CV_SOBEL_3x3_H A gradient filter (horizontal Sobel operator).
//          1 2 1
//          0 0 0
//          -1 -2 -1
// CV_LAPLACIAN_3x3 A 3x3 Laplacian highpass filter.
//          -1 -1 -1
//          -1 8 -1
//          -1 -1 -1
// CV_LAPLACIAN_3x3 A 3x3 Laplacian highpass filter (another kernel)
// This kernel is similar with function: cvLaplace with aperture_size=1
//          0 1 0
//          1 -4 1
//          0 1 0           注：直接用 cvFilter2D 得到的结果与
用 cvLaplace 得到的结果
//                                     略有不同
// CV_LAPLACIAN_5x5 A 5x5 Laplacian highpass filter.
//          -1 -3 -4 -3 -1
//          -3 0 6 0 -3
//          -4 6 20 6 -4
//          -3 0 6 0 -3
//          -1 -3 -4 -3 -1
// CV_GAUSSIAN_3x3 A 3x3 Gaussian lowpass filter.
// This filter uses the kernel A/16, where
//          1 2 1
//          A = 2 4 2
//          1 2 1
// These filter coefficients correspond to a 2-dimensional Gaussian
// distribution with standard deviation 0.85.
```

```
//  
// CV_GAUSSIAN_5x5 A 5x5 Gaussian lowpass filter.  
// This filter uses the kernel A/571, where  
//          2 7 12 7 2  
//          7 31 52 31 7  
//          A = 12 52 127 52 12  
//          7 31 52 31 7  
//          2 7 12 7 2  
  
#include <cv.h>  
#include <highgui.h>  
#include <stdio.h>  
  
int main( int argc, char** argv )  
{  
    IplImage *src = 0, *dst = 0, *dst2 = 0;  
    /*float k[9] = { 0, 1, 0,  
                   1,-4, 1,  
                   0, 1, 0}; */  
    float k[9] = { 1.f/16, 2.f/16, 1.f/16,  
                  2.f/16, 4.f/16, 2.f/16,  
                  1.f/16, 2.f/16, 1.f/16}; //  
    //这里高斯核滤波器归一化  
    CvMat Km;  
    //cvInitMatHeader( &Km, 3, 3, CV_32FC1, k, CV_AUTOSTEP );  
    Km = cvMat( 3, 3, CV_32F, k );  
  
    // 0: force to gray image  
    src = cvLoadImage("lena.jpg", 0);  
    dst = cvCloneImage( src );  
  
    cvNamedWindow("src", 0);  
    cvShowImage("src", src);  
  
    cvNamedWindow("filtering", 0);  
    cvFilter2D( src, dst, &Km, cvPoint(-1,-1));  
    cvShowImage("filtering", dst);  
    cvWaitKey(0);  
  
    cvReleaseImage( &src );  
    cvReleaseImage( &dst );  
    return 0;  
}
```

图像域的填充

效果图：



源代码：

```
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <stdlib.h>

IplImage* color_img0;
IplImage* mask;
IplImage* color_img;
IplImage* gray_img0 = NULL;
IplImage* gray_img = NULL;
int ffill_case = 1;
int lo_diff = 20, up_diff = 20;
int connectivity = 4;
int is_color = 1;
int is_mask = 0;
int new_mask_val = 255;

void on_mouse( int event, int x, int y, int flags, void* param )
{
```

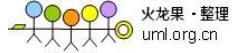
```
if( !color_img )
    return;

switch( event )
{
case CV_EVENT_LBUTTONDOWN:
{
    CvPoint seed = cvPoint(x,y);
    int lo = ffill_case == 0 ? 0 : lo_diff;
    int up = ffill_case == 0 ? 0 : up_diff;
    int flags = connectivity + (new_mask_val << 8) +
                (ffill_case == 1 ?
CV_FLOODFILL_FIXED_RANGE : 0);
    int b = rand() & 255, g = rand() & 255, r = rand()
& 255;
    CvConnectedComp comp;

    if( is_mask )
        cvThreshold( mask, mask, 1, 128,
CV_THRESH_BINARY );

    if( is_color )
    {
        CvScalar color = CV_RGB( r, g, b );
        cvFloodFill( color_img, seed, color,
CV_RGB( lo, lo, lo ),
CV_RGB( up,
up, up ), &comp, flags, is_mask ? mask : NULL );
        cvShowImage( "image", color_img );
    }
    else
    {
        CvScalar brightness = cvRealScalar((r*2
+ g*7 + b + 5)/10);
        cvFloodFill( gray_img, seed, brightness,
cvRealScalar(lo),
cvRealScalar
(up), &comp, flags, is_mask ? mask : NULL );
        cvShowImage( "image", gray_img );
    }

    printf("%g pixels were repainted\n",
comp.area );
}
```



```

        if( is_mask )
            cvShowImage( "mask", mask );
    }
    break;
}

int main( int argc, char** argv )
{
    char* filename = argc >= 2 ? argv[1] : (char*)"fruits.jpg";

    if( (color_img0 = cvLoadImage(filename, 1)) == 0 )
        return 0;

    printf( "Hot keys: \n"
            "\tESC - quit the program\n"
            "\t\tc - switch color/grayscale mode\n"
            "\t\tm - switch mask mode\n"
            "\t\tr - restore the original image\n"
            "\t\ts - use null-range floodfill\n"
            "\t\tf - use gradient floodfill with fixed(absolute)
range\n"
            "\t\tg - use gradient floodfill with
floating(relative) range\n"
            "\t\t4 - use 4-connectivity mode\n"
            "\t\t8 - use 8-connectivity mode" );
}

color_img = cvCloneImage( color_img0 );
gray_img0 = cvCreateImage( cvSize(color_img->width,
color_img->height), 8, 1 );
cvCvtColor( color_img, gray_img0, CV_BGR2GRAY );
gray_img = cvCloneImage( gray_img0 );
mask = cvCreateImage( cvSize(color_img->width + 2,
color_img->height + 2), 8, 1 );

cvNamedWindow( "image", 0 );
cvCreateTrackbar( "lo_diff", "image", &lo_diff, 255, NULL );
cvCreateTrackbar( "up_diff", "image", &up_diff, 255, NULL );

cvSetMouseCallback( "image", on_mouse, 0 );

for(;;)
{
    int c;

```

```
if( is_color )
    cvShowImage( "image", color_img );
else
    cvShowImage( "image", gray_img );

c = cvWaitKey(0);
switch( (char) c )
{
case '\x1b':
    printf("Exiting ... \n");
    goto exit_main;
case 'c':
    if( is_color )
    {
        printf("Grayscale mode is set\n");
        cvCvtColor( color_img, gray_img,
CV_BGR2GRAY );
        is_color = 0;
    }
    else
    {
        printf("Color mode is set\n");
        cvCopy( color_img0, color_img, NULL );
        cvZero( mask );
        is_color = 1;
    }
    break;
case 'm':
    if( is_mask )
    {
        cvDestroyWindow( "mask" );
        is_mask = 0;
    }
    else
    {
        cvNamedWindow( "mask", 0 );
        cvZero( mask );
        cvShowImage( "mask", mask );
        is_mask = 1;
    }
    break;
case 'r':
    printf("Original image is restored\n");
    cvCopy( color_img0, color_img, NULL );
```

```
        cvCopy( gray_img0, gray_img, NULL );
        cvZero( mask );
        break;
    case 's':
        printf("Simple floodfill mode is set\n");
        ffill_case = 0;
        break;
    case 'f':
        printf("Fixed Range floodfill mode is set\n");
        ffill_case = 1;
        break;
    case 'g':
        printf("Gradient (floating range) floodfill mode
is set\n");
        ffill_case = 2;
        break;
    case '4':
        printf("4-connectivity mode is set\n");
        connectivity = 4;
        break;
    case '8':
        printf("8-connectivity mode is set\n");
        connectivity = 8;
        break;
    }
}

exit_main:

cvDestroyWindow( "test" );
cvReleaseImage( &gray_img );
cvReleaseImage( &gray_img0 );
cvReleaseImage( &color_img );
cvReleaseImage( &color_img0 );
cvReleaseImage( &mask );

return 1;
}
```

寻找轮廓实现视频流的运动目标检测（超推荐一下）

效果视频我上传了，浏览网址（个人感觉很牛，有点像生化危机里的一个场景）：

<http://tinypic.com/m/a2epo8/2>

如果上面的卡，可以连这个，就是有点发散图形：

<http://i41.tinypic.com/54xcsml.jpg>

也不说什么了，直接给代码吧（有一句话想说，实际上如果你是拿来做实际项目的，可能并不要学习里面的算法，直接利用里面的模板，也就是外接的调用函数就可以了）：

```
#include "cv.h"
#include "highgui.h"
#include <time.h>
#include <math.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

// various tracking parameters (in seconds)
const double MHI_DURATION = 0.5;
const double MAX_TIME_DELTA = 0.5;
const double MIN_TIME_DELTA = 0.05;
const int N = 3;

// 
const int CONTOUR_MAX_AERA = 16;

// ring image buffer
IplImage **buf = 0;
int last = 0;

// temporary images
IplImage *mhi = 0; // MHI: motion history image

CvFilter filter = CV_GAUSSIAN_5x5;
CvConnectedComp *cur_comp, min_comp;
CvConnectedComp comp;
CvMemStorage *storage;
CvPoint pt[4];

// 参数:
// img - 输入视频帧
// dst - 检测结果
```

```
void update_mhi( IplImage* img, IplImage* dst, int diff_threshold )
{
    double timestamp = clock() / 100.; // get current time in seconds
    CvSize size = cvSize(img->width, img->height); // get current
frame size
    int i, j, idx1, idx2;
    IplImage* silh;
    uchar val;
    float temp;
    IplImage* pyr = cvCreateImage( cvSize((size.width & -2)/2,
(size.height & -2)/2), 8, 1 );
    CvMemStorage *stor;
    CvSeq *cont, *result, *squares;
    CvSeqReader reader;

    if( !mhi || mhi->width != size.width || mhi->height !=
size.height )
    {
        if( buf == 0 )
        {
            buf = (IplImage**)malloc(N*sizeof(buf[0]));
            memset( buf, 0, N*sizeof(buf[0]));
        }

        for( i = 0; i < N; i++ )
        {
            cvReleaseImage( &buf[i] );
            buf[i] = cvCreateImage( size, IPL_DEPTH_8U, 1 );
            cvZero( buf[i] );
        }
        cvReleaseImage( &mhi );
        mhi = cvCreateImage( size, IPL_DEPTH_32F, 1 );
        cvZero( mhi ); // clear MHI at the beginning
    } // end of if(mhi)

    cvCvtColor( img, buf[last], CV_BGR2GRAY ); // convert frame to
grayscale

    idx1 = last;
    idx2 = (last + 1) % N; // index of (last - (N-1))th frame
    last = idx2;

    // 做帧差
    silh = buf[idx2];
    cvAbsDiff( buf[idx1], buf[idx2], silh ); // get difference between
```

frames

```
// 对差图像做二值化
cvThreshold( silh, silh, 30, 255, CV_THRESH_BINARY ); // and
threshold it

cvUpdateMotionHistory( silh, mhi, timestamp, MHI_DURATION ); // /
update MHI
cvCvtScale( mhi, dst, 255./MHI_DURATION,
(MHI_DURATION - timestamp)*255./MHI_DURATION );
cvCvtScale( mhi, dst, 255./MHI_DURATION, 0 );

// 中值滤波，消除小的噪声
cvSmooth( dst, dst, CV_MEDIAN, 3, 0, 0, 0 );

// 向下采样，去掉噪声
cvPyrDown( dst, pyr, 7 );
cvDilate( pyr, pyr, 0, 1 ); // 做膨胀操作，消除目标的不连续空洞
cvPyrUp( pyr, dst, 7 );
//
// 下面的程序段用来找到轮廓
//
// Create dynamic structure and sequence.
stor = cvCreateMemStorage(0);
cont = cvCreateSeq(CV_SEQ_ELTYPE_POINT, sizeof(CvSeq),
sizeof(CvPoint) , stor);

// 找到所有轮廓
cvFindContours( dst, stor, &cont, sizeof(CvContour),
CV_RETR_LIST,
CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0));
/*
for(;cont;cont = cont->h_next)
{
    // Number point must be more than or equal to 6 (for
    cvFitEllipse_32f).
    if( cont->total < 6 )
        continue;

    // Draw current contour.
    cvDrawContours(img, cont, CV_RGB(255,0,0), CV_RGB(255,0,0)
, 0, 1, 8, cvPoint(0,0));
} // end of for-loop: "cont"
*/
```

```

// 直接使用 CONTOUR 中的矩形来画轮廓
for(;cont;cont = cont->h_next)
{
    CvRect r = ((CvContour*)cont)->rect;
    if(r.height * r.width > CONTOUR_MAX_AERA) // 面积小的方形抛弃掉
    {
        cvRectangle( img, cvPoint(r.x, r.y),
                     cvPoint(r.x + r.width, r.y + r.height),
                     CV_RGB(255, 0, 0), 1,
                     CV_AA, 0 );
    }
}
// free memory
cvReleaseMemStorage(&stor);
cvReleaseImage( &pyr );
}

int main(int argc, char** argv)
{
    IplImage* motion = 0;
    CvCapture* capture = 0; //视频获取结构

    if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 && isdigit(argv[1][0])))
        //原型: extern int isdigit(char c);
        //用法: #include <ctype.h>      功能: 判断字符 c 是否为数字      说明:
        当 c 为数字 0-9 时, 返回非零值, 否则返回零。
        capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' :
0 );
    else if( argc == 2 )
        capture = cvCaptureFromAVI( argv[1] );
    if( capture )
    {
        cvNamedWindow( "Motion", 1 );
        for(;;)
        {
            IplImage* image;
            if( !cvGrabFrame( capture ) ) //从摄像头或者视频文件中抓取帧
                break;
            image = cvRetrieveFrame( capture ); //取回由函数 cvGrabFrame 抓取的图像, 返回由函数 cvGrabFrame 抓取的图像的指针
        }
    }
}

```

```
if( image )
{
    if( !motion )
    {
        motion =
cvCreateImage( cvSize(image->width, image->height), 8, 1 );
        cvZero( motion );
        motion->origin = image->origin;
/* 0 - 顶一左结构, 1 - 底一左结构 (Windows bitmaps 风格) */
    }
}

update_mhi( image, motion, 60 );
cvShowImage( "Motion", image );

if( cvWaitKey(10) >= 0 )
    break;
}
cvReleaseCapture( &capture );
cvDestroyWindow( "Motion" );
}
return 0;
}
```

采用金字塔方法进行图像分割

图像分割指的是将数字图像细分为多个图像子区域的过程，在OpenCv中实现了三种跟图像分割相关的算法，它们分别是：分水岭分割算法、金字塔分割算法以及均值漂移分割算法。

分水岭分割算法

分水岭分割算法需要您或者先前算法提供标记，该标记用于指定哪些大致区域是目标，哪些大致区域是背景等等；分水岭分割算法的分割效果严重依赖于提供的标记。OpenCv中的函数cvWatershed实现了该算法

金字塔分割算法

金字塔分割算法由cvPrySegmentation所实现，该函数的使用很简单；需要注意的是图像的尺寸以及金字塔的层数，图像的宽度和高度必须能被2整除，能够被2整除的次数决定了金字塔的最大层数

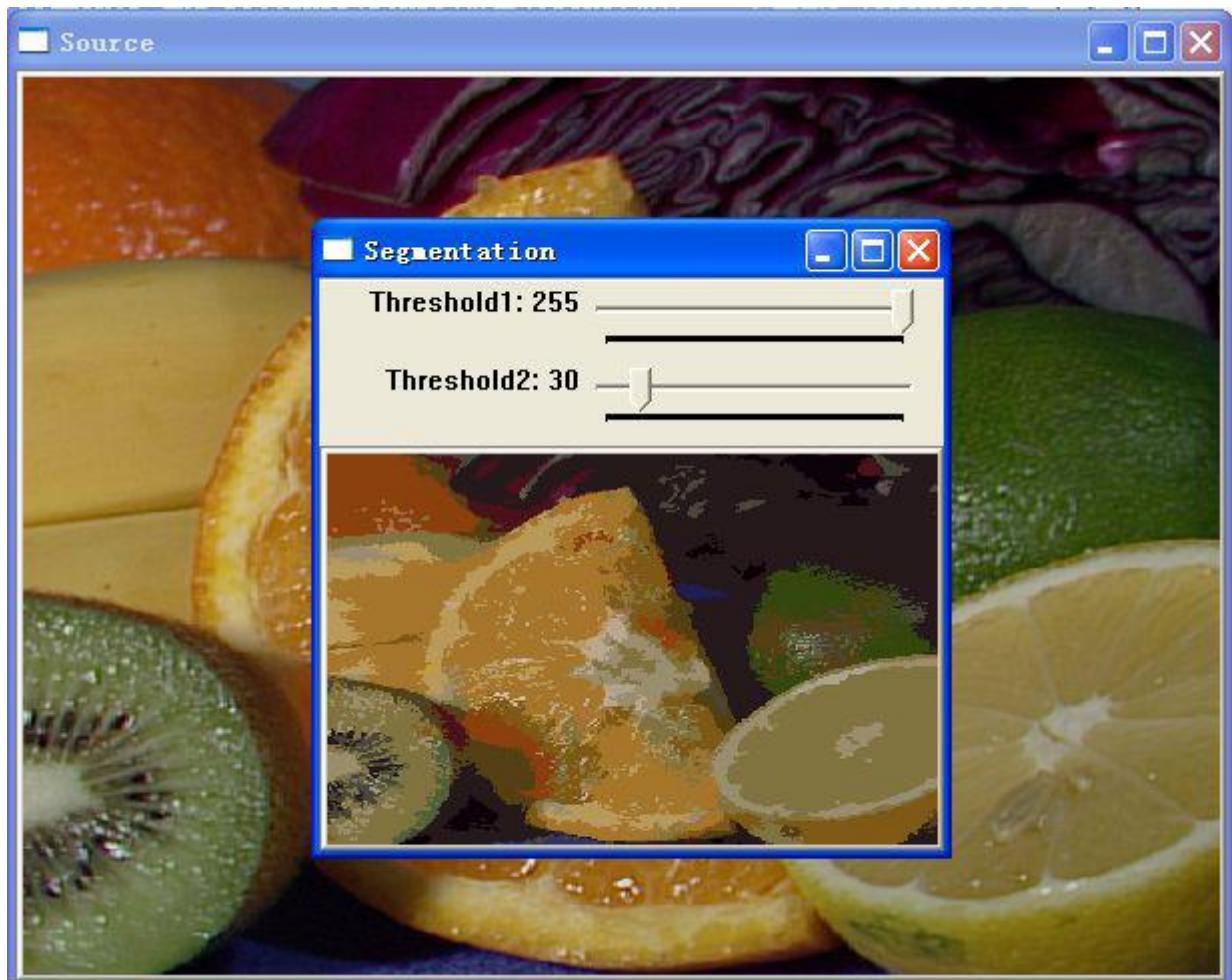
均值漂移分割算法

均值漂移分割算法由cvPryMeanShiftFiltering所实现，均值漂移分割的金字塔层数只能介于[1, 7]之间

友情链接一下，个人感觉比较好的这方面博客：

<http://www.cnblogs.com/xrwang/archive/2010/02/28/ImageSegmentation.html>

效果图：



```
#include "cv.h"
#include "highgui.h"
#include <math.h>

IplImage* image[2] = { 0, 0 }, *image0 = 0, *image1 = 0;
CvSize size;

int w0, h0, i;
int threshold1, threshold2;
int l, level = 4;
int sthreshold1, sthreshold2;
int l_comp;
int block_size = 1000;
```

```
float parameter;
double threshold;
double result, min_result;
CvFilter filter = CV_GAUSSIAN_5x5;
CvConnectedComp *cur_comp, min_comp;
CvSeq *comp;
CvMemStorage *storage;

CvPoint pt1, pt2;

void ON_SEGMENT(int a)
{
    cvPyrSegmentation(image0, image1, storage, &comp,
                       level, threshold1+1,
    threshold2+1);

    /*l_comp = comp->total;

    i = 0;
    min_comp.value = cvScalarAll(0);
    while(i<l_comp)
    {
        cur_comp = (CvConnectedComp*)cvGetSeqElem ( comp, i );
        if(fabs(255- min_comp.value.val[0])>
           fabs(255- cur_comp->value.val[0]) &&
           fabs(min_comp.value.val[1])>
           fabs(cur_comp->value.val[1]) &&
           fabs(min_comp.value.val[2])>
           fabs(cur_comp->value.val[2]) )
            min_comp = *cur_comp;
        i++;
    }*/
    cvShowImage("Segmentation", image1);
}

int main( int argc, char** argv )
{
    char* filename = argc == 2 ? argv[1] : (char*)"fruits.jpg";
    if( (image[0] = cvLoadImage( filename, 1)) == 0 )
        return -1;
    cvNamedWindow("Source", 0);
    cvShowImage("Source", image[0]);

    cvNamedWindow("Segmentation", 0);
```

```
storage = cvCreateMemStorage ( block_size );

image[0]->width &= -(1<<level);
image[0]->height &= -(1<<level);

image0 = cvCloneImage( image[0] );
image1 = cvCloneImage( image[0] );
// 对彩色图像进行分割
l = 1;
threshold1 =255;
threshold2 =30;

ON_SEGMENT(1);

sthreshold1 = cvCreateTrackbar("Threshold1", "Segmentation",
&threshold1, 255,
ON_SEGMENT);
sthreshold2 = cvCreateTrackbar("Threshold2", "Segmentation",
&threshold2, 255,
ON_SEGMENT);

cvShowImage("Segmentation", image1);
cvWaitKey(0);

cvDestroyWindow("Segmentation");
cvDestroyWindow("Source");

cvReleaseMemStorage(&storage );

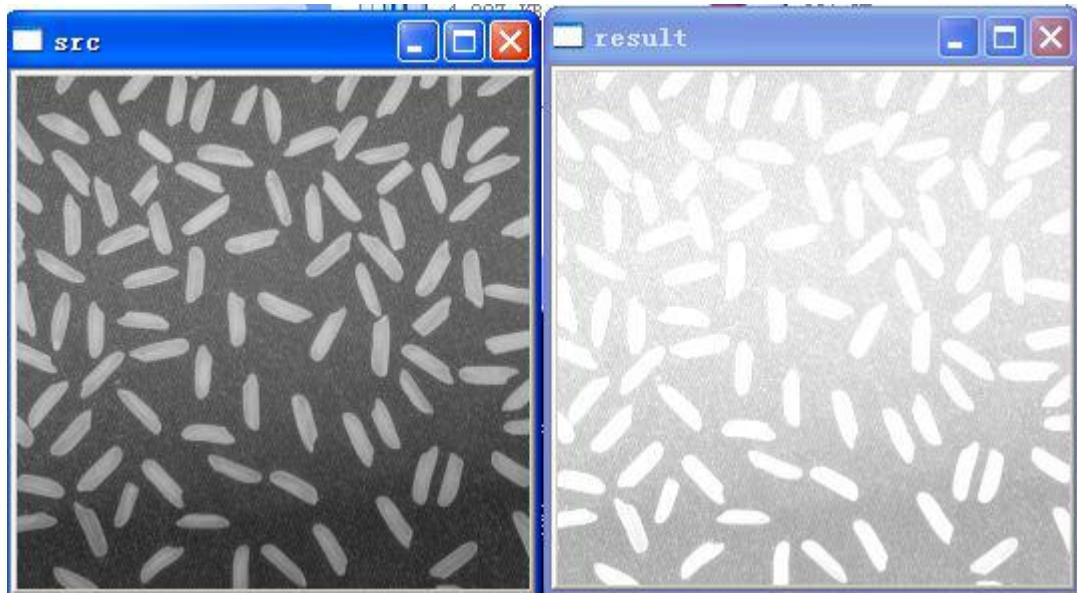
cvReleaseImage(&image[0]);
cvReleaseImage(&image0);
cvReleaseImage(&image1);

return 0;
}
```

图像的亮度变换

郁闷，以前用过 MatLab，很长时间没用了，都不知道怎么使了，据说做这个效果很不错。

效果图：



源代码:

```
#include "cv.h"
#include "highgui.h"
/*
src and dst are grayscale, 8-bit images;
Default input value:
    [low, high] = [0, 1]; X-Direction
    [bottom, top] = [0, 1]; Y-Direction
    gamma ;
if adjust successfully, return 0, otherwise, return non-zero.
*/
int ImageAdjust(IplImage* src, IplImage* dst,
    double low, double high,      // X 方向: low and high are the
    intensities of src
    double bottom, double top, // Y 方向: mapped to bottom and top of
dst
    double gamma )
{
if( low<0 && low>1 && high <0 && high>1&&
bottom<0 && bottom>1 && top<0 && top>1 && low>high)
    return -1;
double low2 = low*255;
double high2 = high*255;
double bottom2 = bottom*255;
double top2 = top*255;
double err_in = high2 - low2;
double err_out = top2 - bottom2;
```

```
int x, y;
double val;

// intensity transform
for( y = 0; y < src->height; y++)
{
    for (x = 0; x < src->width; x++)
    {
        val = ((uchar*) (src->imageData +
src->widthStep*y)) [x];
        val = pow((val - low2)/err_in, gamma) * err_out
+ bottom2;
        if(val>255) val=255; if(val<0) val=0; // Make
sure src is in the range [low,high]
        ((uchar*) (dst->imageData + dst->widthStep*y)) [x]
= (uchar) val;
    }
    return 0;
}

int main( int argc, char** argv )
{
    IplImage *src = 0, *dst = 0;

    if( argc != 2 || (src=cvLoadImage(argv[1], 0)) == NULL) // force
to gray image
        return -1;

    cvNamedWindow( "src", 1 );
    cvNamedWindow( "result", 1 );

    // Image adjust
    dst = cvCloneImage(src);
    // 输入参数 [0,0.5] 和 [0.5,1], gamma=1
    if( ImageAdjust( src, dst, 0, 0.5, 0.5, 1, 1)!=0) return -1;

    cvShowImage( "src", src );
    cvShowImage( "result", dst );
    cvWaitKey(0);

    cvDestroyWindow("src");
    cvDestroyWindow("result");
    cvReleaseImage( &src );
    cvReleaseImage( &dst );
```

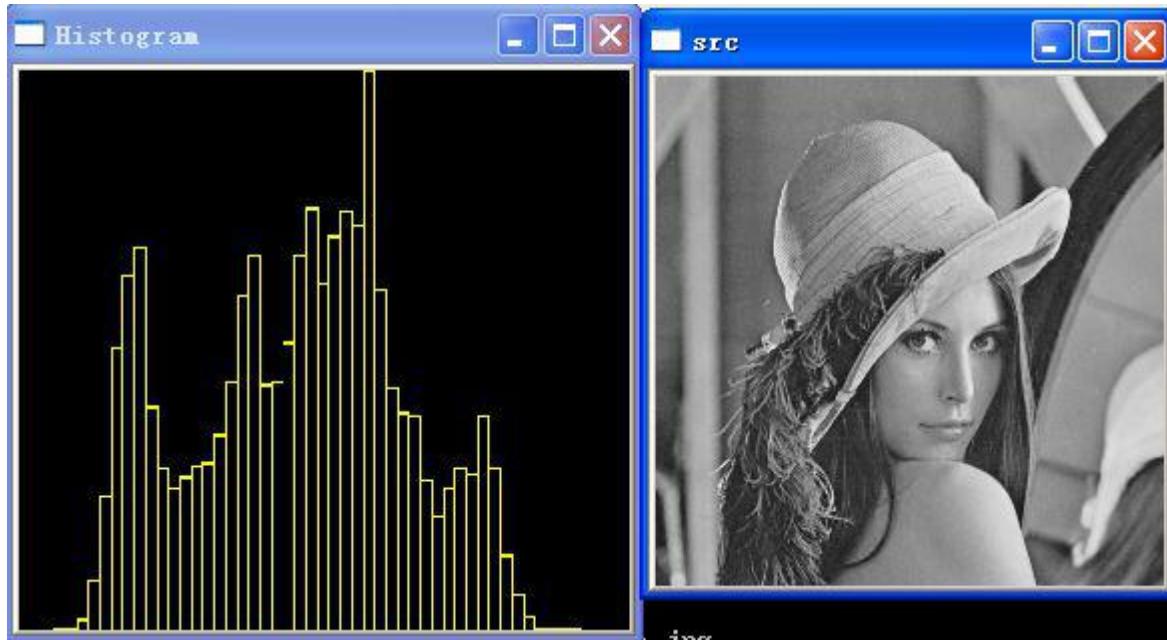
```
    return 0;  
}
```

单通道图像的直方图

原始图：



效果图：



源代码：

```
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <ctype.h>

int main( int argc, char** argv )
{
    IplImage *src = 0;
    IplImage *histimg = 0;
    CvHistogram *hist = 0;

    int hdims = 50;          // 划分 HIST 的个数，越高越精确
    float hranges_arr[] = {0, 255};
    float* hranges = hranges_arr;
    int bin_w;
    float max_val;
    int i;

    if( argc != 2 || (src=cvLoadImage(argv[1], 0)) == NULL) // force
to gray image
        return -1;

    cvNamedWindow( "Histogram", 0 );
    cvNamedWindow( "src", 0 );

    hist = cvCreateHist( 1, &hdims, CV_HIST_ARRAY, &hranges, 1 ); //
```

计算直方图

```
histimg = cvCreateImage( cvSize(320, 200), 8, 3 );
cvZero( histimg );
cvCalcHist( &src, hist, 0, 0 ); // 计算直方图
cvGetMinMaxHistValue( hist, 0, &max_val, 0, 0 ); // 只找最大值
cvConvertScale( hist->bins,
hist->bins, max_val ? 255. / max_val : 0., 0 ); // 缩放 bin 到区间 [0, 255]
cvZero( histimg );
bin_w = histimg->width / hdims; // hdims: 条的个数, 则 bin_w 为
条的宽度

// 画直方图
for( i = 0; i < hdims; i++ )
{
    double val =
( cvGetReal1D(hist->bins, i)*histimg->height/255 );
    CvScalar color = CV_RGB(255, 255, 0);
// (hsv2rgb(i*180. f/hdims));
    cvRectangle( histimg, cvPoint(i*bin_w, histimg->height),
                cvPoint((i+1)*bin_w, (int)(histimg->height -
val)), 
                color, 1, 8, 0 );
}

cvShowImage( "src", src );
cvShowImage( "Histogram", histimg );
cvWaitKey(0);

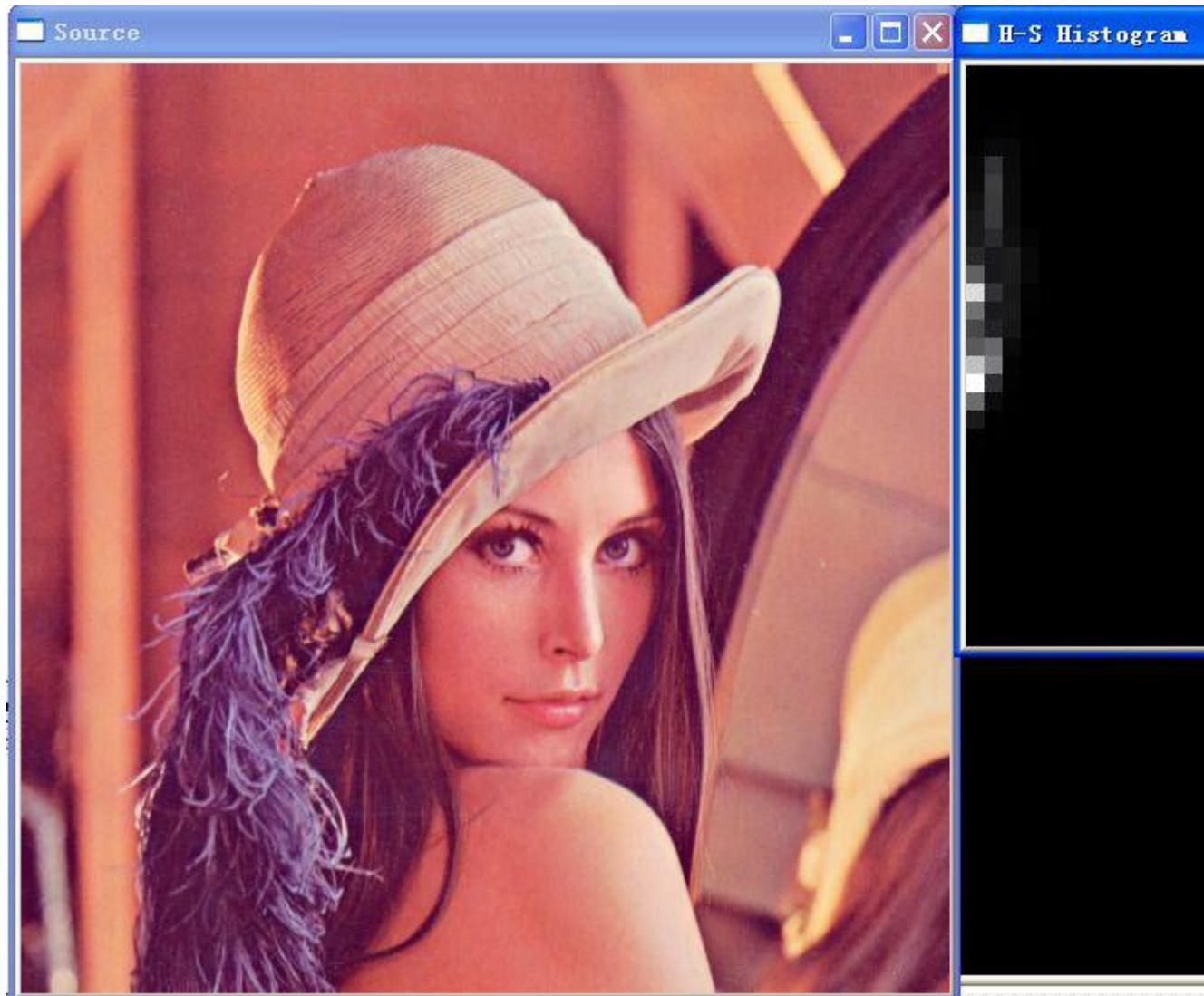
cvDestroyWindow("src");
cvDestroyWindow("Histogram");
cvReleaseImage( &src );
cvReleaseImage( &histimg );
cvReleaseHist ( &hist );

return 0;
}
```

计算和显示彩色图像的二维色调-饱和度图像

对这篇内容很郁闷，不知道以后用来干什么，声明一下，哥不是搞图像处理的。
(业余爱好)

效果图：（好好的一张图，给处理成人不像人，鬼不像鬼）



```
#include <cv.h>
#include <highgui.h>

int main( int argc, char** argv )
{
    IplImage* src;
    if( argc == 2 && (src=cvLoadImage(argv[1], 1))!= 0)
    {
        IplImage* h_plane = cvCreateImage( cvGetSize(src), 8, 1 );
        IplImage* s_plane = cvCreateImage( cvGetSize(src), 8, 1 );
        IplImage* v_plane = cvCreateImage( cvGetSize(src), 8, 1 );
        IplImage* planes[] = { h_plane, s_plane };
        IplImage* hsv = cvCreateImage( cvGetSize(src), 8, 3 );
        int h_bins = 30, s_bins = 32;
        int hist_size[] = {h_bins, s_bins};
        float h_ranges[] = { 0, 180 } ; /* hue varies from 0 (^0° red)
```

```
to 180 (^360° red again) */
    float s_ranges[] = { 0, 255 }; /* saturation varies from
0 (black-gray-white) to 255 (pure spectrum color) */
    float* ranges[] = { h_ranges, s_ranges };
    int scale = 10;
    IplImage* hist_img =
cvCreateImage( cvSize(h_bins*scale, s_bins*scale), 8, 3 );
    CvHistogram* hist;
    float max_value = 0;
    int h, s;

    cvCvtColor( src, hsv, CV_BGR2HSV );
    cvCvtPixToPlane( hsv, h_plane, s_plane, v_plane, 0 );
    hist = cvCreateHist( 2, hist_size, CV_HIST_ARRAY, ranges,
1 );
    cvCalcHist( planes, hist, 0, 0 );
    cvGetMinMaxHistValue( hist, 0, &max_value, 0, 0 );
    cvZero( hist_img );

    for( h = 0; h < h_bins; h++ )
    {
        for( s = 0; s < s_bins; s++ )
        {
            float bin_val =
cvQueryHistValue_2D( hist, h, s );
            int intensity =
cvRound(bin_val*255/max_value);
            cvRectangle( hist_img, cvPoint( h*scale,
s*scale ),
cvPoint( (h+1)*scale - 1,
(s+1)*scale - 1 ),
CV_RGB(intensity, intensity, intensity),
CV_FILLED );
        }
    }
    cvNamedWindow( "Source", 1 );
    cvShowImage( "Source", src );
    cvNamedWindow( "H-S Histogram", 1 );
    cvShowImage( "H-S Histogram", hist_img );
    cvWaitKey(0);
}
}
```

图像的直方图均匀化

直方图均衡化算法可以归一化图像的亮度，并增强图像的对比度

效果图：



源代码：

```
#include "cv.h"
#include "highgui.h"

#define HDIM      256          // bin of HIST, default = 256

int main( int argc, char** argv )
{
    IplImage *src = 0, *dst = 0;
    CvHistogram *hist = 0;

    int n = HDIM;
    double nn[HDIM];
    uchar T[HDIM];
    CvMat *T_mat;

    int x;
    int sum = 0; // sum of pixels of the source image 图像中象素点
    的总和
    double val = 0;
```

```
if( argc != 2 || (src=cvLoadImage(argv[1], 0)) == NULL) // force
to gray image
    return -1;

cvNamedWindow( "source", 1 );
cvNamedWindow( "result", 1 );

// 计算直方图
hist = cvCreateHist( 1, &n, CV_HIST_ARRAY, 0, 1 );
cvCalcHist( &src, hist, 0, 0 );

// Create Accumulative Distribute Function of histgram
val = 0;
for ( x = 0; x < n; x++)
{
    val = val + cvGetReal1D (hist->bins, x);
    nn[x] = val;
}

// 归一化直方图
sum = src->height * src->width;
for( x = 0; x < n; x++ )
{
    T[x] = (uchar) (255 * nn[x] / sum); // range is [0, 255]
}

// Using look-up table to perform intensity transform for source
image
dst = cvCloneImage( src );
T_mat = cvCreateMatHeader( 1, 256, CV_8UC1 );
cvSetData( T_mat, T, 0 );
// 直接调用内部函数完成 look-up-table 的过程
cvLUT( src, dst, T_mat );

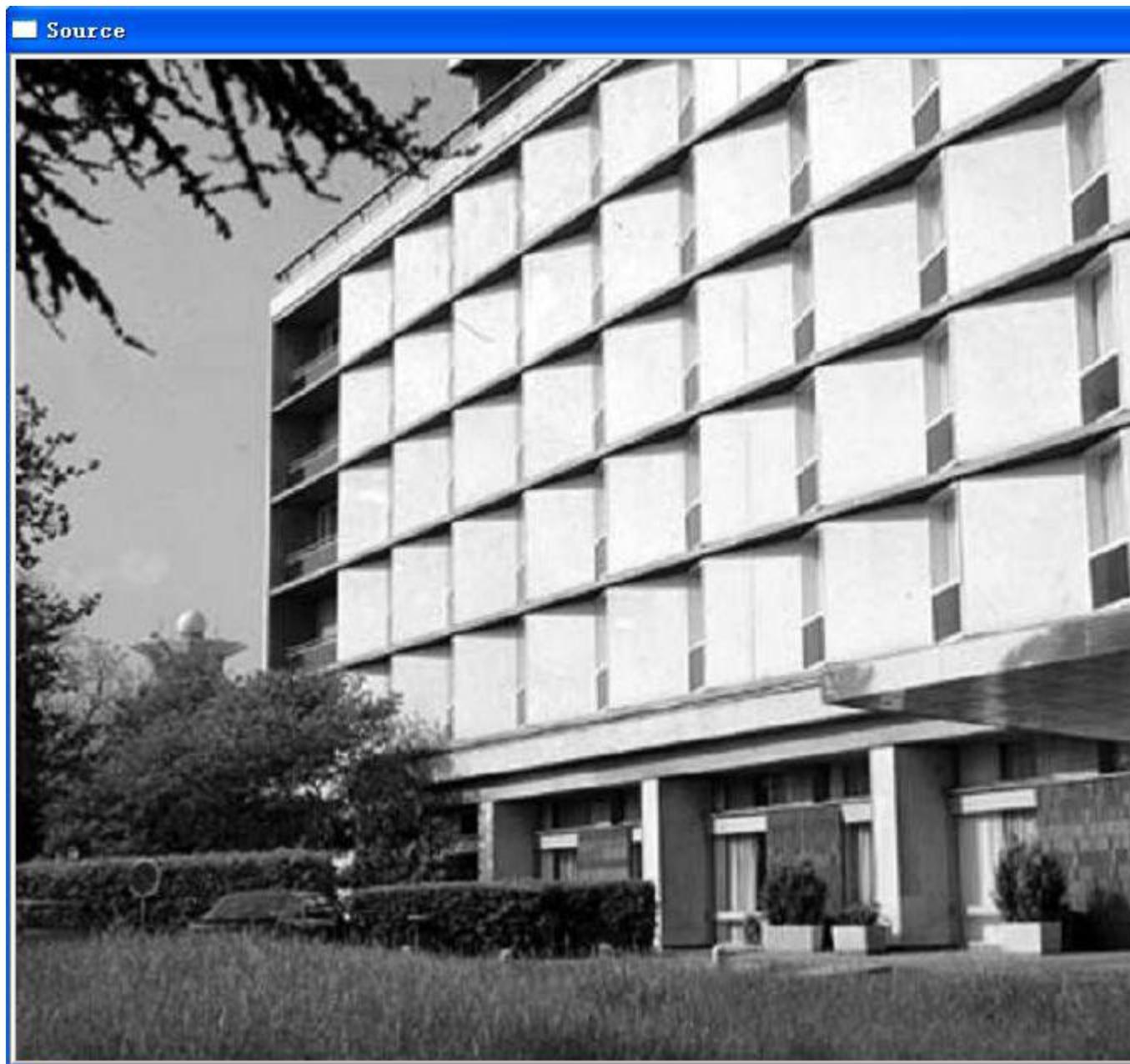
cvShowImage( "source", src );
cvShowImage( "result", dst );
cvWaitKey(0);

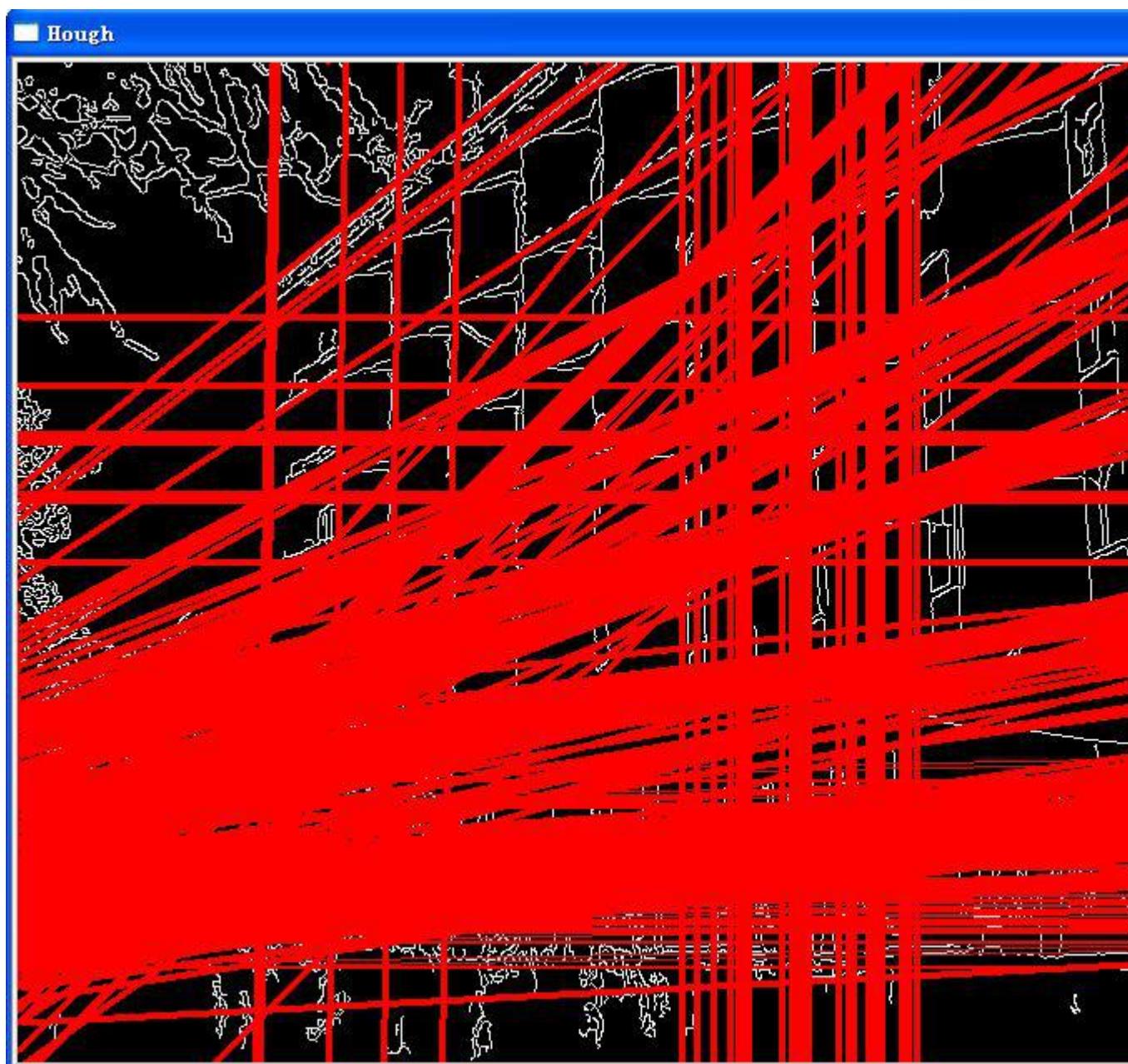
cvDestroyWindow("source");
cvDestroyWindow("result");
cvReleaseImage( &src );
cvReleaseImage( &dst );
cvReleaseHist ( &hist );
```

```
        return 0;  
    }
```

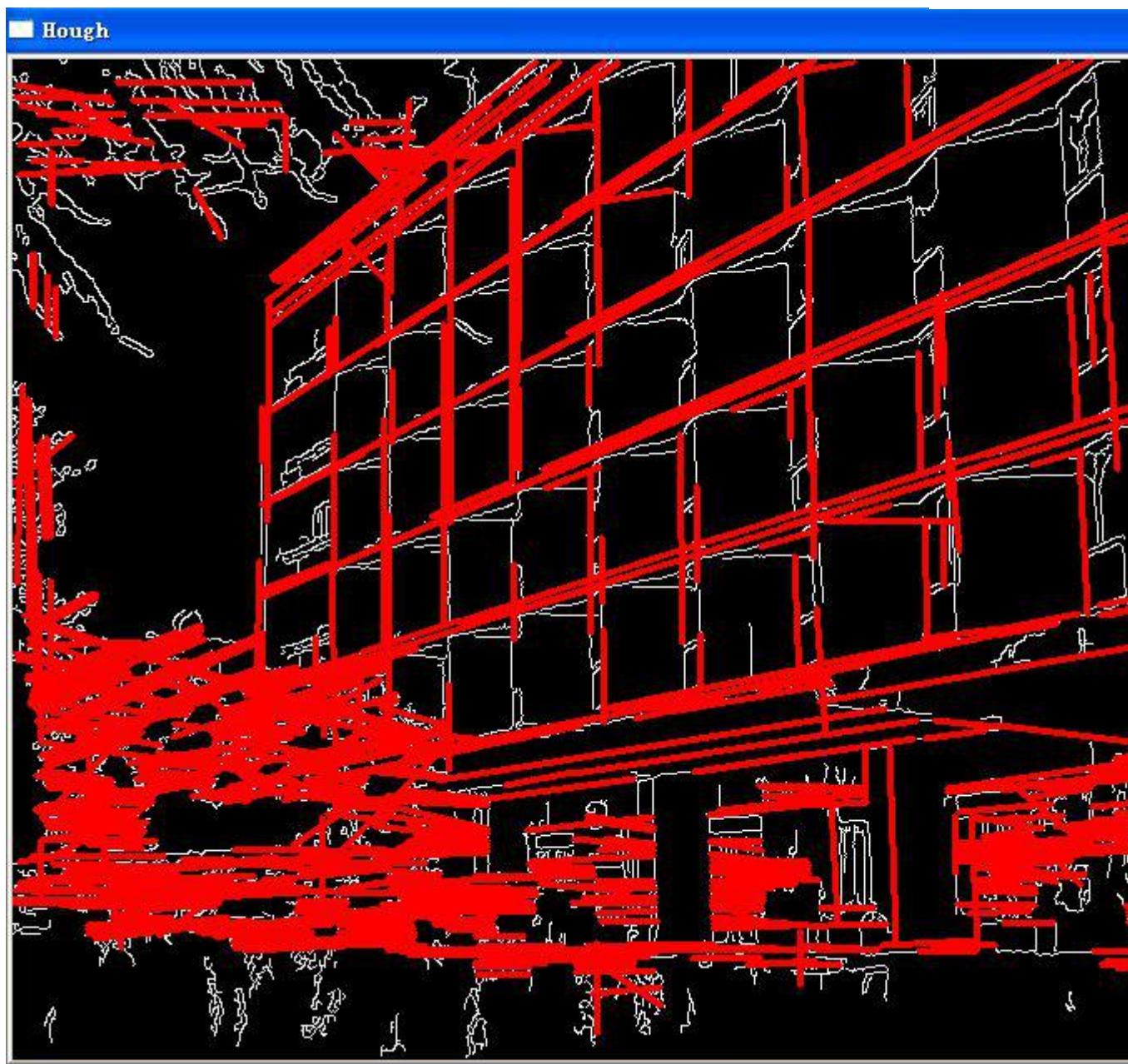
用 Hongh 变换检测线段

效果图（郁闷这么和我想的不一样啊，这是什么东东）：





这幅还像点东东：



源代码：

```
/* 这是一个命令行程序，以图像作为文件输入变量
   编译时选择 “#if 1” 或 “#if 0”，可以使用标准和概率 HOUGH 变换两种
   方法 */
#include <cv.h>
#include <highgui.h>
#include <math.h>

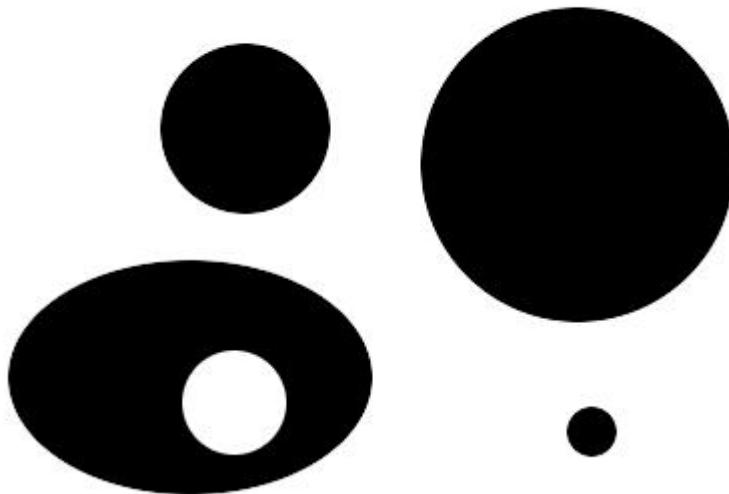
int main(int argc, char** argv)
{
    IplImage* src;
    if( argc == 2 && (src=cvLoadImage(argv[1], 0))!= 0)
```

```
{  
    IplImage* dst = cvCreateImage( cvGetSize(src), 8, 1 );  
    IplImage* color_dst = cvCreateImage( cvGetSize(src), 8,  
3 );  
    CvMemStorage* storage = cvCreateMemStorage(0);  
    CvSeq* lines = 0;  
    int i;  
    cvCanny( src, dst, 50, 200, 3 );  
    cvCvtColor( dst, color_dst, CV_GRAY2BGR );  
#if 1  
    lines = cvHoughLines2( dst, storage, CV_HOUGH_STANDARD,  
1, CV_PI/180, 150, 0, 0 );  
  
    for( i = 0; i < lines->total; i++ )  
    {  
        float* line = (float*)cvGetSeqElem(lines, i);  
        float rho = line[0];  
        float theta = line[1];  
        CvPoint pt1, pt2;  
        double a = cos(theta), b = sin(theta);  
        if( fabs(a) < 0.001 )  
        {  
            pt1.x = pt2.x = cvRound(rho);  
            pt1.y = 0;  
            pt2.y = color_dst->height;  
        }  
        else if( fabs(b) < 0.001 )  
        {  
            pt1.y = pt2.y = cvRound(rho);  
            pt1.x = 0;  
            pt2.x = color_dst->width;  
        }  
        else  
        {  
            pt1.x = 0;  
            pt1.y = cvRound(rho/b);  
            pt2.x = cvRound(rho/a);  
            pt2.y = 0;  
        }  
        cvLine( color_dst, pt1, pt2, CV_RGB(255,0,0), 3,  
8 );  
    }  
#else  
    lines = cvHoughLines2( dst, storage,
```

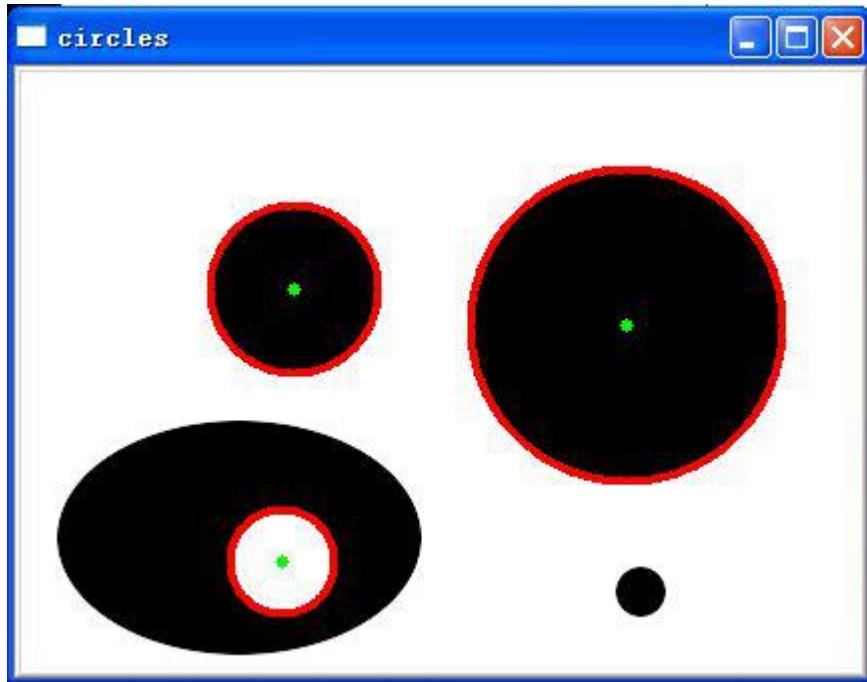
```
CV_HOUGH_PROBABILISTIC, 1, CV_PI/180, 80, 30, 10 );  
    for( i = 0; i < lines->total; i++ )  
    {  
        CvPoint* line = (CvPoint*)cvGetSeqElem(lines, i);  
        cvLine( color_dst, line[0], line[1],  
CV_RGB(255,0,0), 3, 8 );  
    }  
#endif  
    cvNamedWindow( "Source", 1 );  
    cvShowImage( "Source", src );  
  
    cvNamedWindow( "Hough", 1 );  
    cvShowImage( "Hough", color_dst );  
  
    cvWaitKey(0);  
}  
}
```

利用 Hough 变换检测圆（是圆不是椭圆）

原始图:



效果图:



汗，不是到那个小圆是不是圆，怎么没检测出来，我的东东，怎么搞的

源代码：

```
#include <cv.h>
#include <highgui.h>
#include <math.h>

int main(int argc, char** argv)
{
    IplImage* img;
    if( argc == 2 && (img=cvLoadImage(argv[1], 1))!= 0)
    {
        IplImage* gray = cvCreateImage( cvGetSize(img), 8, 1 );
        CvMemStorage* storage = cvCreateMemStorage(0);
        cvCvtColor( img, gray, CV_BGR2GRAY );
        cvSmooth( gray, gray, CV_GAUSSIAN, 9, 9 ); // smooth it,
otherwise a lot of false circles may be detected
        CvSeq* circles = cvHoughCircles( gray, storage,
CV_HOUGH_GRADIENT, 2, gray->height/4, 200, 100 );
        int i;
        for( i = 0; i < circles->total; i++ )
        {
            float* p = (float*)cvGetSeqElem( circles, i );
            cvCircle( img,
cvPoint(cvRound(p[0]),cvRound(p[1])), 3, CV_RGB(0,255,0), -1, 8, 0 );
            cvCircle( img,
```

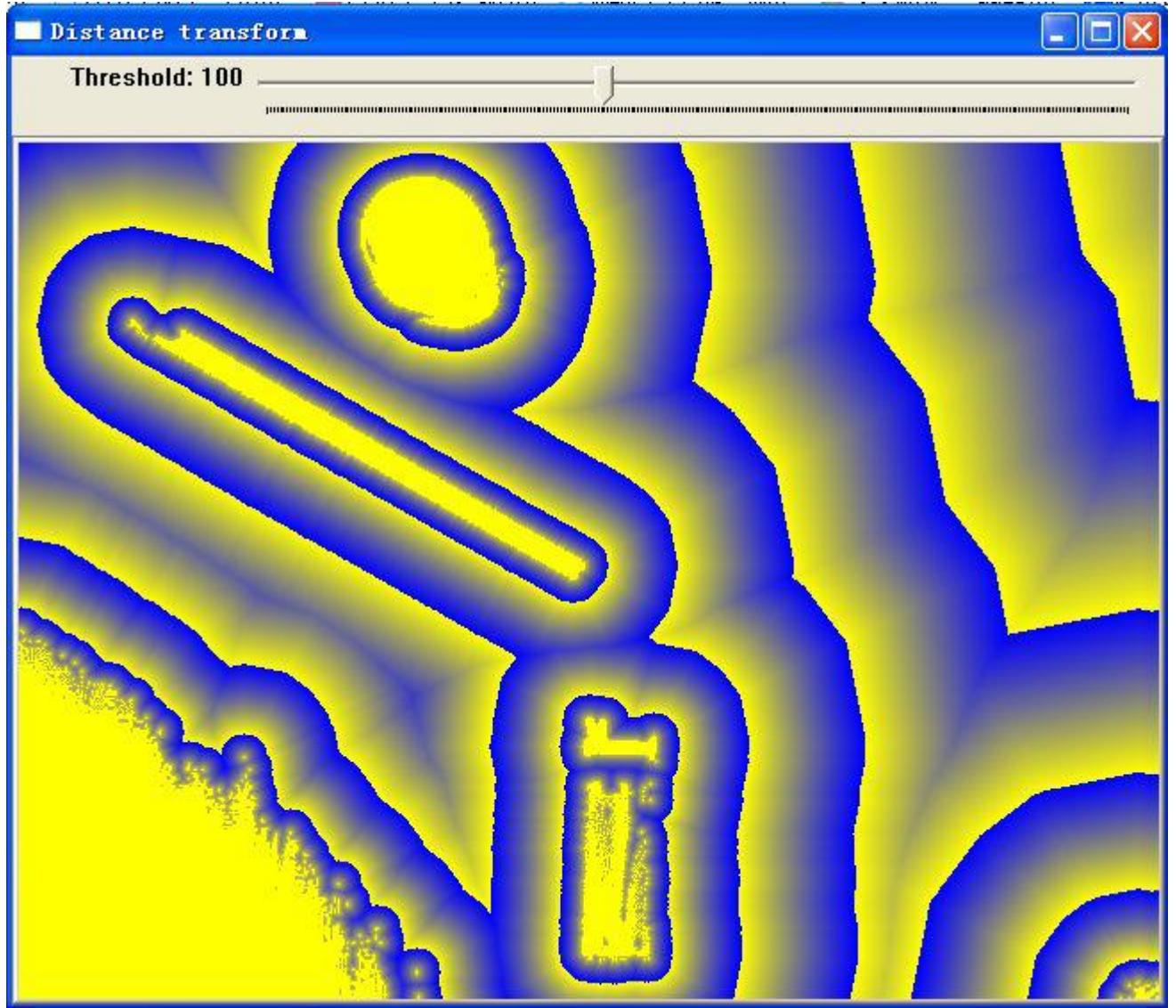
```
cvPoint(cvRound(p[0]), cvRound(p[1])), cvRound(p[2]), CV_RGB(255, 0, 0),  
3, 8, 0 );  
}  
cvNamedWindow( "circles", 1 );  
cvShowImage( "circles", img );  
cvWaitKey(0);  
}  
return 0;  
}
```

距离变换

原图:



处理后的图:



没搞明白这是怎么个距离变换，连图都不一样了这么还叫距离变换，望知道这不剩赐教。

源代码：

```
#include "cv.h"
#include "highgui.h"
#include <stdio.h>

char wndname[] = "Distance transform";
char tbarname[] = "Threshold";
int mask_size = CV_DIST_MASK_5;
int build_voronoi = 0;
int edge_thresh = 100;
```

```
// The output and temporary images
IplImage* dist = 0;
IplImage* dist8u1 = 0;
IplImage* dist8u2 = 0;
IplImage* dist8u = 0;
IplImage* dist32s = 0;

IplImage* gray = 0;
IplImage* edge = 0;
IplImage* labels = 0;

// threshold trackbar callback
void on_trackbar( int dummy )
{
    static const uchar colors[][][3] =
    {
        {0, 0, 0},
        {255, 0, 0},
        {255, 128, 0},
        {255, 255, 0},
        {0, 255, 0},
        {0, 128, 255},
        {0, 255, 255},
        {0, 0, 255},
        {255, 0, 255}
    };

    int msize = mask_size;

    cvThreshold( gray, edge, (float)edge_thresh, (float)edge_thresh,
CV_THRESH_BINARY );

    if( build_voronoi )
        msize = CV_DIST_MASK_5;

    cvDistTransform( edge, dist, CV_DIST_L2, msize, NULL,
build_voronoi ? labels : NULL );

    if( !build_voronoi )
    {
        // begin "painting" the distance transform result
        cvConvertScale( dist, dist, 5000.0, 0 );
        cvPow( dist, dist, 0.5 );

        cvConvertScale( dist, dist32s, 1.0, 0.5 );
    }
}
```

```
        cvAndS( dist32s, cvScalarAll(255), dist32s, 0 );
        cvConvertScale( dist32s, dist8u1, 1, 0 );
        cvConvertScale( dist32s, dist32s, -1, 0 );
        cvAddS( dist32s, cvScalarAll(255), dist32s, 0 );
        cvConvertScale( dist32s, dist8u2, 1, 0 );
        cvMerge( dist8u1, dist8u2, dist8u2, 0, dist8u );
        // end "painting" the distance transform result
    }
else
{
    int i, j;
    for( i = 0; i < labels->height; i++ )
    {
        int* l1 = (int*) (labels->imageData +
i*labels->widthStep);
        float* dd = (float*) (dist->imageData +
i*dist->widthStep);
        uchar* d = (uchar*) (dist8u->imageData +
i*dist8u->widthStep);
        for( j = 0; j < labels->width; j++ )
        {
            int idx = l1[j] == 0 || dd[j] == 0 ? 0 :
(l1[j]-1)%8 + 1;
            int b = cvRound(colors[idx][0]);
            int g = cvRound(colors[idx][1]);
            int r = cvRound(colors[idx][2]);
            d[j*3] = (uchar)b;
            d[j*3+1] = (uchar)g;
            d[j*3+2] = (uchar)r;
        }
    }
}

cvShowImage( wndname, dist8u );
}

int main( int argc, char** argv )
{
    char* filename = argc == 2 ? argv[1] : (char*)"stuff.jpg";

    if( (gray = cvLoadImage( filename, 0 )) == 0 )
        return -1;

    printf( "Hot keys: \n"
            "\tESC - quit the program\n"
```

```
"\t3 - use 3x3 mask\n"
"\t5 - use 5x5 mask\n"
"\t0 - use precise distance transform\n"
"\tv - switch Voronoi diagram mode on/off\n"
"\tENTER - loop through all the modes\n" );

dist = cvCreateImage( cvGetSize(gray), IPL_DEPTH_32F, 1 );
dist8u1 = cvCloneImage( gray );
dist8u2 = cvCloneImage( gray );
dist8u = cvCreateImage( cvGetSize(gray), IPL_DEPTH_8U, 3 );
dist32s = cvCreateImage( cvGetSize(gray), IPL_DEPTH_32S, 1 );
edge = cvCloneImage( gray );
labels = cvCreateImage( cvGetSize(gray), IPL_DEPTH_32S, 1 );

cvNamedWindow( wndname, 1 );

cvCreateTrackbar( tbarname, wndname, &edge_thresh, 255,
on_trackbar ) ;

for(;;)
{
    int c;

    // Call to update the view
    on_trackbar(0);

    c = cvWaitKey(0);

    if( (char)c == 27 )
        break;

    if( (char)c == '3' )
        mask_size = CV_DIST_MASK_3;
    else if( (char)c == '5' )
        mask_size = CV_DIST_MASK_5;
    else if( (char)c == '0' )
        mask_size = CV_DIST_MASK_PRECISE;
    else if( (char)c == 'v' )
        build_voronoi ^= 1;
    else if( (char)c == '\n' )
    {
        if( build_voronoi )
        {
            build_voronoi = 0;
            mask_size = CV_DIST_MASK_3;
```

```
        }

        else if( mask_size == CV_DIST_MASK_3 )
            mask_size = CV_DIST_MASK_5;
        else if( mask_size == CV_DIST_MASK_5 )
            mask_size = CV_DIST_MASK_PRECISE;
        else if( mask_size == CV_DIST_MASK_PRECISE )
            build_voronoi = 1;
    }

}

cvReleaseImage( &gray );
cvReleaseImage( &edge );
cvReleaseImage( &dist );
cvReleaseImage( &dist8u );
cvReleaseImage( &dist8u1 );
cvReleaseImage( &dist8u2 );
cvReleaseImage( &dist32s );
cvReleaseImage( &labels );

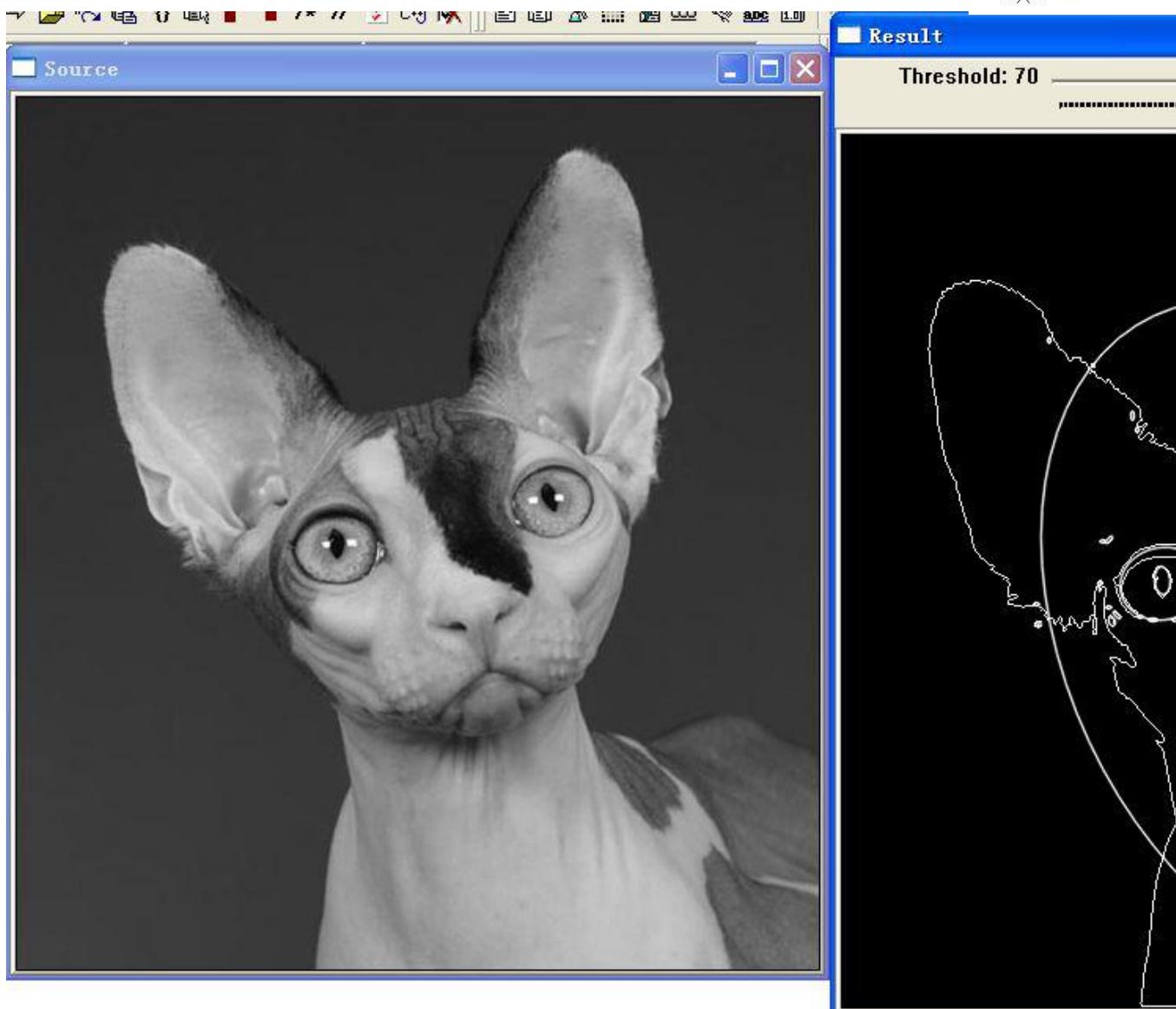
cvDestroyWindow( wndname );

return 0;
}
```

椭圆曲线拟合

程序首先发现图像轮廓，然后用椭圆逼近它

效果图：



还是用 ps 的魔棒工具感觉更好。

源代码：

```
#include "cv.h"
#include "highgui.h"

int slider_pos = 70;

IplImage *image02 = 0, *image03 = 0, *image04 = 0;
void process_image(int h);

int main( int argc, char** argv )
{
    const char* filename = argc == 2 ? argv[1] : (char*)"2.jpg";
```

```
// 读入图像，强制为灰度图像
if( (image03 = cvLoadImage(filename, 0)) == 0 )
    return -1;

// Create the destination images
image02 = cvCloneImage( image03 );
image04 = cvCloneImage( image03 );

// Create windows.
cvNamedWindow("Source", 1);
cvNamedWindow("Result", 1);

// Show the image.
cvShowImage("Source", image03);

// Create toolbars. HighGUI use.
cvCreateTrackbar( "Threshold", "Result", &slider_pos, 255,
process_image );

process_image(0);

// Wait for a key stroke; the same function arranges events
processing
cvWaitKey(0);
cvReleaseImage(&image02);
cvReleaseImage(&image03);

cvDestroyWindow("Source");
cvDestroyWindow("Result");

return 0;
}

// Define trackbar callback functon. This function find contours,
// draw it and approximate it by ellipses.
void process_image(int h)
{
    CvMemStorage* stor;
    CvSeq* cont;
    CvBox2D32f* box;
    CvPoint* PointArray;
    CvPoint2D32f* PointArray2D32f;

    // 创建动态结构序列
    stor = cvCreateMemStorage(0);
```

```
cont = cvCreateSeq(CV_SEQ_ELTYPE_POINT, sizeof(CvSeq),
sizeof(CvPoint) , stor);

// 二值话图像.
cvThreshold( image03, image02, slider_pos, 255,
CV_THRESH_BINARY ) ;

// 寻找所有轮廓.
cvFindContours( image02, stor, &cont, sizeof(CvContour),
CV_RETR_LIST,
CV_CHAIN_APPROX_NONE, cvPoint(0,0)) ;

// Clear images. IPL use.
cvZero(image02);
cvZero(image04);

// 本循环绘制所有轮廓并用椭圆拟合.
for(;cont;cont = cont->h_next)
{
    int i; // Indicator of cycle.
    int count = cont->total; // This is number point in contour
    CvPoint center;
    CvSize size;

    // Number point must be more than or equal to 6 (for
    cvFitEllipse_32f).
    if( count < 6 )
        continue;

    // Alloc memory for contour point set.
    PointArray = (CvPoint*)malloc( count*sizeof(CvPoint) );
    PointArray2D32f=
    (CvPoint2D32f*)malloc( count*sizeof(CvPoint2D32f) );

    // Alloc memory for ellipse data.
    box = (CvBox2D32f*)malloc(sizeof(CvBox2D32f));

    // Get contour point set.
    cvCvtSeqToArray(cont, PointArray, CV_WHOLE_SEQ);

    // Convert CvPoint set to CvBox2D32f set.
    for(i=0; i<count; i++)
    {
        PointArray2D32f[i].x = (float)PointArray[i].x;
```

```
        PointArray2D32f[i].y = (float)PointArray[i].y;
    }

    //拟合当前轮廓.
    cvFitEllipse(PointArray2D32f, count, box);

    // 绘制当前轮廓.
    cvDrawContours(image04, cont, CV_RGB(255, 255, 255),
CV_RGB(255, 255, 255), 0, 1, 8, cvPoint(0, 0));

    // Convert ellipse data from float to integer
representation.
    center.x = cvRound(box->center.x);
    center.y = cvRound(box->center.y);
    size.width = cvRound(box->size.width*0.5);
    size.height = cvRound(box->size.height*0.5);
    box->angle = -box->angle;

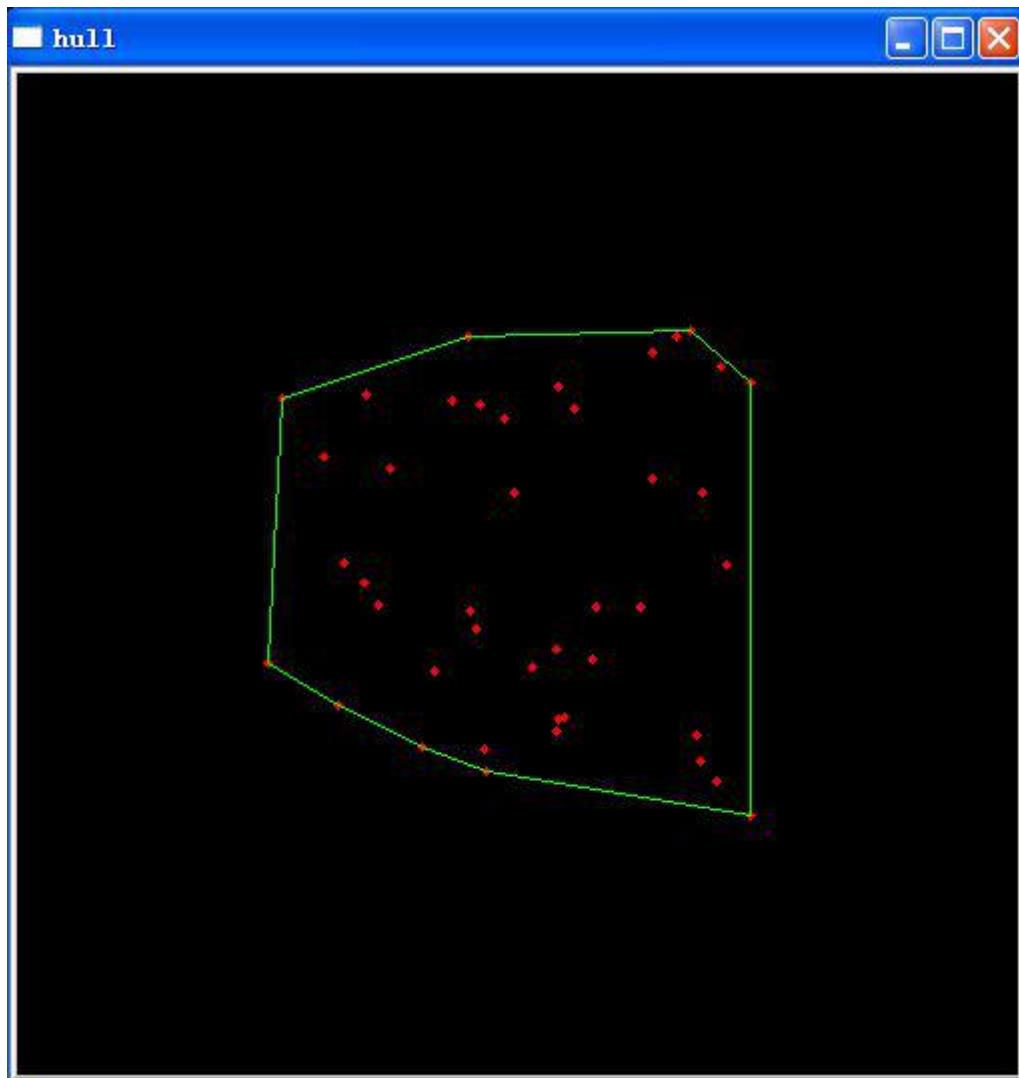
    // Draw ellipse.
    cvEllipse(image04, center, size,
              box->angle, 0, 360,
              CV_RGB(0, 0, 255), 1, CV_AA, 0);

    // Free memory.
    free(PointArray);
    free(PointArray2D32f);
    free(box);
}

// Show image. HighGUI use.
cvShowImage("Result", image04);
}
```

由点集序列或数组创建凸外形

不过说 看效果图就能明白原理:



源代码：

```
#include "cv.h"
#include "highgui.h"
#include <stdlib.h>

#define ARRAY 0 /* switch between array/sequence method by replacing 0<=>1
 */

int main( int argc, char** argv )
{
    IplImage* img = cvCreateImage( cvSize( 500, 500 ), 8, 3 );
    cvNamedWindow( "hull", 1 );

#if !ARRAY
    CvMemStorage* storage = cvCreateMemStorage();
#endif
```

```
for(;;)
{
    int i, count = rand()%100 + 1, hullcount;
    CvPoint pt0;

#if !ARRAY
    CvSeq* ptseq = cvCreateSeq( CV_SEQ_KIND_GENERIC|CV_32SC2,
        sizeof(CvContour),
        sizeof(CvPoint), storage );
    CvSeq* hull;

    for( i = 0; i < count; i++ )
    {
        pt0.x = rand() % (img->width/2) + img->width/4;
        pt0.y = rand() % (img->height/2) + img->height/4;
        cvSeqPush( ptseq, &pt0 );
    }
    hull = cvConvexHull2( ptseq, 0, CV_CLOCKWISE, 0 );
    hullcount = hull->total;
#else
    CvPoint* points = (CvPoint*)malloc( count *
        sizeof(points[0]));
    int* hull = (int*)malloc( count * sizeof(hull[0]));
    CvMat point_mat = cvMat( 1, count, CV_32SC2, points );
    CvMat hull_mat = cvMat( 1, count, CV_32SC1, hull );

    for( i = 0; i < count; i++ )
    {
        pt0.x = rand() % (img->width/2) + img->width/4;
        pt0.y = rand() % (img->height/2) + img->height/4;
        points[i] = pt0;
    }
    cvConvexHull2( &point_mat, &hull_mat, CV_CLOCKWISE, 0 );
    hullcount = hull_mat.cols;
#endif
    cvZero( img );
    for( i = 0; i < count; i++ )
    {
#if !ARRAY
        pt0 = *CV_GET_SEQ_ELEM( CvPoint, ptseq, i );
#else
        pt0 = points[i];
#endif
        cvCircle( img, pt0, 2, CV_RGB( 255, 0, 0 ),
    }
```

```
CV_FILLED );
    }

#if !ARRAY
    pt0 = **CV_GET_SEQ_ELEM( CvPoint*, hull, hullcount - 1 );
#else
    pt0 = points[hull[hullcount-1]];
#endif

    for( i = 0; i < hullcount; i++ )
    {
#if !ARRAY
        CvPoint pt = **CV_GET_SEQ_ELEM( CvPoint*, hull,
i );
#else
        CvPoint pt = points[hull[i]];
#endif
        cvLine( img, pt0, pt, CV_RGB( 0, 255, 0 ) );
        pt0 = pt;
    }

    cvShowImage( "hull", img );

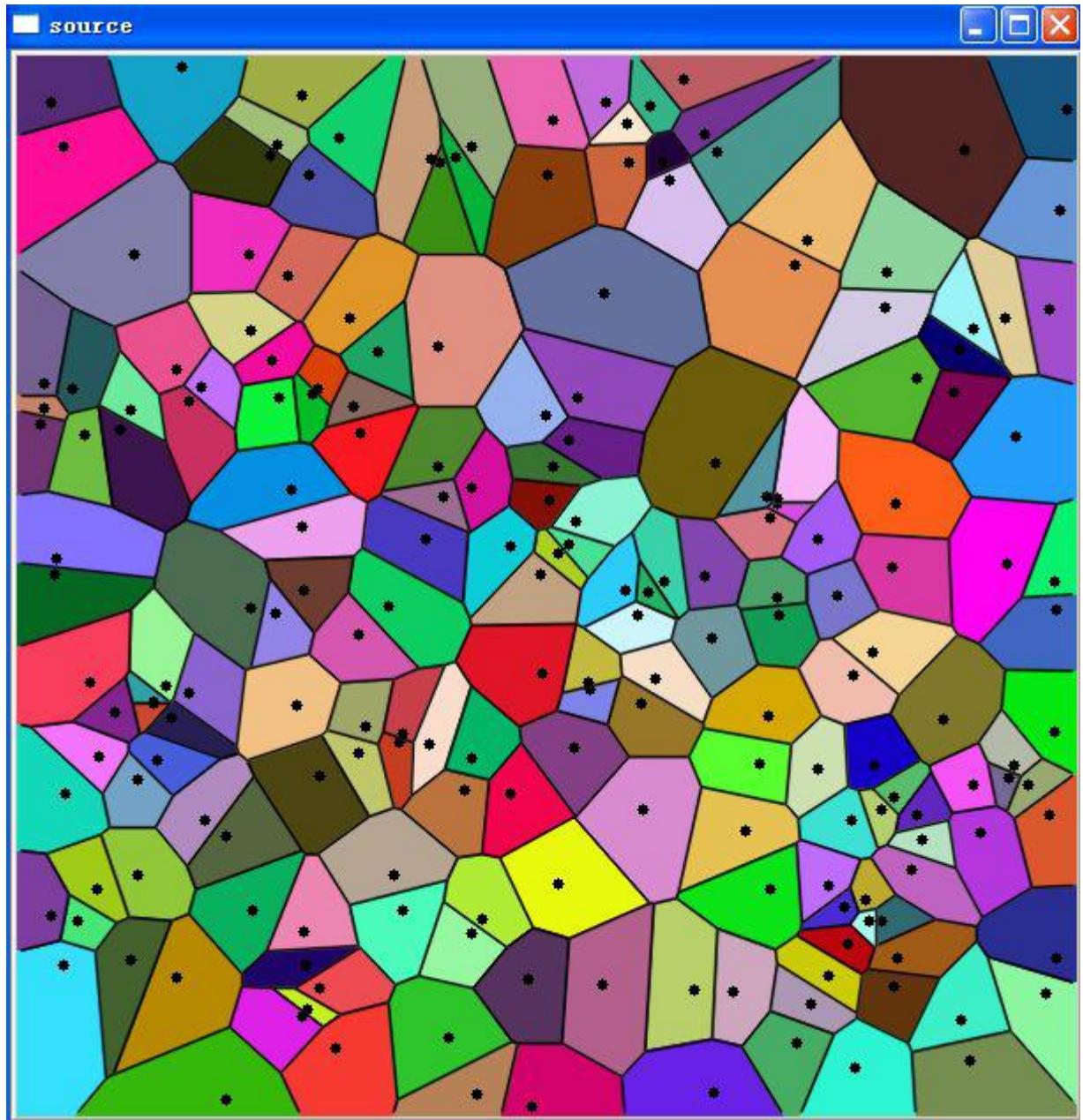
    int key = cvWaitKey(0);
    if( key == 27 ) // 'ESC'
        break;

#endif !ARRAY
    cvClearMemStorage( storage );
#else
    free( points );
    free( hull );
#endif
}
}
```

Delaunay 三角形和 Voronoi 划分的迭代式构造

汗，这题目，我都晕了，什么东东呢。

效果图：（实际是个动画一样的东东，最终效果我截下了）



很欣赏这种 main 函数的写法，就两句，精辟，嘿嘿

源代码：

```
#include "cv.h"
#include "highgui.h"
#include <stdio.h>

/* the script demonstrates iterative construction of
   delaunay triangulation and voronoi tessellation */

CvSubdiv2D* init_delaunay( CvMemStorage* storage,
                           CvRect rect )
```

```
{  
    CvSubdiv2D* subdiv;  
  
    subdiv = cvCreateSubdiv2D( CV_SEQ_KIND_SUBDIV2D, sizeof(*subdiv),  
                               sizeof(C  
vSubdiv2DPoint),  
                               sizeof(C  
vQuadEdge2D),  
                               storage )  
;  
    cvInitSubdivDelaunay2D( subdiv, rect );  
  
    return subdiv;  
}  
  
void draw_subdiv_point( IplImage* img, CvPoint2D32f fp, CvScalar color )  
{  
    cvCircle( img, cvPoint(cvRound(fp.x), cvRound(fp.y)), 3, color,  
CV_FILLED, 8, 0 );  
}  
  
void draw_subdiv_edge( IplImage* img, CvSubdiv2DEdge edge, CvScalar  
color )  
{  
    CvSubdiv2DPoint* org_pt;  
    CvSubdiv2DPoint* dst_pt;  
    CvPoint2D32f org;  
    CvPoint2D32f dst;  
    CvPoint iorg, idst;  
  
    org_pt = cvSubdiv2DEdge0rg(edge);  
    dst_pt = cvSubdiv2DEdgeDst(edge);  
  
    if( org_pt && dst_pt )  
    {  
        org = org_pt->pt;  
        dst = dst_pt->pt;  
  
        iorg = cvPoint( cvRound( org.x ), cvRound( org.y ) );  
        idst = cvPoint( cvRound( dst.x ), cvRound( dst.y ) );  
    }  
}
```

```
        cvLine( img, iorg, idst, color, 1, CV_AA, 0 );
    }
}

void draw_subdiv( IplImage* img, CvSubdiv2D* subdiv,
                  CvScalar delaunay_color, CvScalar
voronoi_color )
{
    CvSeqReader reader;
    int i, total = subdiv->edges->total;
    int elem_size = subdiv->edges->elem_size;

    cvStartReadSeq( (CvSeq*) (subdiv->edges), &reader, 0 );

    for( i = 0; i < total; i++ )
    {
        CvQuadEdge2D* edge = (CvQuadEdge2D*) (reader.ptr);

        if( CV_IS_SET_ELEM( edge ) )
        {
            draw_subdiv_edge( img, (CvSubdiv2DEdge) edge + 1,
voronoi_color );
            draw_subdiv_edge( img, (CvSubdiv2DEdge) edge,
delaunay_color );
        }

        CV_NEXT_SEQ_ELEM( elem_size, reader );
    }
}

void locate_point( CvSubdiv2D* subdiv, CvPoint2D32f fp, IplImage* img,
                  CvScalar active_color )
{
    CvSubdiv2DEdge e;
    CvSubdiv2DEdge e0 = 0;
    CvSubdiv2DPoint* p = 0;

    cvSubdiv2DLocate( subdiv, fp, &e0, &p );

    if( e0 )
    {
        e = e0;
        do
```

```
{  
    draw_subdiv_edge( img, e, active_color );  
    e = cvSubdiv2DGetEdge( e, CV_NEXT_AROUND_LEFT );  
}  
while( e != e0 );  
}  
  
draw_subdiv_point( img, fp, active_color );  
}  
  
void draw_subdiv_facet( IplImage* img, CvSubdiv2DEdge edge )  
{  
    CvSubdiv2DEdge t = edge;  
    int i, count = 0;  
    CvPoint* buf = 0;  
  
    // count number of edges in facet  
    do  
    {  
        count++;  
        t = cvSubdiv2DGetEdge( t, CV_NEXT_AROUND_LEFT );  
    } while (t != edge );  
  
    buf = (CvPoint*)malloc( count * sizeof(buf[0]) );  
  
    // gather points  
    t = edge;  
    for( i = 0; i < count; i++ )  
    {  
        CvSubdiv2DPoint* pt = cvSubdiv2DEdgeOrg( t );  
        if( !pt ) break;  
        buf[i] = cvPoint( cvRound(pt->pt.x), cvRound(pt->pt.y) );  
        t = cvSubdiv2DGetEdge( t, CV_NEXT_AROUND_LEFT );  
    }  
  
    if( i == count )  
    {  
        CvSubdiv2DPoint* pt =  
        cvSubdiv2DEdgeDst( cvSubdiv2DRotateEdge( edge, 1 ));  
        cvFillConvexPoly( img, buf, count,  
        CV_RGB(rand()&255, rand()&255, rand()&255), CV_AA, 0 );  
        cvPolyLine( img, &buf, &count, 1, 1, CV_RGB(0,0,0), 1,  
        CV_AA, 0 );  
        draw_subdiv_point( img, pt->pt, CV_RGB(0,0,0));  
    }  
}
```

```
        }

        free( buf );
    }

void paint_voronoi( CvSubdiv2D* subdiv, IplImage* img )
{
    CvSeqReader reader;
    int i, total = subdiv->edges->total;
    int elem_size = subdiv->edges->elem_size;

    cvCalcSubdivVoronoi2D( subdiv );

    cvStartReadSeq( (CvSeq*)(subdiv->edges), &reader, 0 );

    for( i = 0; i < total; i++ )
    {
        CvQuadEdge2D* edge = (CvQuadEdge2D*)(reader.ptr);

        if( CV_IS_SET_ELEM( edge ) )
        {
            CvSubdiv2DEdge e = (CvSubdiv2DEdge)edge;
            // left
            draw_subdiv_facet( img, cvSubdiv2DRotateEdge( e,
1 ) );

            // right
            draw_subdiv_facet( img, cvSubdiv2DRotateEdge( e,
3 ) );
        }

        CV_NEXT_SEQ_ELEM( elem_size, reader );
    }
}

void run(void)
{
    char win[] = "source";
    int i;
    CvRect rect = { 0, 0, 600, 600 };
    CvMemStorage* storage;
    CvSubdiv2D* subdiv;
    IplImage* img;
    CvScalar active_facet_color, delaunay_color, voronoi_color,
bkgnd_color;
```

```
active_facet_color = CV_RGB( 255, 0, 0 );
delaunay_color = CV_RGB( 0, 0, 0 );
voronoi_color = CV_RGB( 0, 180, 0 );
bknd_color = CV_RGB( 255, 255, 255 );

img = cvCreateImage( cvSize(rect.width, rect.height), 8, 3 );
cvSet( img, bknd_color, 0 );

cvNamedWindow( win, 1 );

storage = cvCreateMemStorage(0);
subdiv = init_delaunay( storage, rect );

printf("Delaunay triangulation will be build now
interactively.\n"
      "To stop the process, press any key\n\n");

for( i = 0; i < 200; i++ )
{
    CvPoint2D32f fp =
cvPoint2D32f( (float)(rand()%(rect.width-10)+5),
               (float)(rand()%(rect.height-10)+5));

    locate_point( subdiv, fp, img, active_facet_color );
    cvShowImage( win, img );

    if( cvWaitKey( 100 ) >= 0 )
        break;

    cvSubdivDelaunay2DInsert( subdiv, fp );
    cvCalcSubdivVoronoi2D( subdiv );
    cvSet( img, bknd_color, 0 );
    draw_subdiv( img, subdiv, delaunay_color,
                 voronoi_color );
    cvShowImage( win, img );

    if( cvWaitKey( 100 ) >= 0 )
        break;
}

cvSet( img, bknd_color, 0 );
paint_voronoi( subdiv, img );
cvShowImage( win, img );
```

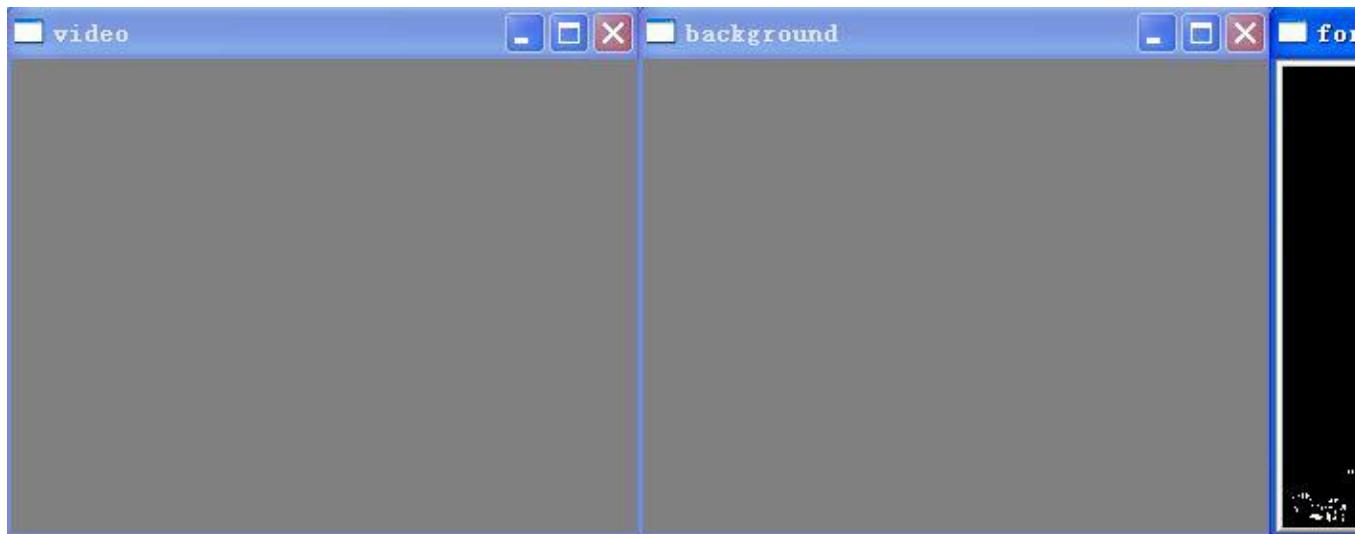
```
cvWaitKey(0);

cvReleaseMemStorage( &storage );
cvReleaseImage(&img);
cvDestroyWindow( win );
}

int main( int argc, char** argv )
{
    run();
    return 0;
}
```

利用背景建模检测运动物体（推荐）

效果图（截的时候前两个图没截出来），视频是一个交通路面的视频（自己可以去下个）：



大家就看个大概吧：

源代码：

```
#include <stdio.h>
#include <cv.h>
#include <highgui.h>

int main( int argc, char** argv )
{
//声明 IplImage 指针
IplImage* pFrame = NULL;
```

```
IplImage* pFrImg = NULL;
IplImage* pBkImg = NULL;

CvMat* pFrameMat = NULL;
CvMat* pFrMat = NULL;
CvMat* pBkMat = NULL;

CvCapture* pCapture = NULL;

int nFrmNum = 0;

//创建窗口
cvNamedWindow("video", 1);
cvNamedWindow("background", 1);
cvNamedWindow("foreground", 1);
//使窗口有序排列
cvMoveWindow("video", 30, 0);
cvMoveWindow("background", 360, 0);
cvMoveWindow("foreground", 690, 0);

if( argc != 2 )
{
    fprintf(stderr, "Usage: bkgrd <video_file_name>\n");
    return -1;
}

//打开视频文件
if( !(pCapture = cvCaptureFromFile(argv[1])))
{
    fprintf(stderr, "Can not open video file %s\n", argv[1]);
    return -2;
}

//逐帧读取视频
while(pFrame = cvQueryFrame( pCapture ))
{
    nFrmNum++;

    //如果是第一帧，需要申请内存，并初始化
    if(nFrmNum == 1)
    {
        pBkImg = cvCreateImage(cvSize(pFrame->width, pFrame->height),
IPL_DEPTH_8U, 1);
        pFrImg = cvCreateImage(cvSize(pFrame->width, pFrame->height),
IPL_DEPTH_8U, 1);
    }
}
```

```
pBkMat = cvCreateMat(pFrame->height, pFrame->width, CV_32FC1);
pFrMat = cvCreateMat(pFrame->height, pFrame->width, CV_32FC1);
pFrameMat = cvCreateMat(pFrame->height, pFrame->width,
CV_32FC1);

//转化成单通道图像再处理
cvCvtColor(pFrame, pBkImg, CV_BGR2GRAY);
cvCvtColor(pFrame, pFrImg, CV_BGR2GRAY);

cvConvert(pFrImg, pFrameMat);
cvConvert(pFrImg, pFrMat);
cvConvert(pFrImg, pBkMat);
}

else
{
    cvCvtColor(pFrame, pFrImg, CV_BGR2GRAY);
    cvConvert(pFrImg, pFrameMat);
    //先做高斯滤波，以平滑图像
    //cvSmooth(pFrameMat, pFrameMat, CV_GAUSSIAN, 3, 0, 0);

    //当前帧跟背景图相减
    cvAbsDiff(pFrameMat, pBkMat, pFrMat);

    //二值化前景图
    cvThreshold(pFrMat, pFrImg, 60, 255.0, CV_THRESH_BINARY);

    //进行形态学滤波，去掉噪音
    //cvErode(pFrImg, pFrImg, 0, 1);
    //cvDilate(pFrImg, pFrImg, 0, 1);

    //更新背景
    cvRunningAvg(pFrameMat, pBkMat, 0.003, 0);
    //将背景转化为图像格式，用以显示
    cvConvert(pBkMat, pBkImg);

    //显示图像
    cvShowImage("video", pFrame);
    cvShowImage("background", pBkImg);
    cvShowImage("foreground", pFrImg);

    //如果有按键事件，则跳出循环
    //此等待也为 cvShowImage 函数提供时间完成显示
    //等待时间可以根据 CPU 速度调整
    if( cvWaitKey(2) >= 0 )
        break;
```

```
    } // end of if-else
    } // end of while-loop

    //销毁窗口
    cvDestroyWindow("video");
    cvDestroyWindow("background");
    cvDestroyWindow("foreground");

    //释放图像和矩阵
    cvReleaseImage(&pFrImg);
    cvReleaseImage(&pBkImg);

    cvReleaseMat(&pFrameMat);
    cvReleaseMat(&pFrMat);
    cvReleaseMat(&pBkMat);

    return 0;
}
```

运动模板检测（摄像头）

效果图：（太黑了）



源代码：

```
#include "cv.h"
#include "highgui.h"
#include <time.h>
#include <math.h>
```

```
#include <ctype.h>
#include <stdio.h>

// various tracking parameters (in seconds)
const double MHI_DURATION = 0.5;
const double MAX_TIME_DELTA = 0.5;
const double MIN_TIME_DELTA = 0.05;
// 用于运动检测的循环帧数，与机器速度以及 FPS 设置有关
const int N = 2;

// ring image buffer
IplImage **buf = 0;
int last = 0;

// temporary images
IplImage *mhi = 0; // MHI: motion history image
IplImage *orient = 0; // orientation
IplImage *mask = 0; // valid orientation mask
IplImage *segmask = 0; // motion segmentation map
CvMemStorage* storage = 0; // temporary storage

// parameters:
// img - input video frame
// dst - resultant motion picture
// args - optional parameters
void update_mhi( IplImage* img, IplImage* dst, int diff_threshold )
{
    double timestamp = clock()/1000.; // get current time in seconds
    CvSize size = cvSize(img->width, img->height); // get current
frame size
    int i, idx1 = last, idx2;
    IplImage* silh;
    CvSeq* seq;
    CvRect comp_rect;
    double count;
    double angle;
    CvPoint center;
    double magnitude;
    CvScalar color;

    // allocate images at the beginning or
    // reallocate them if the frame size is changed
    if( !mhi || mhi->width != size.width || mhi->height !=
size.height )
    {
```

```
if( buf == 0 )
{
    buf = (IplImage**)malloc(N*sizeof(buf[0]));
    memset( buf, 0, N*sizeof(buf[0]));
}

for( i = 0; i < N; i++ )
{
    cvReleaseImage( &buf[i] );
    buf[i] = cvCreateImage( size, IPL_DEPTH_8U, 1 );
    cvZero( buf[i] );
}

cvReleaseImage( &mhi );
cvReleaseImage( &orient );
cvReleaseImage( &segmask );
cvReleaseImage( &mask );

mhi = cvCreateImage( size, IPL_DEPTH_32F, 1 );
cvZero( mhi ); // clear MHI at the beginning
orient = cvCreateImage( size, IPL_DEPTH_32F, 1 );
segmask = cvCreateImage( size, IPL_DEPTH_32F, 1 );
mask = cvCreateImage( size, IPL_DEPTH_8U, 1 );
}

cvCvtColor( img, buf[last], CV_BGR2GRAY ); // convert frame to
grayscale

idx2 = (last + 1) % N; // index of (last - (N-1))th frame
last = idx2;

silh = buf[idx2];
// 相邻两帧的差
cvAbsDiff( buf[idx1], buf[idx2], silh ); // get difference between
frames

// 对差图像做二值化
cvThreshold( silh, silh, diff_threshold, 1, CV_THRESH_BINARY );
// and threshold it
cvUpdateMotionHistory( silh, mhi, timestamp, MHI_DURATION ); // update MHI

// convert MHI to blue 8u image
// cvCvtScale 的第四个参数 shift = (MHI_DURATION -
timestamp)*255./MHI_DURATION
// 控制帧差的消失速率
```

```

cvCvtScale( mhi, mask, 255./MHI_DURATION,
(MHI_DURATION - timestamp)*255./MHI_DURATION );

cvZero( dst );
cvCvtPlaneToPix(mask, 0, 0, 0, dst); // B, G, R, 0 → dist : convert
to BLUE image

// 计算运动的梯度方向以及正确的方向掩模 mask
// Filter size = 3
cvCalcMotionGradient( mhi, mask, orient,
MAX_TIME_DELTA, MIN_TIME_DELTA, 3 );

if( !storage )
    storage = cvCreateMemStorage(0);
else
    cvClearMemStorage(storage);

// 运动分割: 获得运动部件的连续序列
// segmask is marked motion components map. It is not used further
seq = cvSegmentMotion( mhi, segmask, storage, timestamp,
MAX_TIME_DELTA );

// iterate through the motion components,
// One more iteration (i == -1) corresponds to the whole image
(global motion)
for( i = 0; i < seq->total; i++ )
{
    if( i < 0 ) { // case of the whole image, 对整幅图像做
操作
        comp_rect = cvRect( 0, 0, size.width,
size.height );
        color = CV_RGB(255, 255, 255); // white color
        magnitude = 100; // 画线长度以及圆半径的大小控
制
    }
    else { // i-th motion component
        comp_rect =
((CvConnectedComp*)cvGetSeqElem( seq, i ))->rect;
// 去掉小的部分
if( comp_rect.width + comp_rect.height < 100 )
        continue;
        color = CV_RGB(255, 0, 0); // red color
        magnitude = 30;
//if(seq->total > 0) MessageBox(NULL, "Motion

```

```
Detected", NULL, 0) ;
    }

    // select component ROI
    cvSetImageROI( silh, comp_rect );
    cvSetImageROI( mhi, comp_rect );
    cvSetImageROI( orient, comp_rect );
    cvSetImageROI( mask, comp_rect );

    // 在选择的区域内，计算运动方向
    angle = cvCalcGlobalOrientation( orient, mask, mhi,
timestamp,
MHI_DURATION);
    angle = 360.0 - angle; // adjust for images with top-left
origin

    // 在轮廓内计算点数
    // Norm(L1) = sum of total pixel values
    count = cvNorm( silh, 0, CV_L1, 0 );

    // The function cvResetImageROI releases image ROI
    cvResetImageROI( mhi );
    cvResetImageROI( orient );
    cvResetImageROI( mask );
    cvResetImageROI( silh );

    // check for the case of little motion
    if( count < comp_rect.width*comp_rect.height * 0.05 ) // five percent of pixel
        continue;

    // draw a clock with arrow indicating the direction
    center = cvPoint( (comp_rect.x + comp_rect.width/2),
                      (comp_rect.y +
comp_rect.height/2) );

    cvCircle( dst, center, cvRound(magnitude*1.2), color, 3,
CV_AA, 0 );
    cvLine( dst, center, cvPoint( cvRound( center.x +
magnitude*cos(angle*CV_PI/180)),
                                cvRound( center.y -
magnitude*sin(angle*CV_PI/180))), color, 3, CV_AA, 0 );
}
```

```
int main(int argc, char** argv)
{
    IplImage* motion = 0;
    CvCapture* capture = 0;

    if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 &&
isdigit(argv[1][0])))
        capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' :
0 );
    else if( argc == 2 )
        capture = cvCaptureFromAVI( argv[1] );

    if( capture )
    {
        cvNamedWindow( "Motion", 1 );
        for(;;)
        {
            IplImage* image;
            if( !cvGrabFrame( capture ) )
                break;
            image = cvRetrieveFrame( capture );

            if( image )
            {
                if( !motion )
                {
                    cvZero( motion );
                    motion->origin = image->origin;
                }
            }

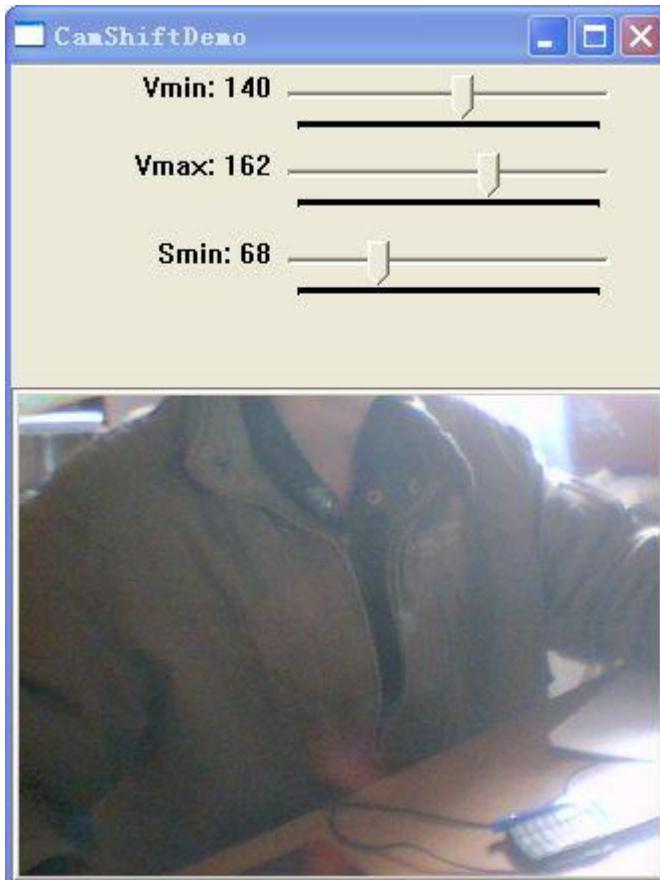
            update_mhi( image, motion, 60 );
            cvShowImage( "Motion", motion );

            if( cvWaitKey(10) >= 0 )
                break;
        }
        cvReleaseCapture( &capture );
        cvDestroyWindow( "Motion" );
    }

    return 0;
}
```

显示如何利用 Camshift 算法进行彩色目标的跟踪

没看出来有跟踪效果，是不是哥摄像头太拉了或得加强一下理论知识的学习：



穿的有点寒碜，嘿嘿

源代码：

```
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <ctype.h>

IplImage *image = 0, *hsv = 0, *hue = 0, *mask = 0, *backproject = 0,
*histimg = 0;
CvHistogram *hist = 0;

int backproject_mode = 0;
int select_object = 0;
int track_object = 0;
int show_hist = 1;
```

```
CvPoint origin;
CvRect selection;
CvRect track_window;
CvBox2D track_box; // tracking 返回的区域 box, 带角度
CvConnectedComp track_comp;
int hdims = 48;           // 划分 HIST 的个数, 越高越精确
float hranges_arr[] = {0, 180};
float* hranges = hranges_arr;
int vmin = 10, vmax = 256, smin = 30;

void on_mouse( int event, int x, int y, int flags )
{
    if( !image )
        return;

    if( image->origin )
        y = image->height - y;

    if( select_object )
    {
        selection.x = MIN(x, origin.x);
        selection.y = MIN(y, origin.y);
        selection.width = selection.x + CV_IABS(x - origin.x);
        selection.height = selection.y + CV_IABS(y - origin.y);

        selection.x = MAX( selection.x, 0 );
        selection.y = MAX( selection.y, 0 );
        selection.width = MIN( selection.width, image->width );
        selection.height = MIN( selection.height,
                               image->height );
        selection.width -= selection.x;
        selection.height -= selection.y;
    }

    switch( event )
    {
        case CV_EVENT_LBUTTONDOWN:
            origin = cvPoint(x, y);
            selection = cvRect(x, y, 0, 0);
            select_object = 1;
            break;
        case CV_EVENT_LBUTTONUP:
            select_object = 0;
            if( selection.width > 0 && selection.height > 0 )
```

```
        track_object = -1;

#ifndef _DEBUG
    printf("\n # 鼠标的选择区域: ");
    printf("\n      X = %d, Y = %d, Width = %d, Height = %d",
           selection.x, selection.y, selection.width,
           selection.height);
#endif
    break;
}
}

CvScalar hsv2rgb( float hue )
{
    int rgb[3], p, sector;
    static const int sector_data[][3]=
        {{0, 2, 1}, {1, 2, 0}, {1, 0, 2}, {2, 0, 1}, {2, 1, 0}, {0, 1, 2}};
    hue *= 0.033333333333333333333333333333f;
    sector = cvFloor(hue);
    p = cvRound(255*(hue - sector));
    p ^= sector & 1 ? 255 : 0;

    rgb[sector_data[sector][0]] = 255;
    rgb[sector_data[sector][1]] = 0;
    rgb[sector_data[sector][2]] = p;

#endif _DEBUG
    printf("\n # Convert HSV to RGB: ");
    printf("\n      HUE = %f", hue);
    printf("\n      R = %d, G = %d, B = %d", rgb[0], rgb[1], rgb[2]);
#endif
    return cvScalar(rgb[2], rgb[1], rgb[0], 0);
}

int main( int argc, char** argv )
{
    CvCapture* capture = 0;
    IplImage* frame = 0;

    if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 &&
isdigit(argv[1][0])))
        capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' :
0 );
```

```
else if( argc == 2 )
    capture = cvCaptureFromAVI( argv[1] );

if( !capture )
{
    fprintf(stderr, "Could not initialize capturing... \n");
    return -1;
}

printf( "Hot keys: \n"
        "\tESC - quit the program\n"
        "\\tc - stop the tracking\n"
        "\t\tb - switch to/from backprojection view\n"
        "\t\tb - show/hide object histogram\n"
        "To initialize tracking, select the object with
mouse\n" );

//cvNamedWindow( "Histogram", 1 );
cvNamedWindow( "CamShiftDemo", 1 );
cvSetMouseCallback( "CamShiftDemo", on_mouse, NULL ); // on_mouse
自定义事件
cvCreateTrackbar( "Vmin", "CamShiftDemo", &vmin, 256, 0 );
cvCreateTrackbar( "Vmax", "CamShiftDemo", &vmax, 256, 0 );
cvCreateTrackbar( "Smin", "CamShiftDemo", &smin, 256, 0 );

for(;;)
{
    int i, bin_w, c;

    frame = cvQueryFrame( capture );
    if( !frame )
        break;

    if( !image )
    {
        /* allocate all the buffers */
        image = cvCreateImage( cvGetSize(frame), 8, 3 );
        image->origin = frame->origin;
        hsv = cvCreateImage( cvGetSize(frame), 8, 3 );
        hue = cvCreateImage( cvGetSize(frame), 8, 1 );
        mask = cvCreateImage( cvGetSize(frame), 8, 1 );
        backproject = cvCreateImage( cvGetSize(frame), 8,
1 );
        hist = cvCreateHist( 1, &hdims, CV_HIST_ARRAY,
&hranges, 1 ); // 计算直方图
    }
}
```

```

histimg = cvCreateImage( cvSize(320, 200), 8, 3 );
cvZero( histimg );
}

cvCopy( frame, image, 0 );
cvCvtColor( image, hsv, CV_BGR2HSV ); // 彩色空间转换
BGR to HSV

if( track_object )
{
    int _vmin = vmin, _vmax = vmax;

    cvInRangeS( hsv,
cvScalar(0, smin, MIN(_vmin, _vmax), 0),
cvScalar(180, 256, MAX(_vmin, _vmax), 0), mask ); // 得到二值的 MASK
    cvSplit( hsv, hue, 0, 0, 0 ); // 只提取 HUE 分量

    if( track_object < 0 )
    {
        float max_val = 0.f;
        cvSetImageROI( hue, selection ); // 得到选择区域 for ROI
        cvSetImageROI( mask, selection ); // 得到选择区域 for mask
        cvCalcHist( &hue, hist, 0, mask ); // 计算直方图
        cvGetMinMaxHistValue( hist, 0, &max_val,
0, 0 ); // 只找最大值
        cvConvertScale( hist->bins, hist->bins,
max_val ? 255. / max_val : 0., 0 ); // 缩放 bin 到区间 [0, 255]
        cvResetImageROI( hue ); // remove ROI
        cvResetImageROI( mask );
        track_window = selection;
        track_object = 1;

        cvZero( histimg );
        bin_w = histimg->width / hdims; // hdims: 条的个数, 则 bin_w 为条的宽度
    }

    // 画直方图
    for( i = 0; i < hdims; i++ )
    {
        int val =

```

```
cvRound( cvGetReal1D(hist->bins, i)*histimg->height/255 );
          CvScalar color =
hsv2rgb(i*180. f/hdims);
          cvRectangle( histimg,
cvPoint(i*bin_w,histimg->height),
          cvPo
int((i+1)*bin_w,histimg->height - val),
          colo
r, -1, 8, 0 );
          }
}

cvCalcBackProject( &hue, backproject, hist ); //  
使用 back project 方法
cvAnd( backproject, mask, backproject, 0 );

// calling CAMSHIFT 算法模块
cvCamShift( backproject, track_window,
cvTermCriteria( CV_TER
MCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ),
&track_comp,
&track_box );
track_window = track_comp.rect;

if( backproject_mode )
cvCvtColor( backproject, image,
CV_GRAY2BGR ); // 使用 backproject 灰度图像
if( image->origin )
track_box.angle = -track_box.angle;
cvEllipseBox( image, track_box, CV_RGB(255, 0, 0),
3, CV_AA, 0 );
}

if( select_object && selection.width > 0 &&
selection.height > 0 )
{
cvSetImageROI( image, selection );
cvXorS( image, cvScalarAll(255), image, 0 );
cvResetImageROI( image );
}

cvShowImage( "CamShiftDemo", image );
cvShowImage( "Histogram", histimg );
```

```
c = cvWaitKey(10);
if( c == 27 )
    break; // exit from for-loop
switch( c )
{
case 'b':
    backproject_mode ^= 1;
    break;
case 'c':
    track_object = 0;
    cvZero( histimg );
    break;
case 'h':
    show_hist ^= 1;
    if( !show_hist )
        cvDestroyWindow( "Histogram" );
    else
        cvNamedWindow( "Histogram", 1 );
    break;
default:
    ;
}
}

cvReleaseCapture( &capture );
cvDestroyWindow("CamShiftDemo");

return 0;
}
```