转 Camshift算法原理及其Opencv实现

2013年10月02日 15:31:06 阅读数：33663

Camshift原理

camshift利用目标的颜色直方图模型将图像转换为颜色概率分布图，初始化一个搜索窗的大小和位置，并根据上一帧得到的结果自适应调整搜索窗口的位置和大小，从而定位出当前图像中目标的中心位置。

分为三个部分：

1--色彩投影图（反向投影）：

(1).RGB颜色空间对光照亮度变化较为敏感，为了减少此变化对跟踪效果的影响，首先将图像从RGB空间转换到HSV空间。(2).然后对其中的H分量作直方图，在直方图中代表了不同H分量值出现的概率或者像素个数，就是说可以查找出H分量大小为h的概率或者像素个数，即得到了颜色概率查找表。(3).将图像中每个像素的值用其颜色出现的概率对替换，就得到了颜色概率分布图。这个过程就叫反向投影，颜色概率分布图是一个灰度图像。

2--meanshift

meanshift算法是一种密度函数梯度估计的非参数方法，通过迭代寻优找到概率分布的极值来定位目标。

算法过程为：

(1).在颜色概率分布图中选取搜索窗W

(2).计算零阶距：

$$M_{00} = \sum_x \sum_y I(x,y)$$

计算一阶距：

$$M_{10} = \sum_x \sum_y xI(x,y); M_{01} = \sum_x \sum_y yI(x,y)$$

计算搜索窗的质心：

$$x_c = M_{10} / M_{00}; y_c = M_{01} / M_{00}$$

(3).调整搜索窗大小

宽度为 $s = \sqrt{M_{00} / 256}$ ；长度为1.2s；

(4).移动搜索窗的中心到质心，如果移动距离大于预设的固定阈值，则重复2)3)4)，直到搜索窗的中心与质心间的移动距离小于预设的固定阈值，或者循环运算的次数达到某一最大值，停止计算。关于meanshift的收敛性证明可以google相关文献。

3--camshift

将meanshift算法扩展到连续图像序列，就是camshift算法。它将视频的所有帧做meanshift运算，并将上一帧的结果，即搜索窗的大小和中心，作为下一帧meanshift算法搜索窗的初始值。如此迭代下去，就可以实现对目标的跟踪。

算法过程为：

(1).初始化搜索窗

(2).计算搜索窗的颜色概率分布（反向投影）

(3).运行meanshift算法，获得搜索窗新的大小和位置。

(4).在下一帧视频图像中用(3)中的值重新初始化搜索窗的大小和位置，再跳转到(2)继续进行。

camshift能有效解决目标变形和遮挡的问题，对系统资源要求不高，时间复杂度低，在简单背景下能够取得良好的跟踪效果。但当背景较为复杂，或者有许多与目标颜色相似像素干扰的情况下，会导致跟踪失败。因为它单纯的考虑颜色直方图，忽略了目标的空间分布特性，所以这种情况下需加入对跟踪目标的预测算法。

OpenCV实现camshift算法的例子：

```cpp
//对运动物体的跟踪：
//如果背景固定,可用帧差法 然后在计算下连通域 将面积小的去掉即可
//如果背景单一,即你要跟踪的物体颜色和背景色有较大区别 可用基于颜色的跟踪 如CAMSHIFT 鲁棒性都是较好的
//如果背景复杂,如背景中有和前景一样的颜色 就需要用到一些具有预测性的算法 如卡尔曼滤波等 可以和CAMSHIFT结合

#ifdef _CH_
#pragma package <opencv>
#endif

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

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

IplImage *image = 0, *hsv = 0, *hue = 0, *mask = 0, *backproject = 0, *histimg = 0;
//用HSV中的Hue分量进行跟踪
CvHistogram *hist = 0;
//直方图类
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;
//Meanshift跟踪算法返回的Box类
//typedef struct CvBox2D{
//CvPoint2D32f center; /* 盒子的中心 */
//CvSize2D32f size; /* 盒子的长和宽 */
//float angle; /* 水平轴与第一个边的夹角，用弧度表示*/
//}CvBox2D;
CvConnectedComp track_comp;
//连接部件
//typedef struct CvConnectedComp{
//double area; /* 连通域的面积 */
//float value; /* 分割域的灰度缩放值 */
//CvRect rect; /* 分割域的 ROI */
//} CvConnectedComp;
int hdims = 16;
//划分直方图bins的个数，越多越精确
float hranges_arr[] = {0,180};
//像素值的范围
float* hranges = hranges_arr;
//用于初始化CvHistogram类
int vmin = 10, vmax = 256, smin = 30;
//用于设置滑动条

void on_mouse( int event, int x, int y, int flags, void* param )
//鼠标回调函数,该函数用鼠标进行跟踪目标的选择
{
    if( !image )
        return;

    if( image->origin )
        y = image->height - y;
    //如果图像原点坐标在左下,则将其改为左上

    if( select_object )
    //select_object为1,表示在用鼠标进行目标选择
    //此时对矩形类selection用当前的鼠标位置进行设置
    {
        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:
        //鼠标按下,开始点击选择跟踪物体
        origin = cvPoint(x,y);
        selection = cvRect(x,y,0,0);
        select_object = 1;
        break;
    case CV_EVENT_LBUTTONUP:
        //鼠标松开,完成选择跟踪物体
        select_object = 0;
        if( selection.width > 0 && selection.height > 0 )
            //如果选择物体有效，则打开跟踪功能
            track_object = -1;
        break;
    }
}


CvScalar hsv2rgb( float hue )
//用于将Hue量转换成RGB量
{
    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 *= 0.033333333333333333333333333333333f;
    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;

    return cvScalar(rgb[2], rgb[1], rgb[0],0);
}

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

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

    if( !capture )
    //打开视频流失败
    {
        fprintf(stderr,"Could not initialize capturing...\n");
        return -1;
    }

    printf( "Hot keys: \n"
        "\tESC - quit the program\n"
        "\tc - stop the tracking\n"
        "\tb - switch to/from backprojection view\n"
        "\th - show/hide object histogram\n"
        "To initialize tracking, select the object with mouse\n" );
    //打印程序功能列表

    cvNamedWindow( "Histogram", 1 );
    //用于显示直方图
    cvNamedWindow( "CamShiftDemo", 1 );
    //用于显示视频
    cvSetMouseCallback( "CamShiftDemo", on_mouse, 0 );
    //设置鼠标回调函数
    cvCreateTrackbar( "Vmin", "CamShiftDemo", &vmin, 256, 0 );
    cvCreateTrackbar( "Vmax", "CamShiftDemo", &vmax, 256, 0 );
    cvCreateTrackbar( "Smin", "CamShiftDemo", &smin, 256, 0 );
    //设置滑动条

    for(;;)
    //进入视频帧处理主循环
    {
        IplImage* frame = 0;
        int i, bin_w, c;

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

        if( !image )
        //image为0,表明刚开始还未对image操作过,先建立一些缓冲区
        {
            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 );
            //分配掩膜图像空间
            backproject = cvCreateImage( cvGetSize(frame), 8, 1 );
            //分配反向投影图空间,大小一样,单通道
            hist = cvCreateHist( 1, &hdims, CV_HIST_ARRAY, &hranges, 1 );
            //分配直方图空间
            histimg = cvCreateImage( cvSize(320,200), 8, 3 );
            //分配用于直方图显示的空间
            cvZero( histimg );
            //置背景为黑色
        }

        cvCopy( frame, image, 0 );
        cvCvtColor( image, hsv, CV_BGR2HSV );
        //把图像从RGB表色系转为HSV表色系

        if( track_object )
        //track_object非零,表示有需要跟踪的物体
        {
            int _vmin = vmin, _vmax = vmax;

            cvInRangeS( hsv, cvScalar(0,smin,MIN(_vmin,_vmax),0),
                        cvScalar(180,256,MAX(_vmin,_vmax),0), mask );
            //制作掩膜板，只处理像素值为H：0~180，S：smin~256，V：vmin~vmax之间的部分
            cvSplit( hsv, hue, 0, 0, 0 );
            //分离H分量

            if( track_object < 0 )
```

```
            //如果需要跟踪的物体还没有进行属性提取，则进行选取框类的图像属性提取
            {
                float max_val = 0.f;
                cvSetImageROI( hue, selection );
                //设置原选择框为ROI
                cvSetImageROI( mask, selection );
                //设置掩膜板选择框为ROI
                cvCalcHist( &hue, hist, 0, mask );
                //得到选择框内且满足掩膜板内的直方图
                cvGetMinMaxHistValue( hist, 0, &max_val, 0, 0 );
                cvConvertScale( hist->bins, hist->bins, max_val ? 255. / max_val : 0., 0 );
                // 对直方图的数值转为0~255
                cvResetImageROI( hue );
                //去除ROI
                cvResetImageROI( mask );
                //去除ROI
                track_window = selection;
                track_object = 1;
                //置track_object为1,表明属性提取完成
                cvZero( histimg );
                bin_w = histimg->width / hdims;
                for( i = 0; i < hdims; i++ )
                //画直方图到图像空间
                {
                    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),
                                 cvPoint((i+1)*bin_w,histimg->height - val),
                                 color, -1, 8, 0 );
                }
            }

            cvCalcBackProject( &hue, backproject, hist );
            //计算hue的反向投影图
            cvAnd( backproject, mask, backproject, 0 );
            //得到掩膜内的反向投影
            cvCamShift( backproject, track_window,
                        cvTermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ),
                        &track_comp, &track_box );
            //使用MeanShift算法对backproject中的内容进行搜索,返回跟踪结果
            track_window = track_comp.rect;
            //得到跟踪结果的矩形框

            if( backproject_mode )
                cvCvtColor( backproject, image, CV_GRAY2BGR );

            if( image->origin )
                track_box.angle = -track_box.angle;
            cvEllipseBox( image, track_box, CV_RGB(255,0,0), 3, CV_AA, 0 );
            //画出跟踪结果的位置
        }

        if( select_object && selection.width > 0 && selection.height > 0 )
        //如果正处于物体选择，画出选择框
        {
            cvSetImageROI( image, selection );
            cvXorS( image, cvScalarAll(255), image, 0 );
            cvResetImageROI( image );
        }

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

        c = cvWaitKey(10);
        if( (char) c == 27 )
            break;
        switch( (char) c )
        //按键切换功能
        {
        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");

```

```
288.     return 0;
289. }
290. 
291. #ifdef _EiC
292. main(1,"camshiftdemo.c");
293. #endif
```

文章标签： OpenCV 算法 camshift

个人分类： OpenCV