

## 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

$$s = \sqrt{M_{00} / 256}$$

宽度为  $s$ ；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]
1. //对运动物体的跟踪：
2. //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
3. //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
4. //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
5.
6. #ifndef _CH_
7. #pragma package <opencv>
8. #endif
9.
10. #ifndef _EiC
11. #include "cv.h"
12. #include "highgui.h"
13. #include <stdio.h>
14. #include <string.h>
```

[illegible]

```

106.     p ^= sector & 1 ? 255 : 0;
107.
108.     rgb[sector_data[sector][0]] = 255;
109.     rgb[sector_data[sector][1]] = 0;
110.     rgb[sector_data[sector][2]] = p;
111.
112.     return cvScalar(rgb[2], rgb[1], rgb[0],0);
113. }
114.
115. int main( int argc, char** argv )
116. {
117.     CvCapture* capture = 0;
118.
119.     if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 && isdigit(argv[1][0])))
120.         //打开摄像头
121.         capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' : 0 );
122.     else if( argc == 2 )
123.         //打开avi
124.         capture = cvCaptureFromAVI( argv[1] );
125.
126.     if( !capture )
127.         //打开视频流失败
128.         {
129.             fprintf(stderr, "Could not initialize capturing...\n");
130.             return -1;
131.         }
132.
133.     printf( "Hot keys: \n"
134.            "\tESC - quit the program\n"
135.            "\tc - stop the tracking\n"
136.            "\tb - switch to/from backprojection view\n"
137.            "\th - show/hide object histogram\n"
138.            "To initialize tracking, select the object with mouse\n" );
139.     //打印程序功能列表
140.
141.     cvNamedWindow( "Histogram", 1 );
142.     //用于显示直方图
143.     cvNamedWindow( "CamShiftDemo", 1 );
144.     //用于显示视频
145.     cvSetMouseCallback( "CamShiftDemo", on_mouse, 0 );
146.     //设置鼠标回调函数
147.     cvCreateTrackbar( "Vmin", "CamShiftDemo", &vmin, 256, 0 );
148.     cvCreateTrackbar( "Vmax", "CamShiftDemo", &vmax, 256, 0 );
149.     cvCreateTrackbar( "Smin", "CamShiftDemo", &smin, 256, 0 );
150.     //设置滑动条
151.
152.     for(;;)
153.         //进入视频帧处理主循环
154.         {
155.             IplImage* frame = 0;
156.             int i, bin_w, c;
157.
158.             frame = cvQueryFrame( capture );
159.             if( !frame )
160.                 break;
161.
162.             if( !image )
163.                 //image为0,表明刚开始还未对image操作过,先建立一些缓冲区
164.                 {
165.                     image = cvCreateImage( cvGetSize(frame), 8, 3 );
166.                     image->origin = frame->origin;
167.                     hsv = cvCreateImage( cvGetSize(frame), 8, 3 );
168.                     hue = cvCreateImage( cvGetSize(frame), 8, 1 );
169.                     mask = cvCreateImage( cvGetSize(frame), 8, 1 );
170.                     //分配掩膜图像空间
171.                     backproject = cvCreateImage( cvGetSize(frame), 8, 1 );
172.                     //分配反向投影图空间,大小一样,单通道
173.                     hist = cvCreateHist( 1, &dims, CV_HIST_ARRAY, &ranges, 1 );
174.                     //分配直方图空间
175.                     histimg = cvCreateImage( cvSize(320,200), 8, 3 );
176.                     //分配用于直方图显示的空间
177.                     cvZero( histimg );
178.                     //置背景为黑色
179.                 }
180.
181.             cvCopy( frame, image, 0 );
182.             cvCvtColor( image, hsv, CV_BGR2HSV );
183.             //把图像从RGB表色系转为HSV表色系
184.
185.             if( track_object )
186.                 //track_object非零,表示有需要跟踪的物体
187.                 {
188.                     int _vmin = vmin, _vmax = vmax;
189.
190.                     cvInRangeS( hsv, cvScalar(0,smin,MIN(_vmin,_vmax),0),
191.                                cvScalar(180,256,MAX(_vmin,_vmax),0), mask );
192.                     //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
193.                     cvSplit( hsv, hue, 0, 0, 0 );
194.                     //分离H分量
195.
196.                     if( track_object < 0 )

```

```

197. //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
198. {
199.     float max_val = 0.f;
200.     cvSetImageROI( hue, selection );
201.     //设置原选择框为ROI
202.     cvSetImageROI( mask, selection );
203.     //设置掩膜板选择框为ROI
204.     cvCalcHist( &hue, hist, 0, mask );
205.     //得到选择框内且满足掩膜板内的直方图
206.     cvGetMinMaxHistValue( hist, 0, &max_val, 0, 0 );
207.     cvConvertScale( hist->bins, hist->bins, max_val ? 255. / max_val : 0., 0 );
208.     // 对直方图的数值转为0~255
209.     cvResetImageROI( hue );
210.     //去除ROI
211.     cvResetImageROI( mask );
212.     //去除ROI
213.     track_window = selection;
214.     track_object = 1;
215.     //置track_object为1,表明属性提取完成
216.     cvZero( histimg );
217.     bin_w = histimg->width / hdims;
218.     for( i = 0; i < hdims; i++ )
219.         //画直方图到图像空间
220.         {
221.             int val = cvRound( cvGetReal1D(hist->bins,i)*histimg->height/255 );
222.             CvScalar color = hsv2rgb(i*180.f/hdims);
223.             cvRectangle( histimg, cvPoint(i*bin_w,histimg->height),
224.                 cvPoint((i+1)*bin_w,histimg->height - val),
225.                 color, -1, 8, 0 );
226.         }
227.     }
228.
229.     cvCalcBackProject( &hue, backproject, hist );
230.     //计算hue的反向投影图
231.     cvAnd( backproject, mask, backproject, 0 );
232.     //得到掩膜内的反向投影
233.     cvCamShift( backproject, track_window,
234.         cvTermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ),
235.         &track_comp, &track_box );
236.     //使用MeanShift算法对backproject中的内容进行搜索,返回跟踪结果
237.     track_window = track_comp.rect;
238.     //得到跟踪结果的矩形框
239.
240.     if( backproject_mode )
241.         cvCvtColor( backproject, image, CV_GRAY2BGR );
242.
243.     if( image->origin )
244.         track_box.angle = -track_box.angle;
245.     cvEllipseBox( image, track_box, CV_RGB(255,0,0), 3, CV_AA, 0 );
246.     //画出跟踪结果的位置
247. }
248.
249. if( select_object && selection.width > 0 && selection.height > 0 )
250.     //如果正处于物体选择,画出选择框
251.     {
252.         cvSetImageROI( image, selection );
253.         cvXorS( image, cvScalarAll(255), image, 0 );
254.         cvResetImageROI( image );
255.     }
256.
257. cvShowImage( "CamShiftDemo", image );
258. cvShowImage( "Histogram", histimg );
259.
260. c = cvWaitKey(10);
261. if( (char) c == 27 )
262.     break;
263. switch( (char) c )
264.     //按键切换功能
265.     {
266.     case 'b':
267.         backproject_mode ^= 1;
268.         break;
269.     case 'c':
270.         track_object = 0;
271.         cvZero( histimg );
272.         break;
273.     case 'h':
274.         show_hist ^= 1;
275.         if( !show_hist )
276.             cvDestroyWindow( "Histogram" );
277.         else
278.             cvNamedWindow( "Histogram", 1 );
279.         break;
280.     default:
281.         ;
282.     }
283. }
284.
285. cvReleaseCapture( &capture );
286. cvDestroyWindow( "CamShiftDemo" );
287.

```

```
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](#)

此PDF由spygg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com