# [javaCV 之目标跟踪](http://blog.csdn.net/gctianyou/article/details/8704150)

**[OpenCV](http://lib.csdn.net/base/opencv" \o "OpenCV知识库" \t "http://blog.csdn.net/gctianyou/article/details/_blank)**应用非常的广泛，最近一段时间在学习它。 基于WEB开发的大型程序，主流是使用**[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "http://blog.csdn.net/gctianyou/article/details/_blank)**进行开发，而java处理底层的东西时速度不理想，于是通过Java调用C开发的库文件变得很流行，JavaCV就是在这个大的环境下产生了。JavaCV它实现了java和OpenCV的关联，为Java程序员提供了一个很好的处理图像和视频的API。JavaCV开发环境的搭配就不在这个地方多口舌了，Google,Baidu都有。

       下面是基于Camshift**[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "http://blog.csdn.net/gctianyou/article/details/_blank)**实现了摄像头下对运动物体的跟踪用JavaCV来实现:

import com.googlecode.javacpp.Pointer;

import com.googlecode.javacv.cpp.opencv\_highgui.CvCapture;

import static com.googlecode.javacv.cpp.opencv\_core.\*;

import static com.googlecode.javacv.cpp.opencv\_imgproc.\*;

import static com.googlecode.javacv.cpp.opencv\_highgui.\*;

import static com.googlecode.javacv.cpp.opencv\_video.\*;

public class JavaCVCamShift{

 IplImage frame, image , hsv , hue , mask , backproject , histimg ;

 IplImage[] imageArray;

 //用HSV中的Hue分量进行跟踪

 CvHistogram hist ;

 //直方图类

 int x1=0,y1=0,x2=0,y2=0;//选取对象的坐标

 int backproject\_mode = 0;

 int select\_object = 0;

 int track\_object = 0;

 int show\_hist = 1;

 CvPoint origin;

 CvPoint  cp1,cp2;

 CvRect selection;

 CvRect track\_window;

 CvBox2D track\_box;

 float[] max\_val=new float[1];

 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

 CvConnectedComp track\_comp;

 public JavaCVCamShift()

 {

   imageArray=new IplImage[1];

   CvCapture capture= cvCreateCameraCapture(0);

   cvNamedWindow("imageName",CV\_WINDOW\_AUTOSIZE);

   Pointer pointer=null;

   cvSetMouseCallback("imageName",new mouseClike(),pointer);

   track\_comp=new CvConnectedComp();

   while(true)

   {

    frame=cvQueryFrame(capture);

    if(frame==null)break;

    if( image==null )

    //image为空,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

       }

      cvCopy(frame,image);

      cvCvtColor( image, hsv, CV\_BGR2HSV );

      if( track\_object !=0)

           //track\_object非零,表示有需要跟踪的物体

           {

       double \_vmin = 10.0, \_vmax = 256.0,smin=30.0;

              cvInRangeS( hsv, cvScalar(0.0,smin,Math.min(\_vmin,\_vmax),0.0), cvScalar(180.0,256.0,Math.max(\_vmin,\_vmax),0.0), mask );

              //，只处理像素值为H：0~180，S：smin~256，V：vmin~vmax之间的部分制作掩膜板

              cvSplit( hsv, hue, null, null, null );

     //分离H分量

              imageArray[0]=hue;

      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

              {

                  cvSetImageROI( imageArray[0],selection );

                  //设置原选择框为ROI

                  cvSetImageROI( mask,selection );

                  //设置掩膜板选择框为ROI

                  cvCalcHist( imageArray,hist,0,mask );

                  //得到选择框内且满足掩膜板内的直方图

                  cvGetMinMaxHistValue( hist, null, max\_val, null, null );

                  cvConvertScale( hist.bins(), hist.bins(),max\_val[0]>0 ? (double)255/ max\_val[0]:0.0,0 );

                  // 对直方图的数值转为0~255

                  cvResetImageROI( imageArray[0] );

                  //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

              }

      cvCalcBackProject( imageArray, backproject, hist );

             //计算hue的反向投影图

             cvAnd( backproject, mask, backproject, null );

             //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

    cp1=cvPoint(track\_window.x(),track\_window.y());

    cp2=cvPoint(track\_window.x()+track\_window.width(),track\_window.y()+track\_window.height());

    if( image.origin()>0 )

          track\_box.angle(-track\_box.angle());

    cvRectangle(frame,cp1,cp2, CV\_RGB(0,255,0),3,CV\_AA,0);

           }

     if( select\_object==1 && selection.width() > 0 && selection.height() > 0 )

           //如果正处于物体选择，画出选择框

           {

               cvSetImageROI( frame, selection );

               cvXorS(frame,cvScalarAll(255),frame,null );

               cvResetImageROI( frame );

           }

    cvShowImage("imageName",frame);

    int c=cvWaitKey(33);

    if(c==27) break;

   }

   cvReleaseCapture(capture);

   cvDestroyWindow("imageName");

 }

 public static void main(String[] args) {

         new JavaCVCamShift();

 }

class mouseClike extends CvMouseCallback

{

  public void call(int event,int x, int y,int flags, Pointer param)

 //鼠标回调函数,该函数用鼠标进行跟踪目标的选择

 {

     if( image==null )

         return;

     if( image.origin()!=0 )

         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==1 )

     //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(Math.min(x,origin.x()));

         selection.y(Math.min(y,origin.y()));

         selection.width(selection.x() + Math.abs(x - origin.x()));

         selection.height(selection.y() + Math.abs(y - origin.y()));

         selection.x(Math.max(selection.x(),0));

         selection.y(Math.max(selection.y(),0 ));

         selection.width(Math.min( selection.width(), image.width() ));

         selection.height(Math.min( selection.height(), image.height()));

         selection.width(selection.width()-selection.x());

         selection.height( 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(0,0,0,0);

         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;

     }

 }

}

}

      使用JavaCV我们要注意的是，Google并没有给我们一些API参考文档，调用函数的名称和参数可能与OpenCV有一定的出入，（回调函数的使用改变很大），需要我们多看看一些JavaCV自带的例子。JavaCV环境的搭配和**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "http://blog.csdn.net/gctianyou/article/details/_blank)**请参考:[http://blog.csdn.net/gctianyou/article/details/8707805](http://blog.csdn.net/gctianyou/article/details/8707805" \t "http://blog.csdn.net/gctianyou/article/details/_blank)

