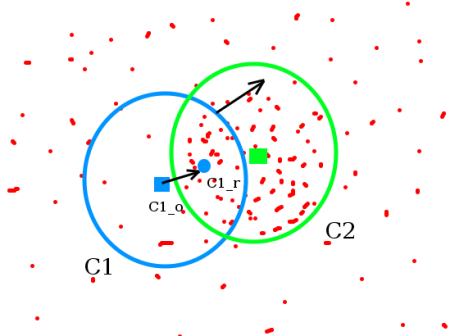
**目标：**

* 学习Meanshift和Camshift算法来寻找和跟踪视频中的物体。

**Meanshift**

Meanshift 算法的基本原理很简单。假设有一堆点（比如直方图反向投影得到的点），和一个小的窗口（可能是一个圆形窗口），然后将这个窗口移动到最大像素密度处（点最多的地方）。如图所示： [](https://camo.githubusercontent.com/6d37f0380bdf428fc6df1c142e16653387d8fd94779f2965e78e587495c27c8e/68747470733a2f2f646f63732e6f70656e63762e6f72672f342e302e302f6d65616e73686966745f6261736963732e6a7067)

初始窗口是蓝色的“C1”，它的圆心为蓝色方框“C1\_o”，而窗口中所有点质心却是“C1\_r”(小的蓝色圆圈)，很明显圆心和点的质心没有重合。所以移动圆心 C1\_o 到质心 C1\_r，这样我们就得到了一个新的窗口。这时又可以找到新窗口内所有点的质心，大多数情况下还是不重合的，所以重复上面的操作：将新窗口的中心移动到新的质心。就这样不停的迭代操作直到窗口的中心和其所包含点的质心重合为止（或者有一点小误差）。按照这样的操作我们的窗口最终会落在像素值（和）最大的地方。如上图所示“C2”是窗口的最后位址，我们可以看出来这个窗口中的像素点最多

## 实例，参考官方demo，网上很多人的都是错误的，没有任何效果

|  |  |
| --- | --- |
|  | **import numpy as np import cv2 as cv  cap = cv.VideoCapture('../datas/slow.mp4')  *# take first frame of the video* ret,frame = cap.read()  *# setup initial location of window # x, y, w, h = 300, 200, 100, 50 # simply hardcoded the values* x, y, w, h = 320,250,125,90 *# simply hardcoded the values* track\_window = (x, y, w, h)  *# set up the ROI for tracking* roi = frame[y:y+h, x:x+w] hsv\_roi = cv.cvtColor(roi, cv.COLOR\_BGR2HSV) mask = cv.inRange(hsv\_roi, np.array((0., 60.,32.)), np.array((180.,255.,255.))) roi\_hist = cv.calcHist([hsv\_roi],[0],mask,[180],[0,180]) cv.normalize(roi\_hist,roi\_hist,0,255,cv.NORM\_MINMAX)  *# Setup the termination criteria, either 10 iteration or move by atleast 1 pt* term\_crit = ( cv.TERM\_CRITERIA\_EPS | cv.TERM\_CRITERIA\_COUNT, 10, 1 )  while(1):  ret ,frame = cap.read()   if ret == True:  hsv = cv.cvtColor(frame, cv.COLOR\_BGR2HSV)  dst = cv.calcBackProject([hsv],[0],roi\_hist,[0,180],1)   *# apply meanshift to get the new location* ret, track\_window = cv.meanShift(dst, track\_window, term\_crit)   *# Draw it on image* x,y,w,h = track\_window  img2 = cv.rectangle(frame, (x,y), (x+w,y+h), 255,2)  cv.imshow('img2',img2)   k = cv.waitKey(60) & 0xff  if k == 27:  break   else:  break  cv.destroyAllWindows() cap.release()** |

### 效果，类似以下面的效果，矩形的大小和位置会根据画面的变化而变化

|  |
| --- |
| https://camo.githubusercontent.com/503cf0de2a733a88ce0febb70281a40a79a28c57ac0086775e28a86fd6834550/68747470733a2f2f646f63732e6f70656e63762e6f72672f342e302e302f6d65616e73686966745f726573756c742e6a7067 |

## Camshift

这里面有一个问题，我们的窗口的大小是固定的，而汽车由远及近（在视觉上）是一个逐渐变大的过程，固定窗口是不合适的。所以我们需要根据目标的大小和角度来对窗口进行调整。1988年，OpenCVLabs 提出了一个解决方案：CAMshift （Continuously Adaptive Meanshift）算法，由Gary Bradsky发表在他的论文“Computer Vision Face Tracking for Use in a Perceptual User Interface”中。

Camshift算法首先应用meanshift。一旦meanshift收敛，它就会更新窗口的大小，$$ s = 2 \times \sqrt{\frac{M\_{00}}{256}} $$。它还计算最佳拟合椭圆的方向。同样，它将新的缩放搜索窗口和先前的窗口位置应用于meanshift。继续该过程直到满足所需的准确度。

## OpenCV 中的 Camshift

它与meanshift几乎相同，但它返回一个旋转的矩形（这是我们的结果）和box参数（用于在下一次迭代中作为搜索窗口传递）

## 实例，参考官方

|  |  |
| --- | --- |
|  | **import numpy as np import cv2 as cv  cap = cv.VideoCapture('../datas/slow.mp4')  *# take first frame of the video* ret,frame = cap.read()  *# setup initial location of window # x, y, w, h = 300, 200, 100, 50 # simply hardcoded the values* x, y, w, h = 320,250,125,90 *# simply hardcoded the values* track\_window = (x, y, w, h)  *# set up the ROI for tracking* roi = frame[y:y+h, x:x+w] hsv\_roi = cv.cvtColor(roi, cv.COLOR\_BGR2HSV) mask = cv.inRange(hsv\_roi, np.array((0., 60.,32.)), np.array((180.,255.,255.))) roi\_hist = cv.calcHist([hsv\_roi],[0],mask,[180],[0,180]) cv.normalize(roi\_hist,roi\_hist,0,255,cv.NORM\_MINMAX)  *# Setup the termination criteria, either 10 iteration or move by atleast 1 pt* term\_crit = ( cv.TERM\_CRITERIA\_EPS | cv.TERM\_CRITERIA\_COUNT, 10, 1 )  while(1):  ret ,frame = cap.read()   if ret == True:  hsv = cv.cvtColor(frame, cv.COLOR\_BGR2HSV)  dst = cv.calcBackProject([hsv],[0],roi\_hist,[0,180],1)   *# apply meanshift to get the new location* ret, track\_window = cv.CamShift(dst, track\_window, term\_crit)  *# Draw it on image* pts = cv.boxPoints(ret)  pts = np.int0(pts)  img2 = cv.polylines(frame,[pts],True,255,2)  cv.imshow('img2',img2)   k = cv.waitKey(60) & 0xff  if k == 27:  break   else:  break  cv.destroyAllWindows() cap.release()** |

## 效果：和meanShift类似，只不过这里的矩形会旋转

|  |
| --- |
| image5 |