track

2016060107001 韦嗣千

- 使用方法: **opencv camshift**
  - 概述
    camshift 算法需要初始化一个搜索窗口，然后计算这个窗口里的颜色直方图(图片)然后将其转化为概率分布图，以这个概率分布直方图寻找下一张图片窗口的中心，然后调整窗口位置。
  - 缺陷
    - 在用摄像头做实验时发现，当物体运动过快时，会跟丢。因为camshift在搜索目标时是在上一帧的搜索框附近搜索。物体运动速度不高时，物体在两帧之间的位置不会差太多，所以这个策略能减少计算。当物体嗖的一下就跑出了搜索框，自然就跟丢了。解决方法有一个叫运动预测。通过运动方程预测它下一帧会出现在哪。
    - 另一个就是用摄像头进行人脸跟踪时转个身它就不认了。这应该是因为我还没秃，后面颜色和前面不一样，颜色直方图算出来不一样的原因吧。
  - 适用范围
    - 首先我觉得如果摄像头是静止的会好点。因为如果摄像头也在动，又不是在跟拍物体的相对速度可能会变快，就会跟丢了。
    - 跟踪物体和背景的颜色差异应该明显一点。因为camshift将色调的直方图作反向投影得到它的概率分布来进行跟踪。如果颜色差别不大，也就是色调差不多，就会有问题。
- 搜索过程
  - 初始化搜索框
    经过实验，找到一个较好的搜索框，直接写到程序里。
  - 颜色空间转换
    光亮的变化对RGB空间有很大影响，所以canshift算法一般将RGB空间转化为HSV空间。
  - 计算颜色分布直方图
  - 计算直方图H的反向投影
  - 计算搜索窗口质心，改变窗口位置，不断迭代，直到收敛。
- 程序设计
  无非是把写好的库直接调用，了解算法的搜索过程后这就没什么难的，就列几个关键函数吧。

```
    cvtColor(InputArray src,OutputArray dst,int code)
    //将Input的图像转化为code的颜色空间存在Output中。
    calcHist()//计算一串图片的颜色分布直方图，参数太多就不列举了。
    calcBackProject()//算反向投影图
```

```
    meanShift()//meanShift本体，也是算搜索框的地方
```

在 opencv.org里都有详细介绍，包括算法，我就从那学的。

- 结果
  大概跟丢了100帧，当鱼再跑进框里的时候又接上了。算是个失败的作品。实际中谁能保证鱼又跑回去呢。

- 总结
  camshift虽然有其固有的缺陷，但还是有其它方法能将其修正，各种滤波，各种预测，我都不会，所以只能交个还凑合的结果。上了老师的课，我最大的收获是觉得数学真美妙。有些算法感觉上都能想出来，若只是傻傻的编程，就算能写出来，代码肯定很复杂。经过数学那么一推导，不仅效率与正确性有保障，代码看着也漂亮。之前学的微积分与线性代数，我都只抱着及格就行的态度去学，现在看来荒谬至极。我的专业是计算机科学与技术，没有科学哪来技术?而数学才是科学的基础。之前我一直浪费时间在所谓的技术上，但我觉得，现在意识到还不算晚。谢谢老师！

Tips:因为我的opencv环境是搭在Linux上的，我就只交代码不叫可执行文件了。如果老师要运行：

P----开始/暂停
ESC–退出
顶上那个是进度条

我再交个录屏结果