

**计算机视觉作业报告**

|  |  |
| --- | --- |
| 作业名称： | 光流法的实现 |
| 姓 名： | 程书意 |
| 学 号： | 21821255 |
| 电子邮箱： | 1541189572@qq.com |
| 联系电话： | 13100673675 |
| 导 师： | 李红 |

2018年 1 月 3 日

光流法的实现

1. 作业已实现的功能简述及运行简要说明
   1. 功能
      1. 读取eval\_data目录下的相应的测试样例，生成相应的result。
      2. 样例通过进行canny算子计算出特征点，再将特征点进行LK算法，计算出两帧之间的该特征点的速度。
      3. 不同的速度，在图片上画不同长度的红色线段。
      4. 读取根目录下的测试视频，并将相应的光流绘制在视频上，保存在根目录。
   2. 运行说明
      1. Python optical\_flow.py 即可运行，并将生成的结果放在相应的目录下。
      2. eval\_data的测试样例，计算结果就放在eval\_data目录下，前缀为result。
      3. 视频output.avi测试样例，计算结果文件是output\_result.avi文件。
2. 作业的开发与运行环境
   1. 操作系统是win7 64bit
   2. 编程语言是python 3
   3. 需要的依赖库有，opencv scipy numpy os等
3. 系统或算法的基本思路、原理、及流程或步骤等
   1. 读取图片或者视频的前两帧
   2. 调用opencv的cvtColor函数，将读入的图片从彩色变成灰度图。
   3. 光流计算部分：
      1. 利用opencv的canny算法，找到边缘点，即特征点。
      2. 将灰度图进行归一化，即img/255
      3. 对归一化的图像分别求卷积，利用scipy的signal.convolve2d求灰度图的卷积。对卷积核的设置不同，将对图像的卷积转化成相应的偏导。
      4. 对每一个特征点，在设置的win大小的方框内，进行LK算法的计算，求出相应的u和v。
   4. 根据返回的起始点和终止点，在彩色图像绘制出红色线段。
4. 具体如何实现，包括关键（伪）代码、主要用到函数与算法等
   1. cv.imread读取图像
   2. cv.cvtColor将读取的彩色图像转换成灰度图
   3. cv.Canny找到特征点
   4. scipy.signal.convolve2d对图像求卷积，相应的卷积核如下：
      1. kernel\_x = np.array([[-1., 1.], [-1., 1.]])，卷积的结果是图像在x方向上的偏导。
      2. kernel\_y = np.array([[-1., -1.], [1., 1.]])，卷积的结果是图像在y方向的偏导。
      3. kernel\_t = np.array([[1., 1.], [1., 1.]])，卷积的结果是两张相邻图像对时间t的偏导。该卷积过程是两个卷积过程的结合，具体代码如下：

ft = signal.convolve2d(s\_img, kernel\_t, boundary='symm', mode='same') + signal.convolve2d(f\_img, -1\*kernel\_t, boundary='symm', mode='same')

* 1. 下面的代码是对每一个特征点进行LK算法的计算，求出速度。

for i,j in e\_points:

#选中窗口

Ix = fx[i-win:i+win+1, j-win:j+win+1].flatten()

Iy = fy[i-win:i+win+1, j-win:j+win+1].flatten()

It = ft[i-win:i+win+1, j-win:j+win+1].flatten()

#求b

b = np.reshape(It, (It.shape[0],1))

#求A

A = np.vstack((Ix, Iy)).T

temp = np.zeros(2)

#求速度

temp = np.matmul(np.linalg.pinv(A), b)

* 1. 有了特征点和相应的速度，可以计算出起点和终点，起点就是特征点的位置，终点就是特征点加上速度。
  2. 调用cv,line函数绘制红色的线段。

1. 实验结果与分析
   1. 控制台输出结果，测试样例采用了Army、Backyard和Basktball

------------------begin parse image---------------------

Army done!

Backyard done!

Basketball done!

------------------end parse image---------------------

------------------begin parse video-------------------

第0帧处理完毕

…

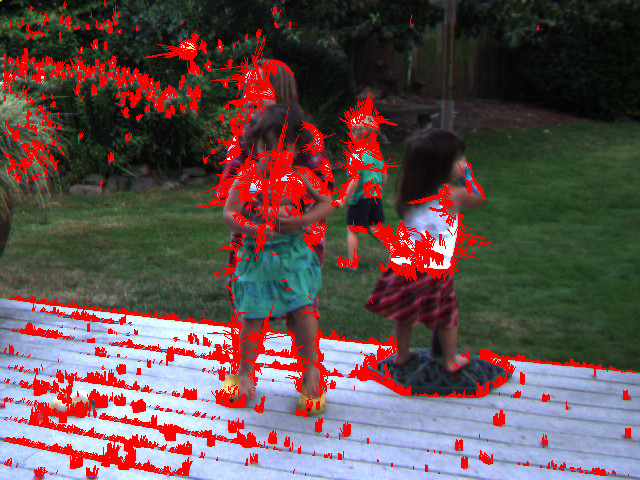
第104帧处理完毕

------------------end parse video---------------------

* 1. Army输出结果部分展示，可以看出边缘部分处理的不错。



* 1. Backyard输出结果部分展示，可以看出边缘部分处理的不错。但是点太多，有点混乱。



* 1. Basktball输出结果部分展示。可以看出，效果不是很好，和Backyard一样，很多杂乱的点（将衣服内部的轮廓视为特征点）。以及篮球的边缘并没有很好的匹配到。



* 1. 视频处理结果见根目录：result-output.avi。

1. 结论与心得体会
   1. 虽然利用LK算法可以计算出特征点的运动速度，但是如果在一开始用canny没有找到合适的特征点的话，会影响最终的效果。
   2. 特征点过于多，不能够很好的表现出物体运动的方向。
2. 参考文献

[1] python 实现lucal kanade的光流算法. <https://sandipanweb.wordpress.com/2018/02/25/implementing-lucas-kanade-optical-flow-algorithm-in-python/>

[2] python numpy库的使用. <http://www.numpy.org/>

[3] python opencv库的使用. https://opencv.org/

[4] python scipy库的使用. https://www.scipy.org/