山东大学 计算机科学与技术 学院

计算机视觉 课程实验报告

|  |  |  |
| --- | --- | --- |
| 学号：201900161140 | 姓名：张文浩 | 10.16 |
| 实验题目：图像统计特征 | | |
| 实验过程中遇到和解决的问题：  （记录实验过程中遇到的问题，以及解决过程和实验结果。可以适当配以关键代码辅助说明，但不要大段贴代码。）    实验原理：  通过搜索下一帧图像中当前帧的位置周围一定区域的直方图，计算直方图之间的显示度，找到最相近的一个框，就判定为下一帧中目标的矩形区域。  实验步骤：  **1**.初始化，启动函数加载视频，并交互框出目标图像  鼠标左键按下开始绘制目标矩形，松开左键绘制结束，关键代码如下：    **2**.绘制RGB颜色直方图  调用opencv自带的计算直方图的函数，分别计算三个颜色通道的直方图    计算完毕，绘制直方图    **3**.因为采用HSV颜色格式最终效果比RGB的效果好，所以现在将图像的RGB颜色格式换成HSV的颜色格式。  在HSV颜色格式下，再次计算刚才抠出来的目标矩形的直方图，方便后续计算相似度    对直方图做归一化处理    **4**.扫描搜索区域，绘制目标矩形。  首先确定搜索范围，如果搜索范围太大，就会增大计算量，造成视频卡顿；如果搜索范围太小，在目标物体移动太快的时候无法追踪到物体。  经过多次尝试，最终将搜索范围定为了目标矩形的三倍大小    C:\WINDOWS\TEMP\WeChat Files\81171d0075b1cb68ff08f6fde564f4f.jpg  如原理图所示，利用两个for loop遍历粉色的搜索区域  在两个for loop中，每次计算搜索到的矩形区域的直方图，利用函数compHist计算当前搜索的矩形区域与目标区域的相似性，cursim越小，说明越相似，最后找到最相似的矩形区域，作为下一帧的目标区域。    在原始视频图像上刷新矩形，只有当与目标直方图很相似时才更新起点搜索区域，满足目标进行移动的场景    绘制最相似的矩形区域    其中计算两个直方图的显示度的函数compHist实现如下;  现将RGB转换为HSV格式，然后计算直方图，也进行归一化操作，然后调用opencv提供的compareHist函数，采用巴氏距离比较两个直方图的相似性。    实验结果： | | |
| 其中红色的矩形是跟踪的物体，紫色的矩形就是搜索范围矩形的左上角的范围，即下面这个图里面的粉色的矩形  C:\WINDOWS\TEMP\WeChat Files\81171d0075b1cb68ff08f6fde564f4f.jpg  这个视频放在了附件中  另一个视频：    结果分析与体会：  在本次实验中，通过实现基于直方图的目标跟踪进行了对直方图的应用。总结一下实验中遇到的几个问题：  1.搜索的时候不能在整张图像上进行搜索，不然会导致计算过量太大，造成视频卡顿，所以我们智能假设物体不会瞬移，即下一帧中的物体不会在距离当前帧物体距离很远的地方出现。于是我们在搜索的时候就可以只在当前物体位置的周围进行搜索，这个范围大小也算是一个超参，如果搜索范围太大，就会增大计算量，造成视频卡顿；如果搜索范围太小，在目标物体移动太快的时候无法追踪到物体。  2.在原始图像上刷新矩阵的问题，因为我们只在物体周围一定范围内进行搜索，所以显然这个搜索矩阵一定是要进行移动的，而移动的时机是当搜索到的目标矩形与目标非常相关时，才更新搜索矩形，这样可以减少噪声的干扰，增加稳定性。但我在实验的过程遇到的问题是，一开始把判定“非常相关”的阈值设置得太苛刻了，所以搜索矩阵始终不移动，等到物体超出搜索矩阵的时候，搜索矩阵还停留在原地，导致“跟丢”物体。于是我把这个阈值设置得大一点后就解决了这个问题。  3.计算速度的问题：整个计算过程中最消耗时间的就是在搜索过程中的两个for loop，所以可以考虑减少循环的次数来加快速度。因为视频中一帧图像的大小通常比较大，都是几百大小的，所以个位数的误差可以忽略不计，我们可以利用这一点，再循环的时候不是每次移动一个像素，而是移动多个像素，即增大补偿step，这样可以明显的减少循环次数，同时对效果影响不大。 | | |