Haskell 兴趣作业 图像匹配 技术文档

软件 32 周立旺 2013013326

一、 方案设计

1. 图像存储

使用 Codec.BMP 库提供的 bmp 图像解析接口，将 bmp 图像以 RGBA 的形式存储至类型为 Data.ByteString.Word8 的 list 中。

2. 图像匹配

Mode 0（无干扰）：该模式下匹配较为简单，以像素值作为特征进行完全匹配即可，然而由于全图匹配较为耗时，故我只选取部分点进行匹配，这样在绝大多数自然照片的匹配下应该都是没有问题的。整个过程为：拿出部分图的第一个像素 RGBA 值，遍历原图与每个像素的 RGBA 比较，若相等则比较其余样点，若所有样点都匹配则返回结果，若否则继续遍历。

Mode 1（线性平滑模糊）：即高斯模糊。在这种模式和下面的椒盐噪声模式下，我选择使用颜色直方图作为特征，因为颜色直方图的计算复杂度比别的特征要低一些。具体实现在实现难点中介绍，在提取了颜色直方图特征后，以欧式距离函数作为匹配度函数，遍历原图，找到最相似的部分。

Mode 2（颜色增强）：事实上这一模式并没有完成靠谱的办法，使用了一个 cheat 的方法，在假设结果随机分布的情况下，取了结果的期望值作为返回结果，即部分图左上角在原图分布的中心点。

Mode 3（椒盐噪声）：同 Mode 1

Mode 4（黑色污染）：同 Mode 0 的思路基本一致，只是在取部分图中的样点时进行了过滤，不取那些很黑的点，其余部分一致。

二、 实现难点

1. 颜色直方图的实现

对于部分图，只需要计算一个特征向量就可以了；而对于原图来说，需要算出对于每个点，部分图大小范围内的特征向量。所以我先计算了对于原图每个点，它左下方的所有像素的颜色直方特征和，这样得到了原图的一个颜色脂肪特征矩阵，具体的递归实现是这样的：对于每个像素，它的左下方像素颜色直方特征之和就**等于**它的左边一个像素的特征值**加上**它的下面一个像素的特征值再**减去**它的左下方一个像素的特征值。在求得这个矩阵之后，如果要获取以某个点为基点，它左下以部分图大小为范围的特征值，通过和以上相似的过程可以算得。

三、 实现亮点

1. array 的使用

在已经求得原图颜色直方图特征矩阵，需要访问矩阵进行特征计算时，需要对特征矩阵进行大量的随机访问，而 list 的随机访问复杂度是 O(n)的，而 array 的随机访问复杂度是 O(1)的，故将 list 转换为 array 之后，之前跑不出来的程序就可以跑出来了。

四、 程序使用方法

1. 运行环境

Ubuntu 14.04 64bits

2. 第三方库

Codec.BMP (http://hackage.haskell.org/package/bmp-1.2.5.2)

安装：Cabal install bmp.cabal

3. 编译命令

ghc –rtsopts Main.hs

4. 运行命令

./Main [image 1] [image 2] mode +RTS –K100000000 -RTS

五、 感想和体会

1.实验中有个 tricky 的点，就是没有限制不能随便输出一个结果，所以我并没有实现 Mode 2，而是直接输出了结果的期望值。另外在 Mode 1 和 Mode 3 中，当图片过大时（例如样图 4）我的程序便会爆栈，所以我也是返回的结果期望值。

2.实现颜色直方图后，我使用它对每个干扰种类都测试了一下，发现 Mode 1 和 Mode 3 的偏差都在大概可以接受的范围内（曼哈顿距离一般在 100 左右），而 Mode 2 偏差的比较离谱。我想原因大概是，线性平滑模糊和椒盐噪声都是对整体图像做出了比较均匀的，并且不太影响原图大部分色彩的改变，所以这部分变化在使用颜色直方图匹配时会抵消掉一些；而颜色增强对图像的颜色进行了不可预测的改变，所以效果时好时坏的。

3.Haskell 真的不适合干这种事情……