2016

UniqImg 项目分析

多媒体技术大作业实验报告

目录

一．项目介绍

本项目命名为 UniqImg，即 Unique Image 的组合。这是一个使用 PyQt4 编写的桌面应用，目的是将指定目录下的图像根据相似度进行归类，然后可以让用户选择删除某些重复的图像来节省磁盘空间。

本项目的实现的基于相似图像检索，诸如 TinyEye（http://www.tineye.com/）之类的网站。图像相似度的判定采用 Image Hash 算法，包括 Average Hash(aHash)，Perceptual Hash(pHash) 以及 Difference Hash(dHash)。

下面文档将分四部分讲述本项目，包括：

1. 操作指南，将讲解如何安装，操作本项目的应用 UniqImg，以及项目的应用场景；
2. 算法分析，将讲解 aHash，pHash，dHash 以及图像检索算法的原理，也是本文档的最核心的部分；
3. 代码结构，将讲解如何运行项目代码，以及代码的结构；
4. 难点解析，将讲解代码部分核心实现，以及项目在开发过程中所遇到的困难。

二．操作指南

本项目提供了 debian 安装包，在 Ubuntu16.04 下打包，所以建议在 Ubuntu16.04 下安装此应用。

1. UniqImg 安装方法

进入项目根目录，可以看到 uniqimg_0.1.2_all.deb 文件，然后在此次打开 Terminal，使用如下命令进行安装。

```
sudo dkpg -i uniqimg_0.1.2_all.deb
sudo apt -f install
```

Table 1 UniqImg **安装命令**

其中第一行命令用于安装应用 UniqImg，第二个命令用于安装应用的相关依赖。

安装完成之后，可以 dash 中看到一个名为 UniqImg 的应用，点击即可运行。另外，也可在 Terminal 下输入 uniqimg 来启动应用。

2. UniqImg 操作方法

UniqImg 的操作十分简单，大概可分为如下 6 个步骤：

- 点击“Open Folder”按钮，在弹出的窗口中选择要查找相似图像的路径
- 在 toolbar 中选择 Image Hash 算法（默认为 aHash）以及路径检索方法（其中 current 指的是只查找指定路径，不包括子目录，而 recursive 指的是递归查找，即包括全部子目录，默认为 recursive）
- 点击“Search”按钮, 开始查找重复或者相似的图像
- 查找结果中会显示多个图像集合，点击打开想用去重的图像集合
- 在弹出的 dialog 中，会显示该集合下图像的路径列表，单击可在右侧看到图像的预览
- 勾选想要删除的图像，点击”Remove“按钮，即可删除冗余的图像。

UniqImg 还提供了如下的快捷键支持：

- 打开首页： Ctrl+H
- 打开目录：Ctrl+O
- 退出应用: Ctrl+Q
- 搜索: Enter（注意，仅当路径编辑行被激活时有效)

更多用户指导，可点击 Menu Bar 中的 Help -> User Guide 查看

3. UniqImg 应用截图

Figure 1 UniqImg 首页

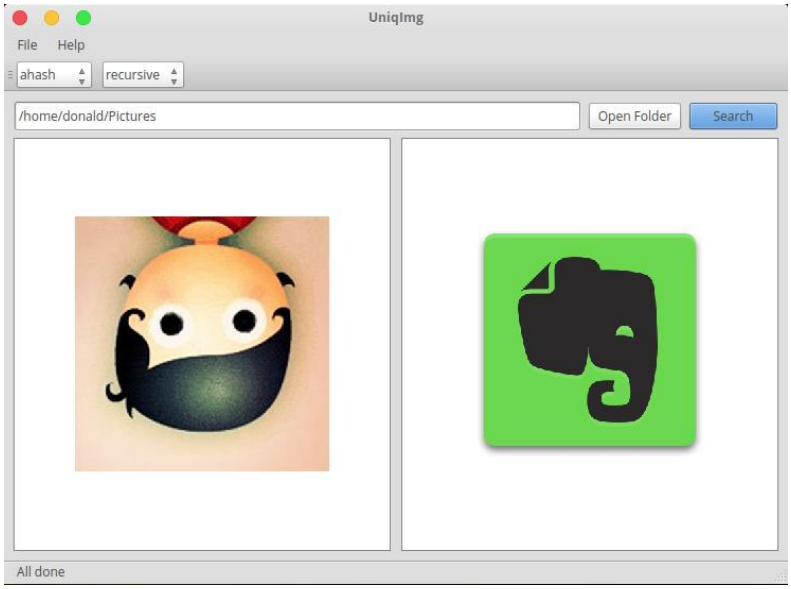

Figure 2 UniqImg 查询结果样例

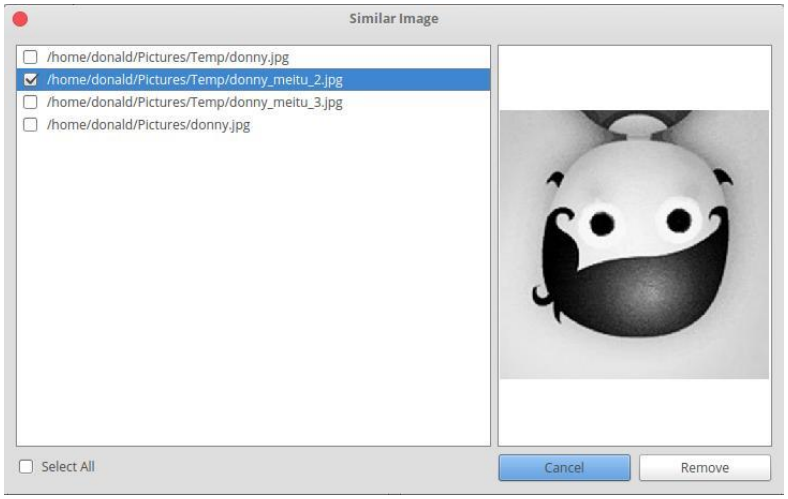

Figure 3 相似图像 1

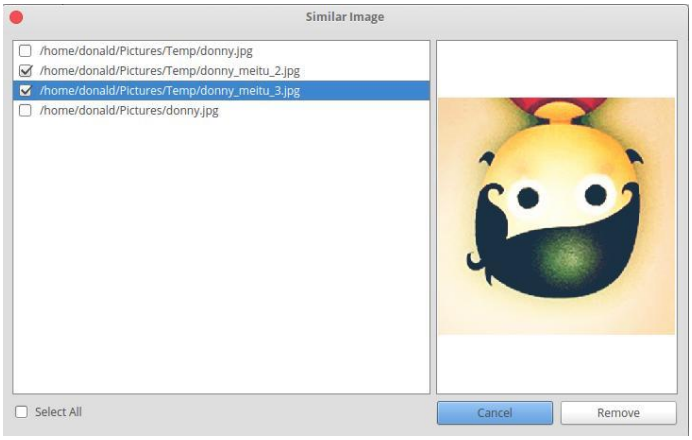

Figure 4 相似图像 2

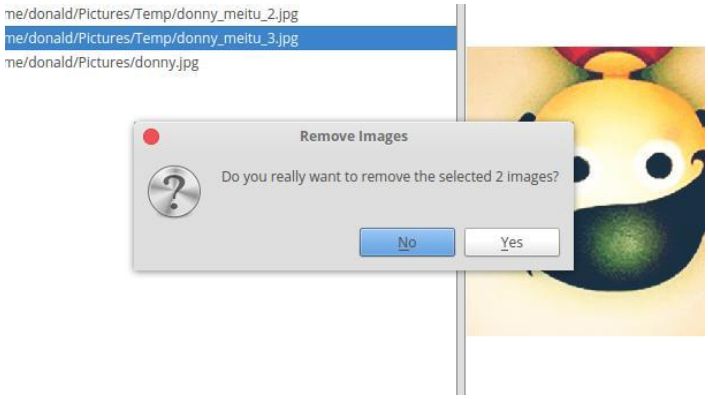

Figure 5 删除图像对话框

Figure 6 使用 pHash 算法，有较好的鲁棒性。上下两图有一点差别，但仍然可以识别为相似图像

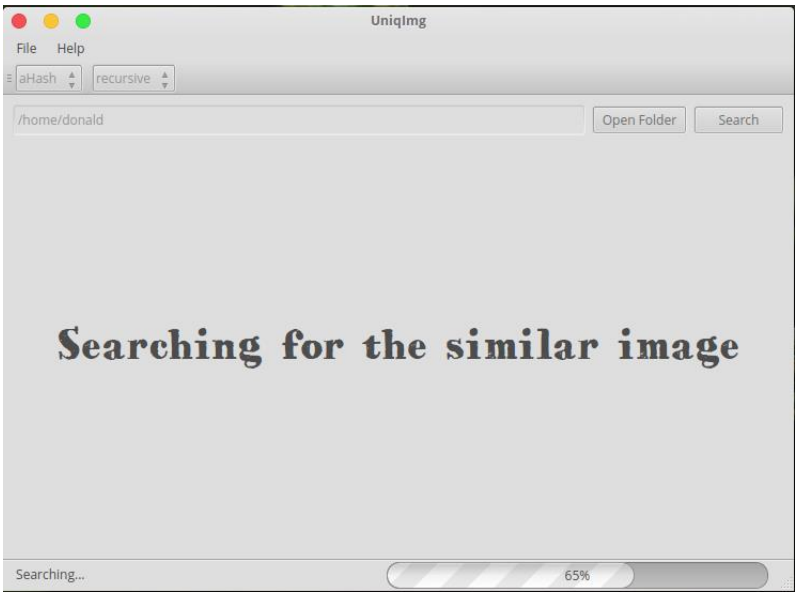

Figure 7 **通过多线程实现等待效果，提升用户体验。详见第五部分难点解析** 3

4. UniqImg 应用场景

如文档开篇所说，本项目可以用于查找指定路径下重复的图像，删除多余的图像来节省磁盘空间。

另外，项目也可以应用于如下场景：用户由于某些原因只能拿到一张图像的缩略图，但记得原高清大图大概在某些路径下，这时就可以把缩略图放到该路径下，然后使用 UniqImg 查找该路径，在查找结果中打开该对应的图像集合，在弹出的 dialog 中便可看到该高清大图的路径。

三．算法分析

UniqImg 会判断指定路径下全部图像的相似度，将相似的图像归入同一集合。但是，直接逐个像素对比来判定两张图像之间是否相同，这是非常耗时的操作，不适用于桌面应用，而且效果不好，因为很容易受干扰。所以在 UniqImg 中，使用了 Image Hash 算法来给图像生成对应的 hash 值，然后通过比较 hash 值来判定图像之间的相似性。

1. Average Hash

也称为 Mean Hash，是 Image Hash 中最基础的算法，它就是把图像变换为 8 x8 的灰度图矩阵，然后根据每个点上的值是否大于平均数来置为 1 或 0，然后将得到的二进制数转化为十六进制数。具体实现步骤如下：

1) **缩小图像**。大的图像中包含了很多细节，这些是高频信号。所以第一步中算法将通过缩小图像来移除高频信号，只保留低频信号。通常将图像缩小为 8x8，不用考虑图像的高度和宽度的比例。
2) **减少色彩**。将步骤 1 中得到的小图像转化为灰度图
3) **求平均值**。求出步骤二中得到的灰度图矩阵的平均值
4) **转为二值表示**。对于步骤二中的灰度图矩阵的每个元素，如果大于步骤三中求得的平均值，就设置为 1，否则置为 0
5) **构造 hash 值**。将步骤 5 中得到的二值矩阵按照一定顺序（比如从左到右，从上到下）展开为一个二进制整数，然后将该二进制数转化为十六进制数，就是得到了最终的图像 hash 值。

Average Hash 算法得到的 hash 值，即使图像缩小或者放大，增加或减少亮度或对比度，改

变颜色，也不会发生太大的变化。而且由于没有太耗时的运算，所以 Average Hash 非常快。

2. Perceptual Hash

Perceptual Hash 是 Average Hash 的一个扩展版，提升了鲁棒性。它大体上与 Average Hash 相似，它不是直接使用灰度值来求均值，而是加了离散余弦变换(DCT)，具体实现步骤：

1) **缩小图像**，与 aHash 相似，只不过大小改为 32x32
2) **减少色彩**，与 aHash 相同
3) **计算 DCT**，与 jpeg 使用的 8 x 8 分块不同，这里用的是 32x32 的 DCT 变换
4) **缩小 DCT**，取步骤四中得到的 DCT 矩阵的前八行中的第二列到第九列，构成一个 8x8 的矩阵，这个矩阵代表了图像的低频信号。DCT 矩阵中（ 0 ， 0 ）为直流系数，与其他交流系数会有较大的差别，会干扰后续的取均值运算，所以这里取的是第二到第九列
5) **将 DCT 矩阵转为二值**，如同 aHash，对 DCT 矩阵求均值，然后根据对应元素是否大于均值来设置 1 或者 0
6) **构造 hash 值**，同 aHash

pHash 具有较高的鲁棒性，可以容忍诸如 gamma 变换之类的，但是由于需要求解 DCT，所以相对耗时。

3. Difference Hash

Difference Hash 也与 Average Hash 相似，只是 aHash 使用的是均值，而 dHash 改为使用梯度，具体实现步骤如下：

1) **缩小图像**，将图像大小变换为 8x9
2) **减少色彩**，将步骤一中得到的图像变为灰度图
3) **计算差别(difference)**，将步骤二中得到的灰度图矩阵，用第 n+1 列减去第 n 列，每一列有 9 个元素，刚好有 8 个差。这样，最后的结果就变为了一个 8x8 的矩阵了
4) **构造 hash 值**，把步骤三中得到的矩阵中，大于 0 的置为 1 ，小于 0 的置为 1 ，然后采用 aHash 的做法，构造出 hash 值。

dHash 结合了 aHash 和 pHash，既具有较好的鲁棒性，同时运算速度也很快。

4. Hamming Distance

Hamming Distance 指的是在两个等长字符中，字符串对应位置的不同字符的个数的和。在上述的三种 Image Hash 算法中，对于任意一张图像，应用算法之后，都会得到一个 64 位长的 hash 值，所以通过求不同图像之间的 hash 值的 Hamming Distance，就可以得出图像之间的相似程度。

一般而言，Hamming Distance 为 0 的两张图像就是极为相似的图像，为 1 到 10 的，可能就是有一些简单的变换，而大于 10 的，一般就是不同的图像了。

5. 检索相似图像算法

UniqImg 中，对于相似图像的检索，使用如下几个步骤实现：

1) 根据用户所选择的路径以及遍历方法，找出其中的所有图像文件；
2) 根据用户所选择的 Image Hash 算法，计算出所有图像文件的 hash 值，并设定所有图像的初始类别为 0
3) 找出图像中类别为 0 的图像，设定其类别为 1 ，然后找出与该图像的 Hamming Distance 小于或者等于 8 的图像，加入到该类别中。
4) 将类别加 1 ，重复步骤 3，直至所有图像都标上了类别，结束算法，返回图像分类的结果。

四．代码结构

Figure 8 **代码结构图**

如图，最外围的 DEBIAN 文件夹，install.sh，uninstall.sh 都是 debian 打包的描述性文件，如果想要打包新的程序，运行 mkpkg.sh 脚本就可以了。

主要的源码在 src 文件夹中，其中

- backend 文件夹中存放了第四部分所讲的算法的实现文件
- ui 则存放了 QtDesigner 设计的应用界面，运行其中的 gen_py.sh 脚本根据 ui 文件自动生成 python 文件。
- Image 中存放的是 UniqImg 的图标，
- src 根目录下的 python 文件为前端交互逻辑控制文件

如果想要直接运行源程序，需要安装以下几个依赖：

- Python2.7
- Python-numpy
- Python-scipy
- Python-pil
- Python-qt4

然后进入 uniqimg/src 目录，运行 python uniqimg.py 即可

五．难点解析

1. 对于结果图像的点击事件，QGraphicsScene 本身并没有实现，所以，需要继承该类，然后实现 mousePressEvent 函数；同时，在新的类中，创建一个 signal，并在初始化时保存图像的类别，并在 mousePressEvent 函数中 emit 该 signal，同时包含类别。这样，在 signal 对应的 slot 函数中，就可以获取到该类别，然后根据该类别获取到对于的图像，传递给弹出的对话框了。详情可以查看 main_win.py 文件中 QScene class 以及 open_rm_dlg(self, img_cls)函数

2. 本程序的布局采用了 grid view，这样可以在调整窗口时，自动改变窗口中控件的布局。

3. 当路径下存在较多图片时，搜索过程会占用一定时间，这时会导致前端界面变成没有响应的状态。所以，需要对后台的操作使用一个新的线程来实现。本程序通过继承 QThead 来实现这一想法。详见/src/backend/service_thread.py。另外，在后台操作图像时，应该屏蔽前端的按钮等相关部件，防止用户再次点击检索，导致过多开销。

4. 如果只想获取当前路径下的文件，调用 os.listdir 函数就可以，但如果想要遍历所有的子目录，就需要使用 os.walk 函数，同时根据相对路径拼接出绝对路径，详见 services.py 文件中的_get_filepaths(directory)函数

5. Image hash 算法中，读入图像使用 PIL 模块，但是，为了方便进行矩阵运算，又将读入矩

阵改为了 numpy array 来操作。所以在 dHash 中，要特别注意 PIL 中是 resize(width, height)，而 numpy 中是 reshape(height, width)，两者参数顺序不同。详见 imagehash.py 中的 dhash(image, hash_size=8)函数中的注释。

六．参考链接

1. http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html
2. http://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html
3. http://zetcode.com/gui/pyqt4/
4. http://www.rkblog.rk.edu.pl/w/p/introduction-pyqt4/

The End
2016-10-30