数据结构2017大作业

——图片网络爬虫设计与图像处理

姓名：汤思怡

学号：161220116

班级：3

邮箱：125256623@qq.com

实验时间：2017/12/29

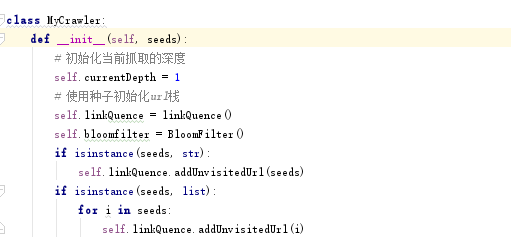
1. **实验概述**

设计、实现图片网络爬虫，从网站上批量下载图片；并对图片进行图像处理。

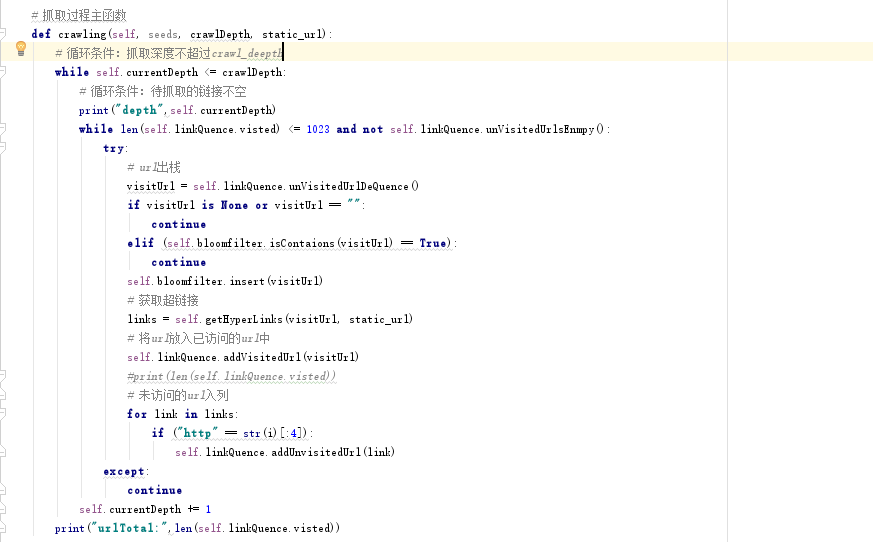
1. **实验环境**
2. 语言：网络爬虫：Python；图片处理：C++
3. 软件环境：Visual Studio2015；Pycharm2017
4. **实验设计思路**
5. **实验实现**
6. 全局变量及导入的库



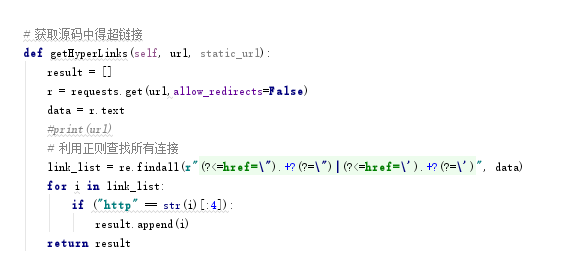
1. 网址获取

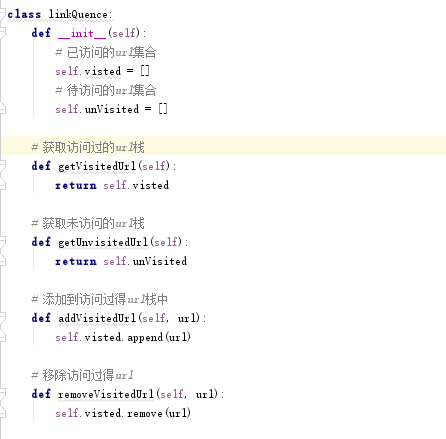


定义了一个MyCrawler类用于获得当前页面的链接网址，其中成员linkQuence用于存储已访问过的网址和即将访问的网址；成员bloomfilter类用于网页去重。



crawling函数用于抓取网页中的链接并将已访问过的网址加入已访问网址栈，当抓取深度超过自定义的需要爬取的深度即停止爬虫；当已访问过的不重复的网址超过1023或者未访问过的网址栈为空，即开始下一深度的爬取。将网址加入已访问网址栈时需要对其利用BloomFilter进行去重。爬取网址结束后打印出已访问网址栈的长度，即已访问的网址个数。

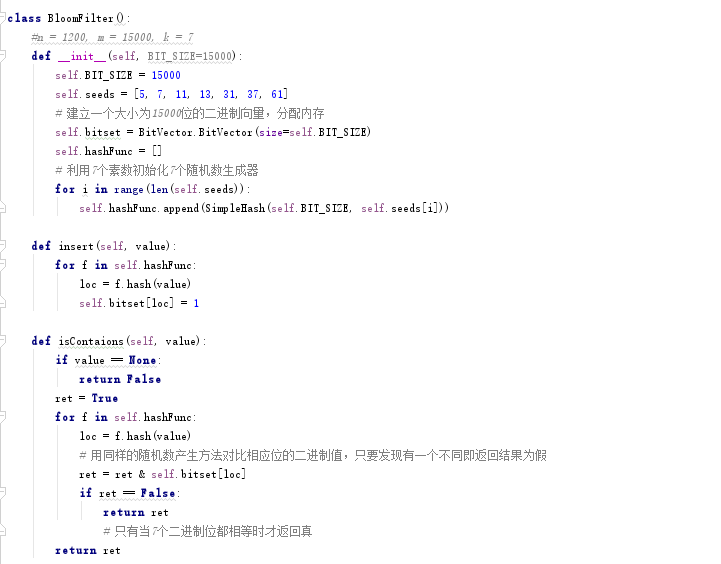
通过requests中的get函数得到网页url中的源码内容存储在data中；正则表达式r"(?<=href=\").+?(?=\")|(?<=href=\').+?(?=\')"通过匹配从data中取出标签href中的内容；if判断筛选出有效超连接。





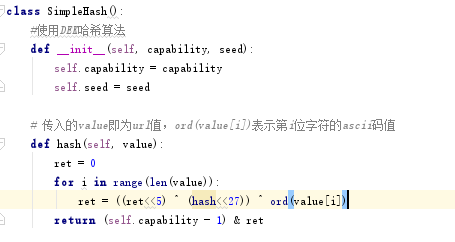
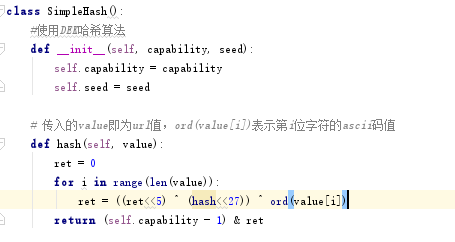
实现一个网址访问栈类。

1. 网页去重



实现一个BloomFilter类。

根据公式以及要求假阳性概率在0.1%之内，且应有至少1000个不重复的网址，则取布隆过滤器的n = 1200, m = 15000, k = 7。



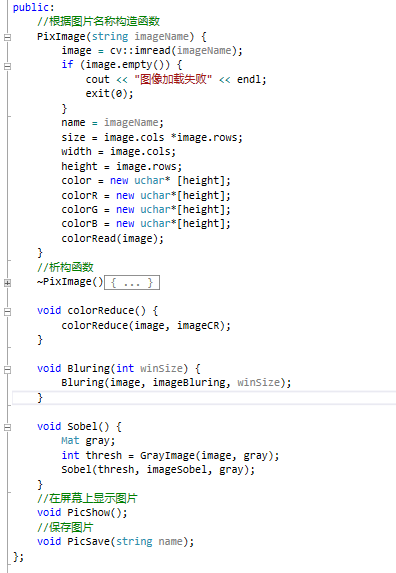
利用DEK哈希算法生成哈希值。

1. 图片下载

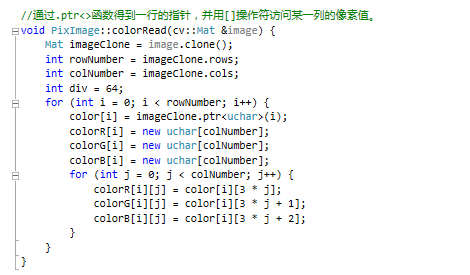
利用request库中的urlopen方法打开一个url地址并用read方法读取url上的数据；正则表达式r'src="([a-zA-Z0-9\.:/]+?\.jpg)"'用来获取url上的图片地址；再利用urlretrieve方法保存图片到本地。

1. 图像处理

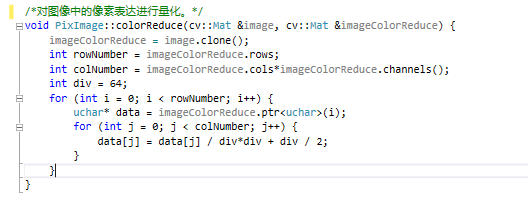


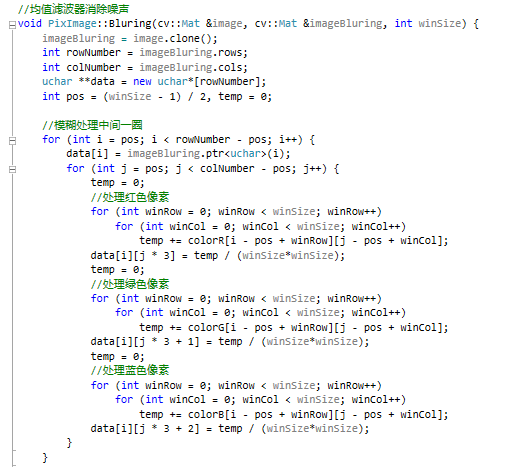


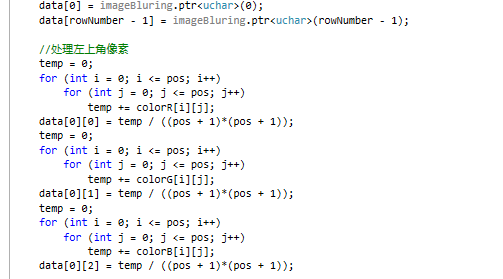
定义了一个PixImage类，private部分存有图片的尺寸、像素信息、处理后的图片信息以及图片处理方式对应的函数；public部分有相应的像素处理函数接口、中值滤波图片模糊处理接口、sobel算子处理接口以及图片显示、保存的函数。



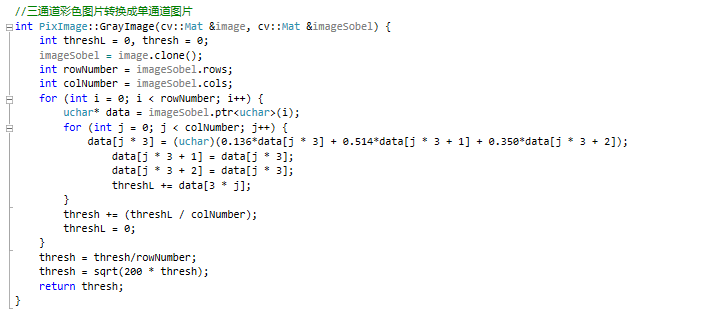
利用.ptr<>函数访问并得到图片的像素信息。

对图像中的像素进行量化处理。如常见的RGB24图像有256\*256\*256中颜色，通过Reduce Color将每个通道的像素减少8倍至256/8=32种，则图像只有32\*32\*32种颜色。假设量化减少的倍数是N，则代码实现时就是简单的value/N\*N，通常会再加上N/2以得到相邻的N的倍数的中间值，最后图像被量化为(256/N)×(256/N)×(256/N)种颜色。





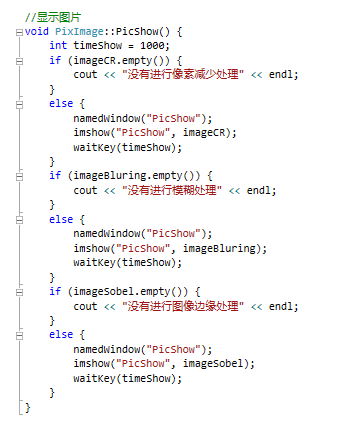
利用均值滤波器消除噪声使图片模糊，即将任意一个像素，取其周围n\*n个像素范围内的平均值来置换该像素；为了便于操作，将图片中间的像素与边缘像素分开处理。



灰度处理，即将多通道图片转换成单通道图片。将RGB像素值按照比例相加，并且使RGB三个通道的值相等即完成转换；并且求得该幅图片所有点灰度的平均值，即Sobel阈值。经多次试验比较，得出scale在200左右比较合适。



利用索贝尔算子检测图片边缘。将图像与两组矩阵分别作平面卷积，得出横向及纵向的亮度差分近似值。若某点灰度值大于Sobel阈值，则将其赋值为255；若该点灰度值小于等于Sobel阈值，则将其赋值为0。





图片处理和保存函数，若没有进行相应的处理则输出提示切不保存。

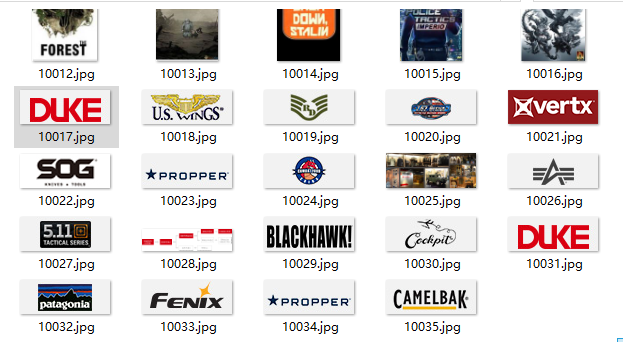


图片处理部分主函数，保存图片至相应的路径并加后缀命名指出进行了何种处理。

1. **运行结果**
2. 爬虫部分：



爬取了1013个不重复的网址以及10035张图片。



1. 图片处理部分



像素减少处理 索贝尔算子边缘检测处理



模糊处理 原图