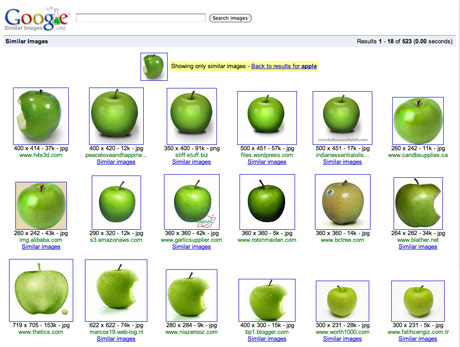
**相似图片搜索的原理（二）**

作者： [阮一峰](http://www.ruanyifeng.com/)

日期： [2013年3月31日](http://www.ruanyifeng.com/blog/2013/03/)

二年前，我写了[《相似图片搜索的原理》](http://www.ruanyifeng.com/blog/2011/07/principle_of_similar_image_search.html)，介绍了一种最简单的实现方法。

昨天，我在[isnowfy](http://www.isnowfy.com/similar-image-search/)的网站看到，还有其他两种方法也很简单，这里做一些笔记。



**一、颜色分布法**

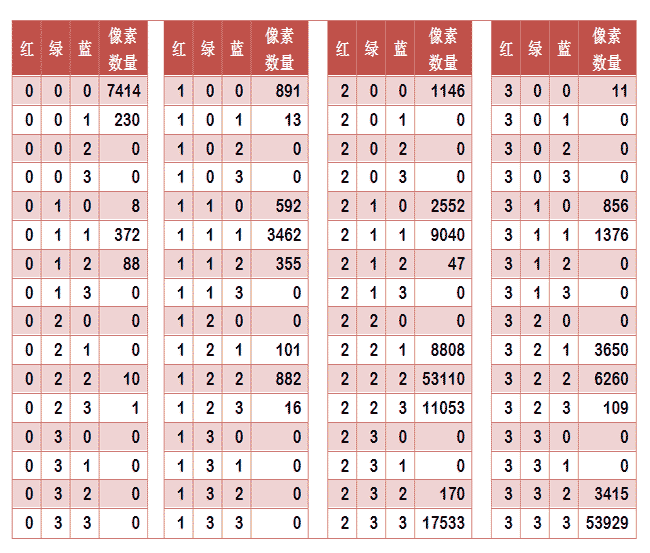
每张图片都可以生成[颜色分布的直方图](http://en.wikipedia.org/wiki/Color_histogram)（color histogram）。如果两张图片的直方图很接近，就可以认为它们很相似。



任何一种颜色都是由红绿蓝三原色（RGB）构成的，所以上图共有4张直方图（三原色直方图 + 最后合成的直方图）。

如果每种原色都可以取256个值，那么整个颜色空间共有1600万种颜色（256的三次方）。针对这1600万种颜色比较直方图，计算量实在太大了，因此需要采用简化方法。可以将0～255分成四个区：0～63为第0区，64～127为第1区，128～191为第2区，192～255为第3区。这意味着红绿蓝分别有4个区，总共可以构成64种组合（4的3次方）。

任何一种颜色必然属于这64种组合中的一种，这样就可以统计每一种组合包含的像素数量。



上图是某张图片的颜色分布表，将表中最后一栏提取出来，组成一个64维向量(7414, 230, 0, 0, 8, ..., 109, 0, 0, 3415, 53929)。这个向量就是这张图片的特征值或者叫"指纹"。

于是，寻找相似图片就变成了找出与其最相似的向量。这可以用[皮尔逊相关系数](http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient)或者[余弦相似度](http://www.ruanyifeng.com/blog/2013/03/cosine_similarity.html)算出。

**二、内容特征法**

除了颜色构成，还可以从比较图片内容的相似性入手。

首先，将原图转成一张较小的灰度图片，假定为50x50像素。然后，确定一个阈值，将灰度图片转成黑白图片。

如果两张图片很相似，它们的黑白轮廓应该是相近的。于是，问题就变成了，第一步如何确定一个合理的阈值，正确呈现照片中的轮廓？

显然，前景色与背景色反差越大，轮廓就越明显。这意味着，如果我们找到一个值，可以使得前景色和背景色各自的"类内差异最小"（minimizing the intra-class variance），或者"类间差异最大"（maximizing the inter-class variance），那么这个值就是理想的阈值。

1979年，日本学者大津展之证明了，"类内差异最小"与"类间差异最大"是同一件事，即对应同一个阈值。他提出一种简单的算法，可以求出这个阈值，这被称为["大津法"](http://en.wikipedia.org/wiki/Otsu's_method)（Otsu's method）。下面就是他的计算方法。

假定一张图片共有n个像素，其中灰度值小于阈值的像素为 n1 个，大于等于阈值的像素为 n2 个（ n1 + n2 = n ）。w1 和 w2 表示这两种像素各自的比重。

　　w1 = n1 / n

　　w2 = n2 / n

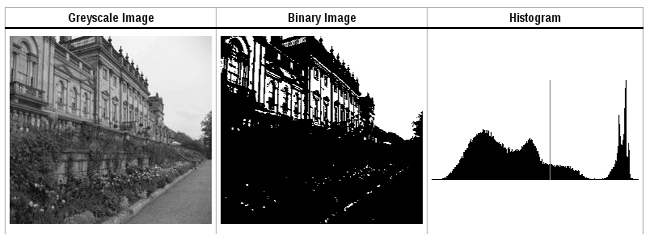
再假定，所有灰度值小于阈值的像素的平均值和方差分别为 μ1 和 σ1，所有灰度值大于等于阈值的像素的平均值和方差分别为 μ2 和 σ2。于是，可以得到

　　类内差异 = w1(σ1的平方) + w2(σ2的平方)

　　类间差异 = w1w2(μ1-μ2)^2

可以证明，这两个式子是等价的：得到"类内差异"的最小值，等同于得到"类间差异"的最大值。不过，从计算难度看，后者的计算要容易一些。

下一步用"穷举法"，将阈值从灰度的最低值到最高值，依次取一遍，分别代入上面的算式。使得"类内差异最小"或"类间差异最大"的那个值，就是最终的阈值。具体的实例和Java算法，请看[这里](http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html)。



有了50x50像素的黑白缩略图，就等于有了一个50x50的0-1矩阵。矩阵的每个值对应原图的一个像素，0表示黑色，1表示白色。这个矩阵就是一张图片的特征矩阵。

两个特征矩阵的不同之处越少，就代表两张图片越相似。这可以用"异或运算"实现（即两个值之中只有一个为1，则运算结果为1，否则运算结果为0）。对不同图片的特征矩阵进行"异或运算"，结果中的1越少，就是越相似的图片。

（完）