RGB 图像灰度化

作者：armstrong
原文链接：https://ld246.com/article/1359968457263
来源网站：链滴

     灰度化在图像处理中很常见。 生产应用中普遍适用公式 Gray = *0.299 + G*0.587 + B*0.114。关于运行效率与精度的取舍，请参考http://bbs.ednchina.com/BLO_ARTICLE_1999487.HTM。
      下面使用前文参考文章中提到的Adobe RGB (1998) [gamm=2.20] 公式： Gray = (R^2.2 * 0.2973 + G^2.2 * 0.6274 + B^.2 * 0.0753)^(1/2.2)。
      Java实现如下：

<pre>   // (R^2.2 * 0.2973 + G^2.2 * 0.6274 + B^2.2 * 0.0753)^(1/2.2)
    // Adobe RGB (1998) [gamma=2.20]
    // http://bbs.ednchina.com/BLOG_ARTICLE_1999487.HTM
    public BufferedImage grarify(final BufferedImage image) {
        final int width = image.getWidth();
        final int height = image.getHeight();
        final BufferedImage grayImage = new BufferedImage(width, height,
                BufferedImage.TYPE_BYTE_GRAY);
        for (int i = 0; i < width; i++) {
            for (int j = 0; j < height; j++) {
                final int color = image.getRGB(i, j);
                final int r = (color >> 16) & 0xff;
                final int g = (color >> 8) & 0xff;
                final int b = color & 0xff;
                final int gray = (int) Math
                        .pow((Math.pow(r, 2.2) * 0.2973 + Math.pow(g, 2.2)
                                * 0.6274 + Math.pow(b, 2.2) * 0.0753),
                                (1 / 2.2));
                grayImage.setRGB(i, j, gray);
            }
        }
        return grayImage;
    }</pre>

     附：参考文章《RGB转灰度图的几种算法》 http://bbs.ednchina.com/LOG_ARTICLE_1999487.HTM

 方法一：
     对于彩色转灰度，有一个很著名的心理学公式：
             nbsp;   ray = R*0.299 + G*0.587 + B*0.114
      方法二：
      而实际应用时，希望避免低速的浮点运算，所以需要整数算法。
      注意到系数都是3位精度的没有，我们可以将它们缩放1000倍来实现整数运算法：
             nbsp;   ray = (R*299 + G*587 + B*114 + 500) / 1000
      RGB一般是8位精度，现在缩放1000倍，所以上面的运算是32位整型运算。注意后面那个除法是整数 除法，所以需要加上500来实现四舍五入。
    &

bsp; 就是由于该算法需要32位运算，所以该公式的另一个变种很流行： </p>
<p>
nbsp;
ray = (R*30 + G*59 + B*11 + 50) / 100 </p>
<p>   方法三： </p>
<p> 上面的整数算法已经很快了，但是有一点仍制约速度，就是最后的那个除法。移位比除法快多
，所以可以将系数缩放成 2的整数幂。
 习惯上使用16位精度，2的16次幂是65536，所以这样
算系数： </p>
<p>
nbsp;   0
299 * 65536 = 19595.264 ≈ 19595
nbsp;
nbsp;   0.587 * 65536 + (0.264) = 38469.632 + 0.264 = 38469.896
 38469

  0.114 * 65536 + (0.896) =   7471.104 + 0.896 = 7472 </p>
<p> 可能很多人看见了，我所使用的舍入方式不是四舍五入。四舍五入会有较大的误差，应该将以
的计算结果的误差一起计算进去，舍入方式是去尾法： </p>
<p> 写成表达式是： </p>
<p>
nbsp;
ray = (R*19595 + G*38469 + B*7472) >> 16 </p>
<p> 2至20位精度的系数： </p>
<p>
nbsp;
ray = (R*1 + G*2 + B*1) >> 2
nbsp;
nbsp;   Gray = (R*2 + G*5 + B*1) >> 3

           Gray = (R*4 + G*10
 B*2) >> 4
          &nb
p;   &nb
p;   Gray = (R*9 + G*19 + B*4) >> 5
     &
bsp;   &
bsp;   Gray = (R*19 + G*37 + B*8) >>

Gray = (R*38 + G*75 + B*15) >> 7
       &
bsp;   &
bsp;   Gray = (R*76 + G*150 + B*30) >> 8
 &n
sp;   &n
sp;   Gray = (R*
53 + G*300 + B*59) >> 9
        &nb
p;   &nb
p;   Gray = (R*306 + G*601 + B*117) >> 10
  &
bsp;   &
bsp;   Gray = (R*612 +
G*1202 + B*234) >> 11

     Gray = (R*1224 + G*2405 + B*467) >> 12
  &
bsp;   &
bsp;   Gray = (R*2449
 G*4809 + B*934) >> 13
        &nbs
;

     Gray = (R*4898 + G*9618 + B*1868) >> 14
nbsp;
nbsp;   Gray = (R*9797
+ G*19235 + B*3736) >> 15
nbsp;
nbsp;   Gray = (R*19595 + G*38469 + B*7472) >> 16

             Gray
(R*39190 + G*76939 + B*14943) >> 17
      &nb
p;   &nb
p;   Gray = (R*78381 + G*153878 + B*29885) >
gt; 18
nbsp;
nbsp; Gray = (R*156762 + G*307757 + B*59769) >> 19
    &n
sp;   &n
sp;   Gray = (R*313524 + G*615514 +
B*119538) >> 20 </p>
<p> 仔细观察上面的表格，这些精度实际上是一样的：3与4、7与8、10与11、13与14、19与20
 所以16位运算下最好的计算公式是使用7位精度，比先前那个系数缩放100倍的精度高，而且速度
： </p>
<p>
nbsp;
ray = (R*38 + G*75 + B*15) >> 7 </p>
<p> 其实最有意思的还是那个2位精度的，完全可以移位优化： </p>
<p>
nbsp;
ray = (R + (WORD)G<<1 + B) >> 2 </p>
<p>   另一种是 Adobe Photoshop 里的公式&
bsp;
    Adobe RGB (1998) [gamma=2.20]&nb
p;
    Gray = (R^2.2 * 0.2973 + G^2.2 * 0.6274 + B^2.2 * 0
0753)^(1/2.2) </p>
<p> 该方法运行速度稍慢，但是效果很好。 </p>
<p>   还有就是   平均值方法 </stro
g> </p>
<p>   GRAY = (RED+BLUE+GREEN)/3 </p>
<p> <strong>   （GRAY,GRAY,GRAY ） 替代 （RED,GREEN,BLUE）</stro
g> </p>
<p></p>

原文链接：RGB 图像灰度化