Java实现图像灰度化

24 May 2014

概要

本文主要介绍了灰度化的几种方法，以及如何使用Java实现灰度化。同时分析了网上一种常见却并不妥当的Java灰度化实现，以及证明了opencv的灰度化是使用“加权灰度化”法

24位彩色图与8位灰度图

首先要先介绍一下24位彩色图像，在一个24位彩色图像中，每个像素由三个字节表示，通常表示为RGB。通常，许多24位彩色图像存储为32位图像，每个像素多余的字节存储为一个alpha值，表现有特殊影响的信息[1]。

在RGB模型中，如果R=G=B时，则彩色表示一种灰度颜色，其中R=G=B的值叫灰度值，因此，灰度图像每个像素只需一个字节存放灰度值（又称强度值、亮度值），灰度范围为0-255[2]。这样就得到一幅图片的灰度图。

几种灰度化的方法

- 分量法：使用RGB三个分量中的一个作为灰度图的灰度值。
- 最值法：使用RGB三个分量中最大值或最小值作为灰度图的灰度值。
- 均值法：使用RGB三个分量的平均值作为灰度图的灰度值。
- 加权法：由于人眼颜色敏感度不同，按下一定的权值对RGB三分量进行加权平均能得到较合理的灰度图像。一般情况按照：`Y = 0.30R + 0.59G + 0.11B`。

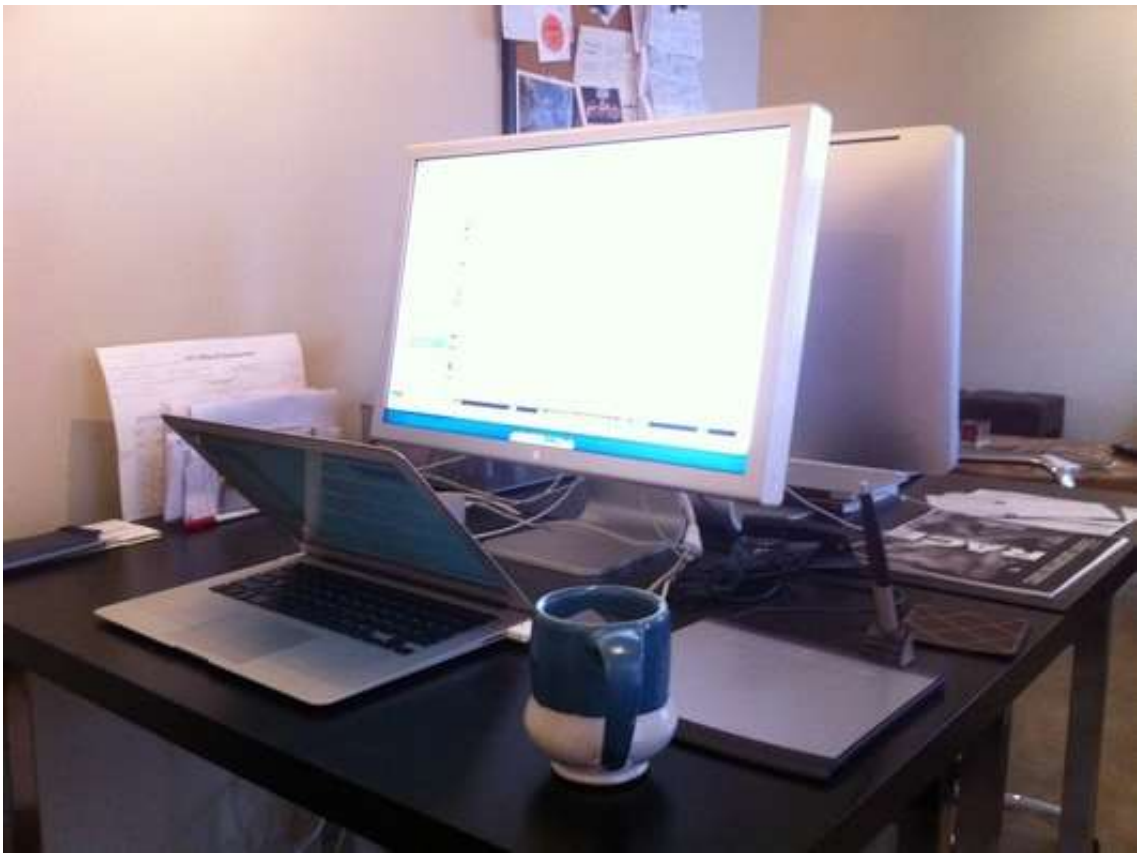
[注]加权法实际上是取一幅图片的亮度值作为灰度值来计算，用到了YUV模型。在[3]中会发现作者使用了`Y = 0.21 * r + 0.71 * g + 0.07 * b`来计算灰度值（显然三个权值相加并不等于1，可能是作者的错误？）。实际上，这种差别应该与是否使用伽马校正有关[1]。

一种Java实现灰度化的方法

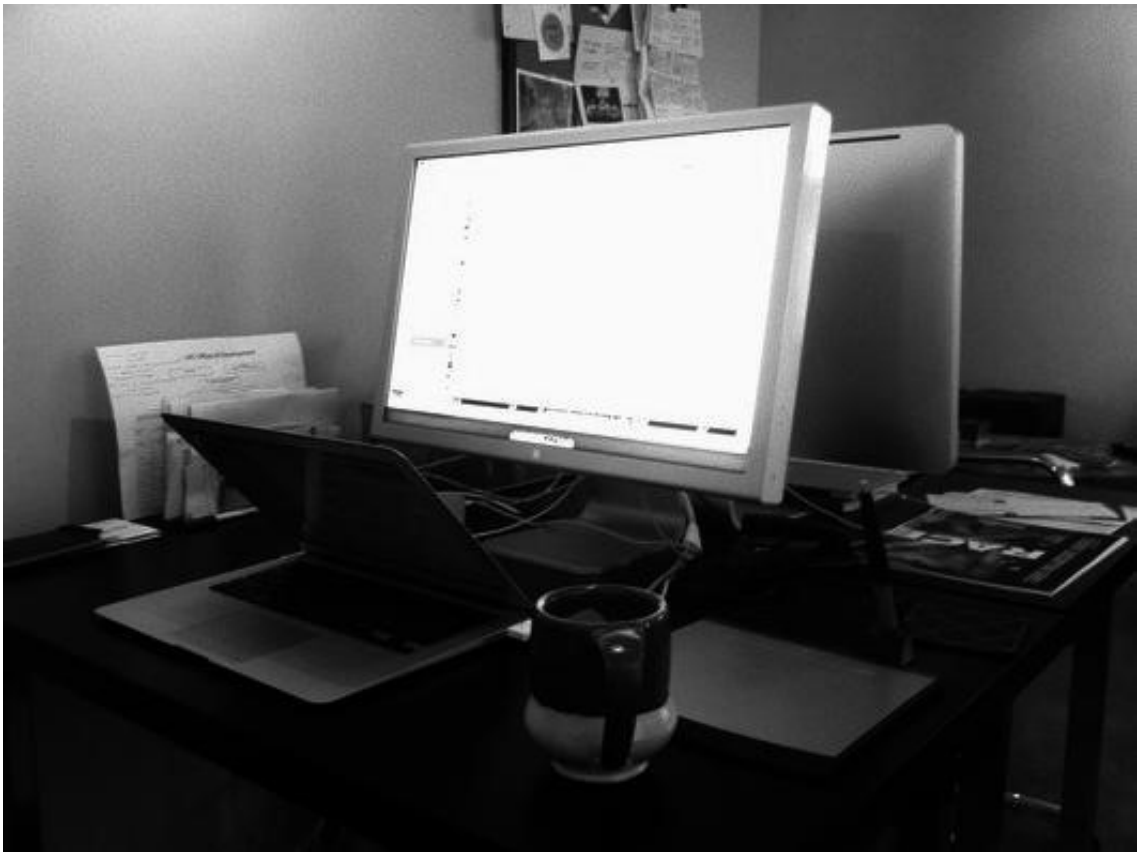
如果你搜索“Java实现灰度化”，十有八九都是一种方法（代码）：

```
public void grayImage() throws IOException{
    File file = new File(System.getProperty("user.dir")+"/test.jpg");
    BufferedImage image = ImageIO.read(file);

    int width = image.getWidth();
    int height = image.getHeight();

    BufferedImage grayImage = new BufferedImage(width, height, BufferedImage.TYPE_BYTE_GRA
Y);
    for(int i= 0 ; i < width ; i++){
        for(int j = 0 ; j < height; j++){
        int rgb = image.getRGB(i, j);
        grayImage.setRGB(i, j, rgb);
        }
    }

    File newFile = new File(System.getProperty("user.dir")+"/method1.jpg");
    ImageIO.write(grayImage, "jpg", newFile);
}
```

test.jpg的原图为：

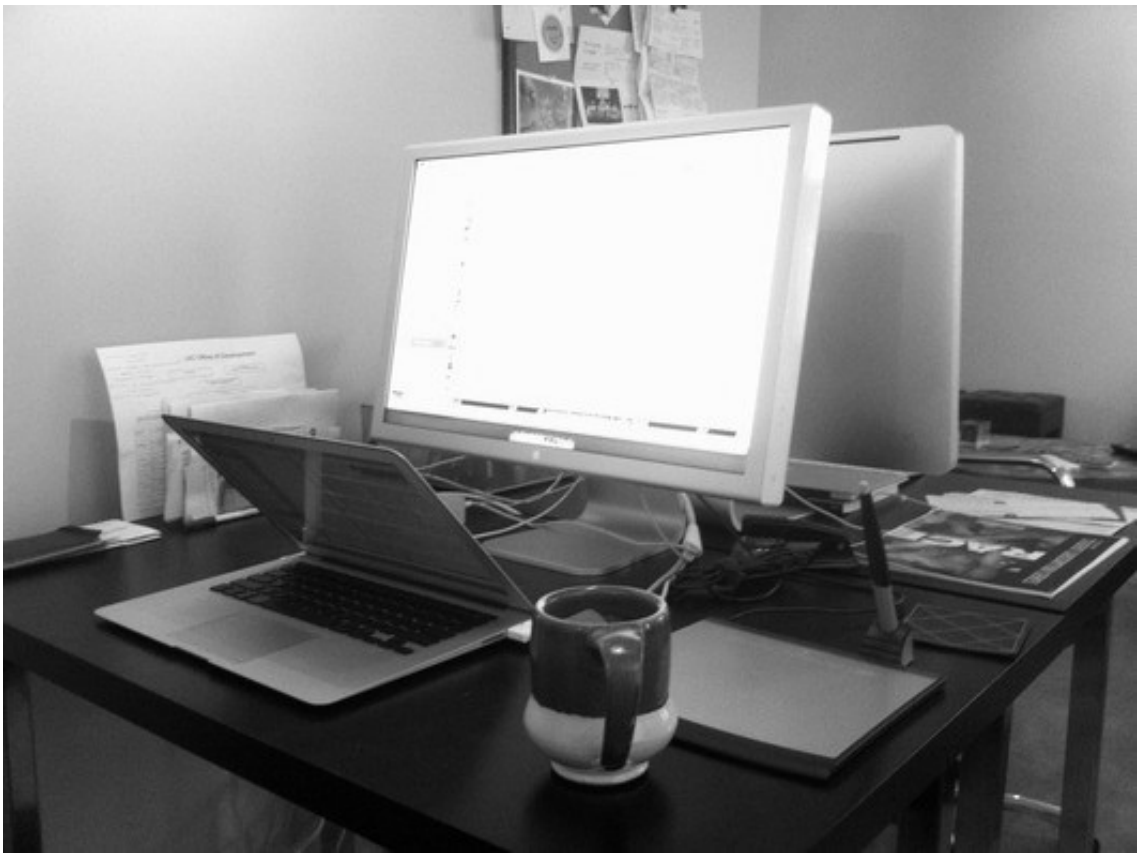


使用上述方法得到的灰度图：



看到这幅灰度图，似乎还真是可行，但是如果我们使用opencv来实现灰度化或使用PIL（Python），你会发现效果相差很大：

```
img = cv2.imread('test.jpg',cv2.IMREAD_COLOR)
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
cv2.imwrite('PythonMethod.jpg', gray)
```



可以清楚的看到，使用opencv（PIL也是一样的）得到的灰度图要比上面Java方法得到的方法好很多，很多细节都能够看得到。这说明，网上这种流行的方法一直都存在这某种问题，只是一直被忽略。

opencv如何实现灰度化

如果读过opencv相关的书籍或代码，大概都能知道opencv灰度化使用的是加权法，之所以说是大概，因为我们不知道为什么opencv灰度化的图像如此的好，是否有其他的处理细节被我们忽略了？

验证我们的猜想很简单，只要查看像素值灰度化前后的变化就知道了，可以如下测试：

```
img = cv2.imread('test.jpg',cv2.IMREAD_COLOR)
h, w = img.shape[:2]
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
for j in range(w):
        for i in range(h):
                print str(i) + " : " + str(j) + " " + str(gray[i][j])
print img[h-1][w-1][0:3]
```

以下打印了这么多像素点，我们也很难判断，但是我们只要关注一下最后一个像素点，就能够发现端倪： 原图最后的像素点RGB值为44，67，89，而灰度化之后的值为71。正好符合加权法计算的灰度值。如果你检查之前用Java灰度化的图片的像素值，你会发现不仅仅像素值不符合这个公式，甚至相差甚远。

到此，我们猜测opencv（也包括PIL）是使用加权法实现的灰度化。

Java实现加权法灰度化

如果网上那段流行的方法不行，我们该如何使用Java实现灰度化？实际上[3]已经成功的实现了（多种方法的）灰度化（外国友人搞技术还是很给力的），在此仅仅提取必要的代码：

```
private static int colorToRGB(int alpha, int red, int green, int blue) {

                int newPixel = 0;
                newPixel += alpha;
                newPixel = newPixel << 8;
                newPixel += red;
                newPixel = newPixel << 8;
                newPixel += green;
                newPixel = newPixel << 8;
                newPixel += blue;

                return newPixel;

}
public static void main(String[] args) throws IOException {
        BufferedImage bufferedImage
                = ImageIO.read(new File(System.getProperty("user.dir" + "/test.jpg"));
        BufferedImage grayImage =
                new BufferedImage(bufferedImage.getWidth(),
                                                bufferedImage.getHeight(),
                                                bufferedImage.getType());


        for (int i = 0; i < bufferedImage.getWidth(); i++) {
                for (int j = 0; j < bufferedImage.getHeight(); j++) {
                        final int color = bufferedImage.getRGB(i, j);
                        final int r = (color >> 16) & 0xff;
                        final int g = (color >> 8) & 0xff;
                        final int b = color & 0xff;
                        int gray = (int) (0.3 * r + 0.59 * g + 0.11 * b);;
                        System.out.println(i + " : " + j + " " + gray);
                        int newPixel = colorToRGB(255, gray, gray, gray);
                        grayImage.setRGB(i, j, newPixel);
                }
        }
        File newFile = new File(System.getProperty("user.dir") + "/ok.jpg");
        ImageIO.write(grayImage, "jpg", newFile);
}
```

上面的代码会打印出灰度化后的像素值，如果再与上面的Python代码做对比，你会发现像素值完全的对应上了。colorToRGB方法中对彩色图的处理正好是4个字节，其中之一是alpha参数（前文所讲），下图是这段代码灰度化后的图像：

对于其他方法，依次同理可得。

总结

本文的成因本是希望使用Java实现几种灰度化操作，并使用opencv来验证转化的对错，但在实际测试中发现了一些问题（转化后的图片有差异，以及如何在转化后根据灰度值生成灰度图等问题），并就此进行了一定的思考与验证。

这里需要注意的是，网上的一些文章或多或少没有做更进一步的思考（甚至很多都是照搬，尤其是国内的文章），而对于这些实际问题，动手实现并验证是非常重要的方法。希望本文对大家有所帮助。

参考

- [1] 《多媒体技术教程》Ze-Nian Li，Mark S.Drew著，机械工业出版社。
- [2] 百度百科：灰度值 (http://baike.baidu.com/view/3701940.htm)
- [3] Java color image to grayscale conversion algorithm(s) (http://zerocool.is-a-geek.net/java-color-image-to-grayscale-conversion-algorithm/)

java 19 (/categories.html#java-ref)

Java 19 (/tags.html#Java-ref) 图像处理 1 (/tags.html#图像处理-ref)

← Previous (/life/2014/04/24/recent-summary) Archive (/archive.html)

Next → (/java/2014/05/27/java-node)