[百度空间](http://hi.baidu.com/index.htm)

* [个人中心](http://hi.baidu.com/leeihcy_hi/ihome)
* [我的主页](http://hi.baidu.com/leeihcy_hi/home)
* [好友](http://hi.baidu.com/leeihcy_hi/ihome/frdme)
* [消息](http://hi.baidu.com/reallty/blog/item/msg/index) [http://img.baidu.com/hi/img/email_s.gif?v=43198120.gif](http://hi.baidu.com/reallty/blog/item/40a49b02d5b8bb024bfb511c.html)

站内消息 ：无新消息提醒

互动请求 ：无新消息提醒

系统通知 ：1条新

leeihcy

|

[装扮](http://hi.baidu.com/leeihcy_hi/modify/spmodules/0)

|

[设置](http://hi.baidu.com/leeihcy_hi/ihome/basicconfig)

|

[退出](https://passport.baidu.com/?logout&bdstoken=23f28e3cd4503b0a76cc392bd4bb035a&u=)

[I'm a lazy BOY](http://hi.baidu.com/reallty)

I'm a lazy BOY---COME BACK!!!

[主页](http://hi.baidu.com/reallty/home)[博客](http://hi.baidu.com/reallty/blog)[相册](http://hi.baidu.com/reallty/album)|[个人档案](http://hi.baidu.com/reallty/profile)|[好友](http://hi.baidu.com/reallty/friends)|[i贴吧](http://tieba.baidu.com/i/sys/jump?un=reallty&fr=spacebar)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 查看文章 |  |  |

【转】GDI+ ColorMatrix的完全揭秘

2010-06-12 9:46

转载自 [wwwanq](http://hi.baidu.com/wwwanq)

最终编辑 [reallty](http://hi.baidu.com/reallty)

|  |  |
| --- | --- |
| |  | | --- | | 一篇好文章，引用自：  http://blog.csdn.net/maozefa/archive/2008/09/08/2896752.aspx      无论是用何种语言，只要使用过Windows的GDI+的人对ColorMatrix都不陌生，我的BLOG文章中也多次提到过，并在《GDI+ for VCL基础 -- 颜色调整矩阵ColorMatrix详解》一文中对其功能作了较为详细的讲解，虽然自认对ColorMatrix使用已经相当熟练，但对其原理也是知其然而不知其所以然。直到前几天有位朋友就ColorMatrix实现图像去反功能不正常而问我（见http://topic.csdn.net/u/20080830/20/070c83de-d45b-441f-996e-3c68892855cd.html）， 我认为不大可能！众所周知，用RGB主对角线-1矩阵实现图像求反是ColorMatrix重要功能之一，可是我试验了多次，那张图片确实不能正常去反。 我这人对自己自认熟悉的东西产生怀疑后，就有种不搞明白不罢休的冲动，为此，本人对ColorMatrix的原理作了透彻的解剖，不仅搞清楚了图像去反不 正常的原因，还用代码完整地实现了ColorMatrix功能！      准备用3篇文章的内容来与大家分享研究心得，上篇主要介绍ColorMatrix原理揭秘；中篇讲述ColorMatrix的代码实现以及GDI+应用举 例，下篇专门介绍不使用GDI+，而用Delphi和BCB的TBitmap甚至TGraphic实现ColorMatrix的全部功能（其实C++用户 和VB用户也可作借鉴）。      一、ColorMatrix原理揭秘      虽然我的BLOG中已有2篇文章引用了那段经典的关于ColorMatrix功能介绍的文字，但这里还是再引用一次，作为原理揭秘的开始：          GDI+ 提供用于存储和操作图像的 Image 和 Bitmap 类。Image 和 Bitmap 对象将每个像素的颜色都存储为 32 位的数：红色、绿色、蓝色和 alpha 各占 8 位。这四个分量的值都是 0 到 255，其中 0 表示没有亮度，255 表示最大亮度。alpha 分量指定颜色的透明度：0 表示完全透明，255 表示完全不透明。         颜色矢量采用 4 元组形式（红色、绿色、蓝色、alpha）。例如，颜色矢量 (0, 255, 0, 255) 表示一种没有红色和蓝色但绿色达到最大亮度的不透明颜色。         表示颜色的另一种惯例是用数字 1 表示亮度达到最大。使用这种惯例，上一段中描述的颜色将用 (0, 1, 0, 1) 表示。GDI+ 在进行颜色变换时使用以 1 表示最大亮度的惯例。         可通过用 4×4 矩阵乘以这些颜色矢量将线性变换（旋转和缩放等）应用到颜色矢量中。但是，您不能使用 4×4 矩阵进行平移（非线性）。如果在每个颜色矢量中再添加一个虚拟的第 5 坐标（例如，数字 1），则可使用 5×5 矩阵应用任何组合形式的线性变换和平移。由线性变换组成的后跟平移的变换称为仿射变换。      说实话，不仅是刚开始接触ColorMatrix的时候，就是现在的我看到这段文字，都觉得有点玄，前3段很好理解，特别是后面那段，更是使那些初识 GDI+的人觉得ColorMatrix深奥无比：ColorMatrix是怎样实现颜色的缩放、旋转、剪切及平移的？靠这些功能能实现图像的哪些效果？ 或者说，某种效果能用ColorMatrix实现吗？     下面，就让我们一步步揭开ColorMatrix的神秘面纱，我相信，当你看完这篇文章后，一定会说：哦，原来如此，这么简单！其实，这就是所谓“江湖一 张纸，戳破不值一分钱”！很多看起来技术含量很高的东西，被人千方百计保密，申请专利，搞的神秘无比，其实一旦公开，就那么回事！  http://hiphotos.baidu.com/reallty/pic/item/0c338744f10e2172500ffe50.jpg  http://hiphotos.baidu.com/reallty/pic/item/ebf81a4c4df9e2c7d62afc50.jpg      要揭秘ColorMatrix，就要解析它的缩放、旋转、剪切及平移功能是怎么实现的，为了方便叙述，我画了个ColorMatrix矩阵图贴在上面，在下面的表述中，大写ARGB表示颜色各分量现有的值，而小写argb表示运算后得到的新值。      1、颜色缩放：颜色缩放很简单，就是按照给定的比例值，在图像像素现有A、R、G、B各分量数值基础上计算出新的分量值。这个比例值就是 ColorMatrix主对角线除m55外的其它4个值。比如某像素的RGBA值现在分别为255、128、64和255，而主对角线m11 - m44的值分别为0.8、0.5、-1及0.5，那么该像素新的rgba值应该是：          r = R \* m11 = 255 \* 0.8 = 204;          g = G \* m22 = 128 \* 0.5 = 64;          b = B \* m33 = 64 \* -1 = 192;          a = A \* m44 = 255 \* 0.5 = 128;  是否很简单？！可能有初学者说我上面b值计算错了，64 \* -1应该等于-64。没错，64用32位数表示为0xFFFFFFC0，无符号字节饱和取整，取最后8位0xC0,等于192。关于负值运算的问题，后面还要详细讲到的。      2、颜色剪切：一般说来，图像像素R、G、B各分量按照与另一种颜色分量成比例的量来增加或减少颜色分量就是剪切。其实这种表述并不完全，像素的A分量也是参与其中的！      以红色分量R举例，如果要按绿色分量G进行剪切，那么m21就是剪切比例值，m21 \* G就得到了G对R的剪切量。同理，m31 \* B、m41 \* A可分别得到B和A对R的剪切量，将这些剪切量加起来，就是R总的剪切量。用公式表示为：         r = G \* m21 + B \* m31 + A \* m41;          g = R \* m12 + B \* m32 + A \* m42;          b = R \* m13 + G \* m23 + A \* m43;           a = R \* m14 + G \* m24 + B \* m34;      3、颜色旋转：颜色旋转的描述比较复杂，就是在图像像素中，用其中的2个分量按照一定的角度围绕另外1个分量作运算的结果，就是颜色的旋转。以红色分量R和绿色分量G围绕蓝色分量G旋转60度为例：          m11 = cos(60) = 0.5, m12 = sin(60) = 0.866,   m21 = -sin(60) = -0.866, m22 = cos(60) = 0.5，那么，R和G 所得到的旋转量分别为：         r = R \* m11(0.5) + G \* m21(-0.866);         g = R \* m12(0.866) + G \* m22(0.5);  从上面的公式看，所谓的颜色旋转量，其实就是旋转的2个分量自身的缩放量加上与对方的剪切量而已！就运算角度看，同其它分量没有任何关系。      4、颜色平移：上面的缩放、剪切和旋转属于颜色的线性变换（都是乘法运算的累积和），而平移是颜色的非线性变换，就是对颜色各分量做一个加法而已：图像像 素各分量的平移量用所谓的虚拟位，即第5行的各个值来表示，各分量加上所在列的虚拟行的值就是颜色平移，其实质就是非线性地调整了该分量的亮度值。用公式 表示各分量的平移量：          r = R + m51 \* 255;          g = G + m52 \* 255;          b = B + m53 \* 255;          a = A + m54 \* 255;      综合以上颜色的缩放、旋转、剪切及平移公式，对于颜色的每个分量R、G、B、A来说，运用ColorMatrix后所得到的实际值r、g、b、a，用公式表示为：         r = R \* m11 + G \* m21 + B \* m31 + A \* m41 + m51 \* 255;          g = R \* m12 + G \* m22 + B \* m32 + A \* m42 + m52 \* 255;          b = R \* m13 + G \* m23 + B \* m33 + A \* m43 + m53 \* 255;           a = R \* m14 + G \* m24 + B \* m34 + A \* m44 + m54 \* 255;  从技术层面上讲，这个公式的含义就是颜色每个分量的新值，等于这个分量在ColorMatrix中对应列的前4行的值与R、G、A、B当前值的乘积之和加上第5行的值与常数255的乘积，而虚拟列（第5列）不起任何作用。      关于ColorMatrix功能的实现原理就这些，最后说说前面所提到的负值运算问题：ColorMatrix的负值运算无论是从原理上还是纯技术实现上都很有些麻烦：      a)、在前面缩放举例中，也就是只有主对角线有值的情况下，b的负值运算为 b = B \* m33 = 64 \* -1 = 192， 这是正确的，但是，如果加上一个剪切量或者平移量，这个值就发生了一些变化。比如在缩放基础上加上0.1的平移量，这似乎是很简单的问题，直接用0.1 \* 255加上去不就得了！可是，这个看是简单的问题，却有2个方案：是在缩放后的192基础上加平移量0.1 \* 255等于218？还是应该做连续运算： b = B \* m33 = 64 \* -1 + 255 \* 0.1= -38（饱和处理后为0）呢？ 如果是前者，运算法则上似乎说不过去；如果是后者，感官上难以接受，本来b=192使颜色值“蓝蓝”的，本想让它再“蓝”一点，而加个平移（亮度）量，可 即使这个平移量相当小，例如只平移0.001，却让b值变成了0，不仅没有变的更“蓝”，反而使颜色失去了蓝色分量！可ColorMatrix偏偏用了使 你感官难以接受的方案。其实也难怪设计者，只要顺其自然去实现代码，最后自然就成了这个结果。能够造成这个结果的，不仅仅是平移量，只要矩阵中主对角线以 外的任何一个值不为0，包括不起作用的第5列，都会使主对角线中，具有负值所对应的RGBA分量发生这种变化。具体请看本文中篇的代码实现。      b)、前面也提到了有朋友用-1矩阵对图像求反不合要求的问题。通过上面的缩放公式计算一下，不难发现，用-1矩阵所得到的图像取反图只是近似的！为什么 这么说呢？所谓取反，就是可逆，如果用255减去R（GB）取反后，再用255去减就可还原；用255异和R（GB），不用说也是可逆的，而-1 \* R（GB）却是不完全可逆的：1 \* -1饱和取整后为255，2 \* -1为254 ......等等；反过来还原：结果是255 \* -1 = 1、254 \* -1 = 2......等等，从1 - 255都是可还原的，虽然这个还原于原本意义上的去反还原有1的误差，但从人的感官上，这点误差还是能接受的，只有“0”这个值是不可逆的，：0 \* -1饱和取整还是0，无论是取反，还是还原，“0”依然是“0”，从而破坏了取反图的协调，使人的感官完全无法接受，例如原来纯红色的像素、取反后应该为 青色，可是这里却变成了黑色，你能接受吗？所以只要图像像素RGB任何一个分量为0，那么这幅图的取反效果就会出问题！有人可能说，我们都是这么做的，除 了极少数图片，大部分取反图看上去还是正常的，难道这些正常图里面就没有RGB分量为0的像素？其实，一幅图中间歇的分布几个RGB分量为0的像素，取反 后，看上去是不那么明显的，所以我们认为这是成功的，只有那些“0”分量集中的部分，才严重破坏了图的协调。      关于ColorMatrix原理上的揭秘就到此，由于我的文化理论水平有限，只能从技术角度来揭秘ColorMatrix的实现原理，说得不好，请原谅！      通过上面对ColorMatrix详尽的技术解剖，我们推出了它的运算公式，有了这些，用程序代码实现ColorMatrix功能，应该很容易了，本文中篇将介绍ColorMatrix功能的具体实现，相信各位高手应该比我更在行。      如有不同看法和疑问，可写在文章评论中，本人将尽可能予以解答。欢迎指导和建议，来信邮箱：maozefa@hotmail.com | |

[类别：默认分类](http://hi.baidu.com/reallty/blog/category/%C4%AC%C8%CF%B7%D6%C0%E0)|[http://img.baidu.com/hi/apps/share/share_btn.gif](http://hi.baidu.com/reallty/blog/item/40a49b02d5b8bb024bfb511c.html)|[添加到搜藏](http://cang.baidu.com/do/add) |[分享到i贴吧](http://hi.baidu.com/reallty/blog/item/40a49b02d5b8bb024bfb511c.html)|浏览(124)|[评论](http://hi.baidu.com/reallty/blog/item/40a49b02d5b8bb024bfb511c.html#send) (0)

上一篇：[亮度，饱和度调节等效果](http://hi.baidu.com/reallty/blog/item/9e1fc813fee39d0f5aaf5310.html)    下一篇：[C++ Memory Management Innovati...](http://hi.baidu.com/reallty/blog/item/fe456b8b210fa61fc8fc7ae4.html)

最近读者：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [http://tx.bdimg.com/sys/portraitn/item/c9d0696e7669736961626c65b602.jpg](http://hi.baidu.com/invisiable) | [http://tx.bdimg.com/sys/portraitn/item/5284b4f3c4aeb9c2caa83101.jpg](http://hi.baidu.com/大漠孤狮) | [http://tx.bdimg.com/sys/portraitn/item/daeb7a7a686f753030319617.jpg](http://hi.baidu.com/zzhou001) | [http://tx.bdimg.com/sys/portraitn/item/090c6b766f6f76613010.jpg](http://hi.baidu.com/kvoova) | [http://tx.bdimg.com/sys/portraitn/item/e87663726169676769617263e603.jpg](http://hi.baidu.com/craiggiarc) | [http://tx.bdimg.com/sys/portraitn/item/dc4bcff0c6a4caf7b5c4b2a9bfcdd508.jpg](http://hi.baidu.com/橡皮树的博客) | [http://tx.bdimg.com/sys/portraitn/item/e569666c79736b795f323030359f02.jpg](http://hi.baidu.com/flysky_2005) |  |
| [invisiable](http://hi.baidu.com/invisiable) | [大漠孤狮](http://hi.baidu.com/大漠孤狮) | [zzhou001](http://hi.baidu.com/zzhou001) | [kvoova](http://hi.baidu.com/kvoova) | [craiggiarc](http://hi.baidu.com/craiggiarc) | [橡皮树的博客](http://hi.baidu.com/橡皮树的博客) | [flysky\_2005](http://hi.baidu.com/flysky_2005) |  |

网友评论：

窗体顶端



窗体底端

窗体顶端



发表评论：

|  |  |
| --- | --- |
| 姓　名： | leeihcy  \*姓名最长为50字节 |
|  |  |
| 内　容： | |  |  |  |  | | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | 插入表情 |  | ▼ 闪光字 | | |  | |
|  |  |
|  | 同时将此文章分享给好友 |
| 验证码： | 请点击后输入四位验证码，字母不区分大小写  [看不清?](http://hi.baidu.com/reallty/blog/item/40a49b02d5b8bb024bfb511c.html) |
|  | [取消回复](http://hi.baidu.com/reallty/blog/item/40a49b02d5b8bb024bfb511c.html) |

窗体底端

|  |  |  |
| --- | --- | --- |
|  |  |  |

[帮助中心](http://help.baidu.com/question?prod_en=space) | [空间客服](http://tieba.baidu.com/f?kw=%B0%D9%B6%C8%BF%D5%BC%E4&fr=wwwt) | [投诉中心](http://tousu.baidu.com/hi/) | [空间协议](http://www.baidu.com/search/hi_contract.html)

©2012 Baidu



