BitMap 图像处理

1900012476 夏罗生

使用方法

```
process_bmp.exe input.bmp rotate angle out.bmp   // 旋转角度
process_bmp.exe input.bmp resize newWidth newHeight out.bmp // resize
process_bmp.exe input.bmp h_flip out.bmp     // 水平翻转
process_bmp.exe input.bmp v_flip out.bmp     // 垂直翻转
```

BitMap 文件格式

文件头

位图格式的文件头长度可变，而且其中参数繁多。但是我们日常生活中遇到的 .bmp 格式图片的文件头长度绝大多数都是 54 字节，其中包括 14 字节的 Bitmap 文件头以及 40 字节的 DIB (Device Independent Bitmap) 数据头，或称位图信息数据头（BItmap Information Header）。

Bitmap File Header			
位置（hex/dec）		尺寸（byte）	描述
00	0	2	头文件字段 对于常见的 bmp 文件，内容为 0x424D（对应 ASCII 码 BM）
02	2	4	整个 .bmp 文件的大小（little endian）
06	6	2	预留字段，通常为 0
08	8	2	
0A	10	4	图片信息的开始位置

Bitmap Information Header			
位置（hex/dec）		尺寸（byte）	描述
0E	14	4	DIB header 的大小 通常为 40 bytes 即 0x28
12	18	4	图像宽度（little endian）
16	22	4	图像高度（little endian）
1A	26	2	色彩平面（color plane）的数量 必须为 1
1C	28	2	每像素用多少 bit 来表示
1E	30	4	采用何种压缩方式 通常不压缩，即 BI_RGB，对应值为 0
22	34	4	图片大小（原始位图数据的大小） 对于不压缩的图片，通常表示为 0
26	38	4	横向分辨率（像素/米）
2A	42	4	纵向分辨率（像素/米）
2E	46	4	调色板中颜色数量 通常为 0（不表示没有颜色）
32	50	4	重要颜色的数量（通常被忽略） 通常为 0，表示每种颜色都重要

原始位图数据 Raw Bitmap Data

数据按照像素行进行包装，便于读取。但是这并不是全部，因为其中还可能会有补零（zero-padding）。这涉及到计算机的数据结构对齐（data structure alignment）的问题。

主流的 CPU 每次从内存中读取并处理数据块（chunk），且通常为 32 比特（4 字节）。因此，为了提升读取效率，位图每行的数据（字节）都需要是 4 的倍数。不可避免地，有些时候每行的结尾就会出现补零（其实补其他任意数字也是可以的，但常见都是补 0）。

每行补零的字节数的计算公式为：

```
(4 - ((biWidth * BYTE_PER_PIX) & 0x3)) & 0x3
```

`&` 符号表示 按位与 运算符，`&3` 表示对 4 取余。

图像旋转

坐标变换

规定顺时针旋转为正，x 轴向右，y 轴向上。

以图像中心（a,b）为旋转中心，旋转角度 θ，计算图像上任一点（x1,y1）旋转之后的坐标（x2,y2）。

以（a,b）为新坐标原点，则（x1,y1）的新坐标为

$$x_1' = x_1 - a$$
$$y_1' = y1 - b$$

（x2,,y2）的新坐标为

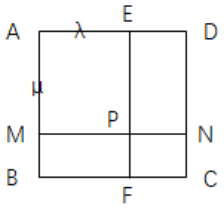
$$x_2' = cos\theta x_1' + sin\theta y_1'$$
$$y_2' = -sin\theta x_1' + cos\theta y_1'$$

则（x2,y2）的原坐标为

$$x_2 = x_2' + a = cos\theta(x_1 - a) + sin\theta(y_1 - b) + a$$
$$y_2 = y_2' + b = -sin\theta(x_1 - a) + cos\theta(y_1 - b) + b$$

双线性插值

如果旋转之后的像素点并不是很如人意的落在像素点上，而是落在临近的四个像素点构成的正方形区域内（而且这种情况应该是很常见的一种），我们通过双线性插值确定旋转之后的像素的颜色值。



假设我们旋转之后的点P落在四个像素点A、B、C、D构成的正方形区域，而且假设点P离点A的水平距离为λ（也即AE为λ），离A的处置距离为μ（也即AM为μ），那么要求的就是使用A，B，C，D四点的颜色值，以及λ和μ来表示点P的颜色值。

解：我们这里使用的是双线性插值的方式：
使用这种方式我们需要假设颜色值在每一条线上是呈线性变化的：
首先在线段AD上，那么我们就可以计算出点E的颜色值：

$$C_E = (1 - \lambda)C_A + \lambda C_D$$

同理我们可以计算出点F的颜色值：

$$C_F = (1 - \lambda)C_B + \lambda C_C$$

在线段EF中我们就可以算出点P的颜色值：

$$C_P = (1 - \mu)C_E + \mu C_F$$

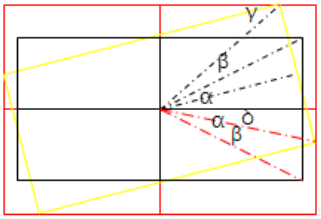
将以上的C_E和C_F带入则我们得到：

$$C_P = (1 - \mu)(1 - \lambda)C_A + (1 - \mu)\lambda C_D + \mu(1 - \lambda)C_B + \mu\lambda C_C$$

以上就是我们的双线性插值公式

实现旋转

新图像的高度和宽度取决于两条对角线。只需要求出两条对角线旋转后的宽度和高度，两者的最大值就是新图像的宽度和高度。下面给出求一条对角线宽度和高度的方法。



在左边的图中，我们的原始图像是黑色矩形，假设旋转β°之后得到黄色的矩形，我们想要完全显示这个图像就至少需要如图红色矩形的空间，我们现在的已知就是原始图像的宽为width，高为height，旋转角度为β°，求旋转之后的newWidth和newHeight。

解：如图我们在有γ的RtΔ中，我们易知

$$\gamma = \alpha + \beta$$

$$sin\alpha = \frac{height}{2R}, \quad cos\alpha = \frac{width}{2R}$$

$$sin\gamma = \sin(\alpha + \beta) = \frac{newHeight}{2R}$$

（此处的R应当是原始图像对角线的一半）

$$\therefore newHeight = height * cos\beta + width * sin\beta$$

如图我们在有δ的RtΔ中，我们易知

$$\delta = \alpha - \beta$$

$$sin\alpha = \frac{height}{2R}, \quad cos\alpha = \frac{width}{2R}$$

$$sin\delta = \cos(\alpha - \beta) = \frac{newWidth}{2R}$$

$$\therefore newWidth = width * cos\beta + height * sin\beta$$

然后对应新图层的每一个像素点，计算出对应的原始图像中的像素点，使用双线性插值确定颜色值。