2020

# 数字图像处理第一次报告

学生姓名：任泽华

班级：自动化 71

学号：2171411498

提交日期：2020-2-29

摘要 ：

本报告主要工作：针对一个具体的 bmp 图像对 bmp 图像的格式进行了简介；对于一幅灰度图像进行了 8-1 灰度级别的变换，解决了最后一张图片灰度不对的问题；对该图片进行了求取均值方差的操作；分别采用最近邻、双线性、双三次三种插值法对图片进行了插值，并且比较了三种插值法得到的图片的质量和分别需要的时间；对两幅图像分别进行了错切变换和旋转变换，解决了水平变成垂直错切的问题，比较了两种旋转方式的效果。本报告软件运行环境为 MATLAB R2018b，所有代码均为自己编写，在编写过程中主要参考了 CSDN 相关帖子与 MATLAB 官方网站。（参考文献）

# 一、 Bmp 图像格式简介,以 7.bmp 为例说明[2]~[6]

## 1. 位图文件与矢量图文件

位图文件可以理解为由一个个像素点组成的图像，每个像素点都由一组数据对应表示；而矢量图文件是由直线和曲线等几何图形来表示的图像。区别就是位图文件在不断放大后最终会成为一个个像素点，而矢量图文件可以无限放大。

而对于 bmp 文件来说，它的每一个像素都有对应是数据表示，并且没有经过压缩，所以不管图像细节多少，质量高低，只要它们的像素点相同并且像素的属性一致，它们所占用的空间的一样的。

## 2. bmp 文件结构

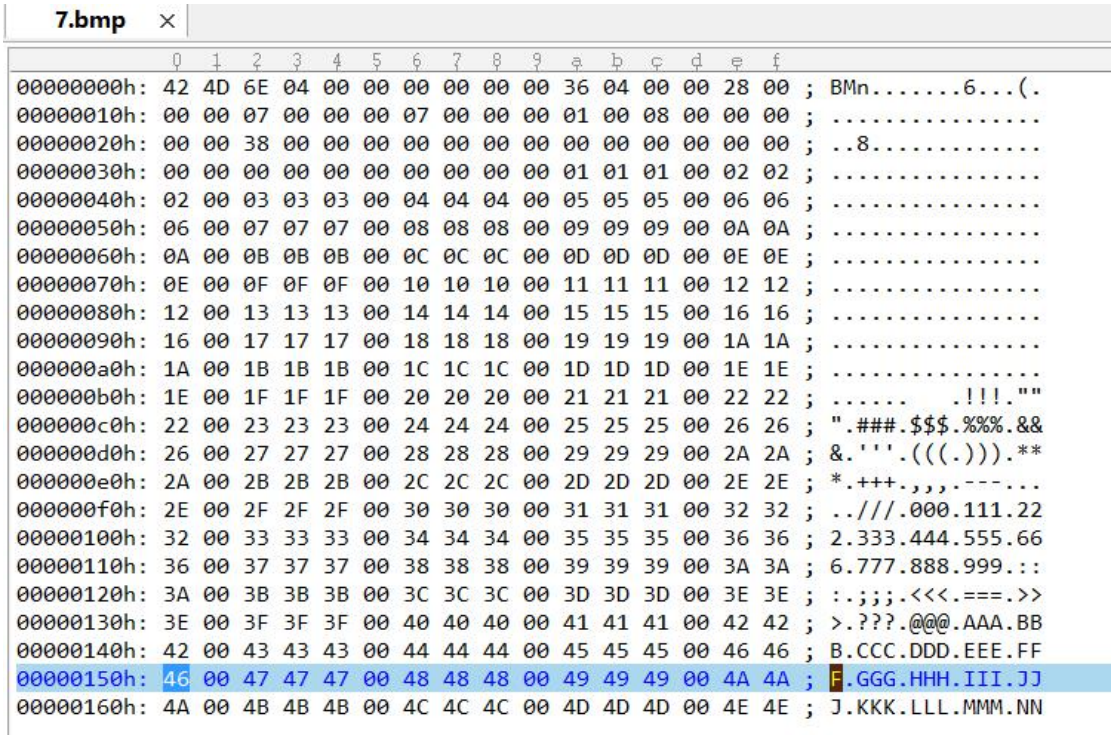
BMP 文件由 4 部分组成：

1. 位图文件头(bitmap-file header)
2. 位图信息头(bitmap-informationheader)
3. 颜色表(color table)
4. 颜色点阵数据(bits data)

24 位真彩色位图没有颜色表，所以只有 1、2、4 这三部分。

## 3. 对应于文件结构，分析 7.bmp 文件

用软件打开 7.bmp：




## （1）位图文件头(bitmap-file header)

| 名称 | 字节 | 意义 | 数据（十六进制） | 说明 |
|---|---|---|---|---|
| bfType | 2 | 标识，就是“BM” | 42 4D | BM（对应ASCII 码） |
| bfSize | 4 | 整个 BMP 文件的大小 | 0000046E(1134) | 占空间1134 字节 |
| bfReserved1/2 | 4 | 保留字，没用 | 0 | |
| bfOffBits | 4 | 偏移数，即位图文件头、位图信息头、调色板的大小 | 00000436（1078） | 偏移内容占空间1078 字节 |

**注：由于 Windows 的数据是倒着念的，如果 bfSize 的数据为 36 00**

**0C 00，实际上就成了 0x000C0036，也就是 0xC0036。**

## （2）位图信息头(bitmap-informationheader)

| 名称 | 字节 | 意义 | 数据（十六进制） | 说明 |
|---|---|---|---|---|
| **biSize** | 4 字节 | 位图信息头的大小，为 40 | 0x28(40) | 信息头大小 40 字节 |
| **biWidth** | 4 字节 | 位图的宽度，单位是像素 | 0x07(7) | 宽度 7 像素 |
| **biHeight** | 4 字节 | 位图的高度，单位是像素 | 0x07(7) | 高度 7 像素 |
| **biPlanes** | 2 字节 | 固定值 1 | 1 | |
| **biBitCount** | 2 字节 | 每个像素的位数 1-黑白图，4-16 色，8-256 色，24-真彩色 | 0x08(8) | 表示这是一个 256 色的图片 |
| biCompression | 4 字节 | 压缩方式，BI_RGB(0) 为不压缩 | 0 | 不压缩 |
| biSizeImage | 4 字节 | 位图全部像素占用的字节数，BI_RGB 时可设为 0 | 0x38（56） | 全部像素占用 56 字节 |

| | | | | |
|---|---|---|---|---|
| biXPelsPerMeter | 4 字节 | 水平分辨率(像素/米) | 0 | |
| biYPelsPerMeter | 4 字节 | 垂直分辨率(像素/米) | 0 | |
| biClrUsed | 4 字节 | 位图使用的颜色数如果为 0，则颜色数为 2 的 biBitCount 次方 | 0 | $2^8$=256 一共 256 个颜色 |
| biClrImportant | 4 字节 | 重要的颜色数，0 代表所有颜色都重要 | 0 | |

注：

**当 biBitCount=1 时，8 个像素占 1 个字节；**

**当 biBitCount=4 时，2 个像素占 1 个字节；**

**当 biBitCount=8 时，1 个像素占 1 个字节；**

**当 biBitCount=24 时，1 个像素占 3 个字节；（真彩色位图）**

作为真彩色位图，我们主要关心的是 biWidth 和 biHeight 这两个数值，两个数值告诉我们图像的尺寸。biSize，biPlanes，biBitCount 这几个数值是固定的。[2]

## （3）颜色表(color table)

颜色表的数据结构为：

```
typedef struct tagRGBQUAD
{
BYTE rgbBlue;
```

BYTE rgbGreen;
BYTE rgbRed;
BYTE rgbReserved;
}RGBQUAD;

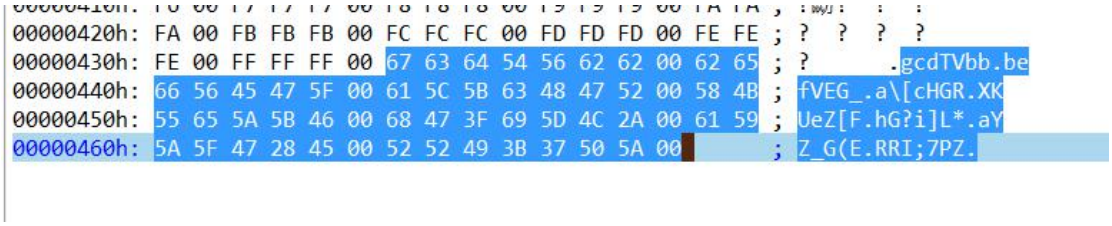
24 位位图在存储时，每行像素所占的存储空间大小=(31+24*width)/(4*32)。

对于 1,4,8 位图像来说，在位图信息后会有一个颜色表项，项数的多少与位图像素位数有关。1 位像素有 2 个颜色表项，4 位有 16 个，8 位有 256 个。其中，这些位数的像素值并不是真正的颜色，而是指向颜色表的索引，根据索引得到颜色表中的颜色项。在位图信息结构体中有变量 biClrUsed 变量来表示在实际使用的颜色表中的颜色数，biClrImportant 来表示位图显示过程中重要的颜色数。[6]

**此图为 256 色，故颜色表共有 256×4=1024 字节。**

## （4）颜色点阵数据(bits data)

由前文可知，偏移量 1078 字节，其中位图文件头 14 字节，位图信息头 40 字节，颜色表 1024 字节，剩下颜色点阵数据仅有 1134-1078=56 字节。如下图：




真正存储的与像素有关的数据就这么一点。如果图像比较大了以后，

文件头和颜色表的意义才能显示出来。

注：关于 7 行 7 列 7×7=49 而像素数据为 56 位的问题：

**Windows 规定一个扫描行所占的字节数必须是 4 的倍数(即以 long 为单位),不足的以 0 填充。**[4]

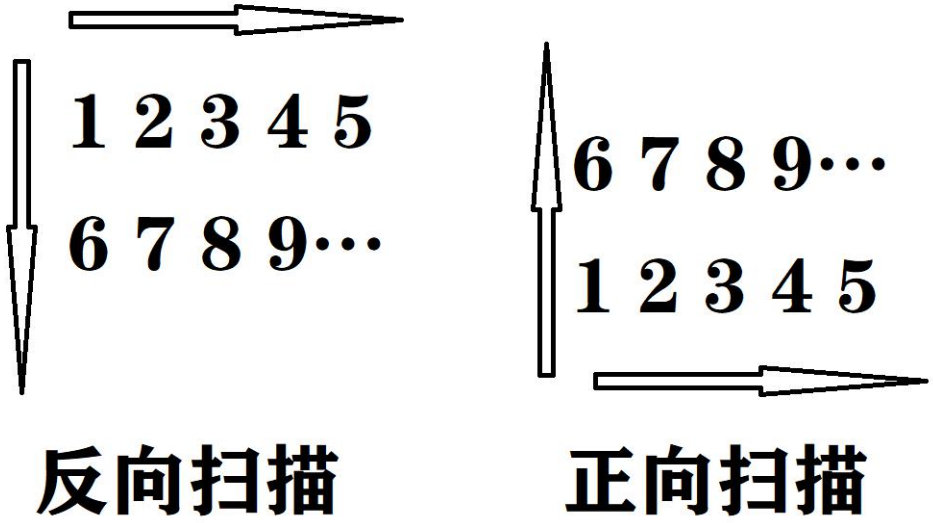
所以一行 7 个像素只能用 8 个字节来填充，7 行就是 7×8=56 个字节！

## 4. 关于扫描方式：

前面三篇文献关于 bmp 图片的扫描方式的描述有矛盾。（文献 1）：位图全部的像素，是按照自下向上，自左向右的顺序排列的。（文献 2）：第四部分是图像数据类，一幅图的数据顺序是从左往右，然后从上往下，（举个例子 2x2，顺序就是 0,0　0,1　1,0　1,1 ）（文献 3）：位图数据记录了位图的每一个像素值，记录顺序是在扫描行内是从左到右，扫描行之间是从下到上。

而这些在文献 4 中找到了答案：BMP 文件头偏移 0x16h 处，此位置是个 DWORD 值，用于存储 BMP 文件的高度。**这里如果是负数则 BMP 文件数据记录顺序从上向下，这里是正数则 BMP 文件数据扫描顺序从下向上。**本图片该参数（即 biHeight）为正值，故应该是

自下向上，自左向右扫描。（画一个灵魂示意图）

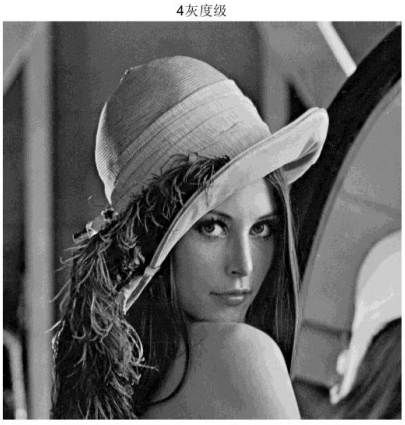


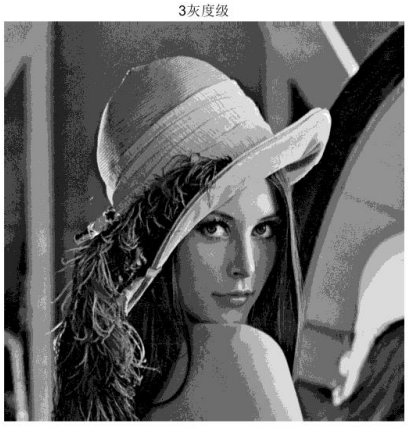

# 二、 把 lena 512*512 图像灰度级逐级递减 8-1 显示[7]

## 1. 输出效果

利用 matab 导入 lena.bmp 并进行处理以后得到如图所示的八幅图片：



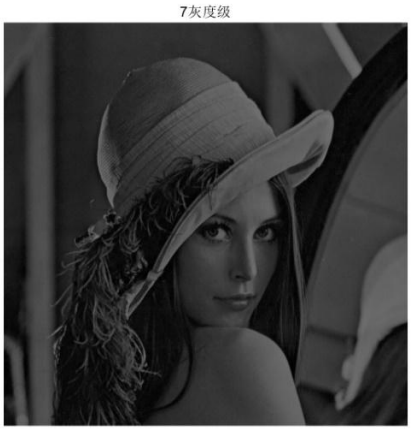


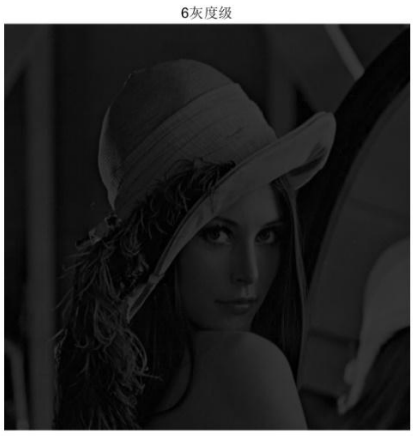

（8 灰度为直接显示的原图，所有图片保存为 png 格式，原始 bmp 格式见附件）

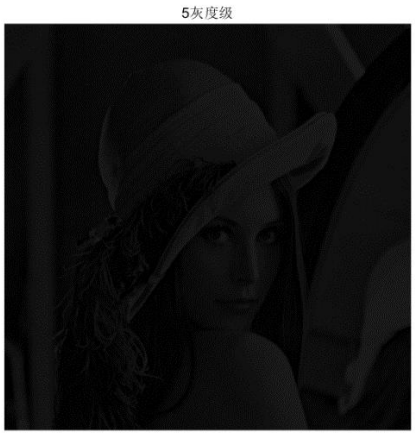
## 2. 遇到的问题与解决方案：

### （1）显示变暗

发现显示的图片越来越黑，到 5 灰度时已不可见，如图所示：


原因：上网查阅资料发现“imread() 返回的图像类型是 uint8 类型，这时用 imshow 显示图像的时候， imshow 会认为输入矩阵的范围在 0-255， 如果 imshow 的参数为 double 类型的，那么 imshow 认为输入矩阵的值为 0-1.”[7]在默认下按照 0-255 分配灰度，虽然值变小

了，但是仍然是按照 0-255 输出，就导致图像越来越黑。将参数随每次迭代不断调整，就可以避免这种情况。

## （2）最后一张图像黑色不黑

最后一张图片按理来说应该是黑白两色，但是实际出来的图片是全白色：


要求图像　　实际图像

刚开始显示范围设置为（0，2）但是 imread 读入的像素数据是 uint8 格式，如果像素值已经为 1，在进行除 2 运算后仍然为 1，所以在显示范围为（0，2）的情况下值为 1 的位置显示为灰，但是若将其改为（0，1），由于全部都是大于 1 的，会成为一片白色，故采用（1，2），以此类推，其他图像也采用（1，range）格式，range 为当前显示范围的大小。这样的范围不会影响显示效果，也更符合常理。

# 三、 计算 lena 图像的均值方差

## 1. 输出结果

```
m =

   99.0512


s2 =

   2.7960e+03
```

均值=99.0512 方差=2.7960e+03

## 2. 相关讨论

这道题比较简单，使用 matlab 的 mean2、std2 函数即可完成求解，

注意 std2 函数求得的是标准差，所以要求平方才行。

# 四、 把 lena 图像用近邻、双线性和双三次插值法 zoom 到 2048*2048[8]

## 1. 输出效果


原图片


最近邻


双线性


双三次

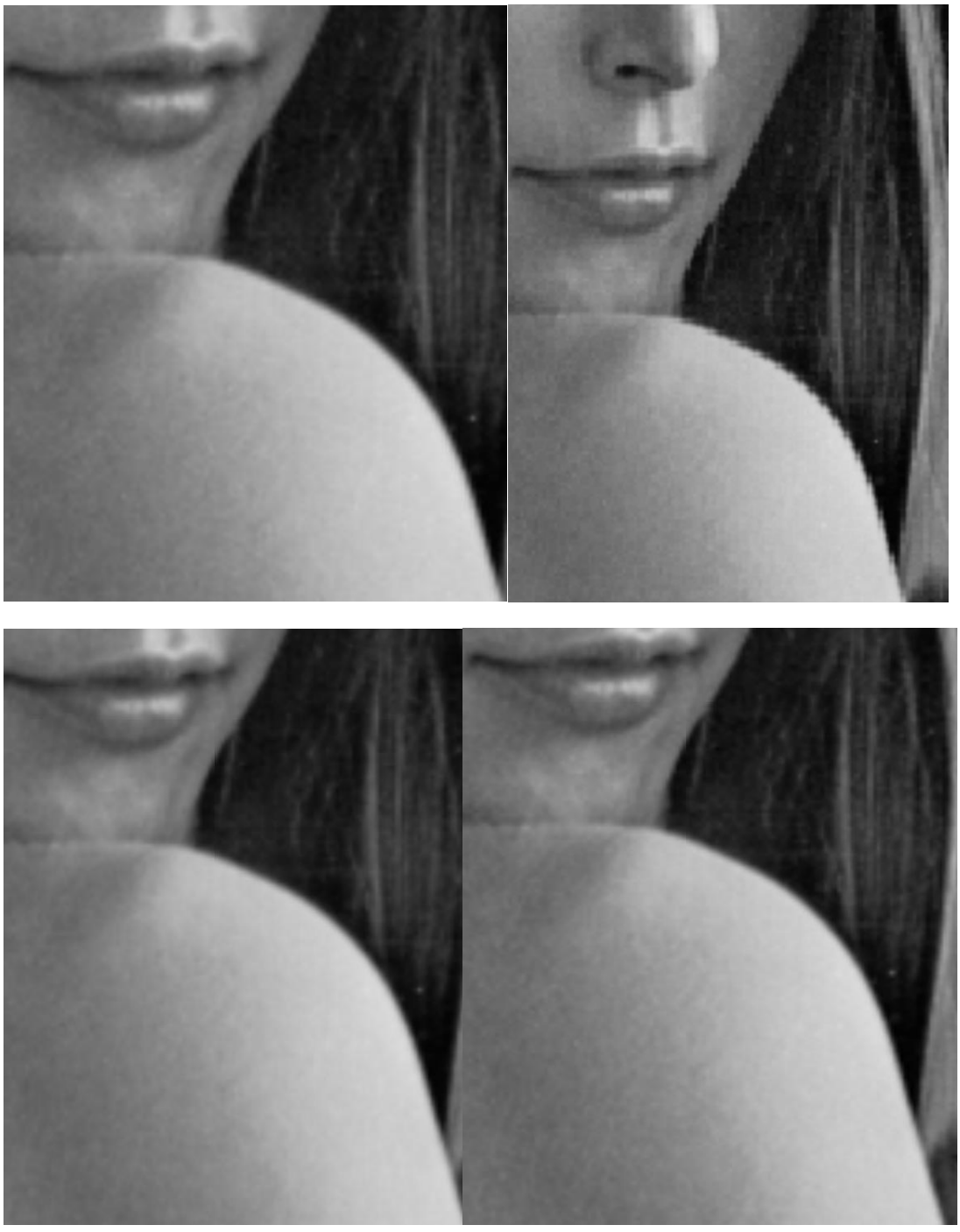
注：此处展示的是 png 格式，bmp 原图见附件。

从小图直观看来并没有什么明显区别，下面让我们放大看看。

## 2. 分析

### （1）图像对比

下面是局部放大后的图像，它们的位置分别对应上面的四张图片。



可以看出，对应于不同的插值方式，人脸与肩部的轮廓有较明显的差别。与原图相比，最近邻插值的轮廓有许多毛刺，在放大的条件下非

常明显。而双线性插值法轮廓就明显圆滑了不少，而与之相比，双三次插值的轮廓更加圆润，显得自然。

我查看了 matlab 的 imresize 函数简介，官方是这样介绍的：

'nearest'最近邻插值；赋给输出像素的值就是其输入点所在像素的值。不考虑其他像素。

'bilinear'双线性插值；输出像素值是最近 2×2 邻点中的像素的加权平均值

'bicubic'双三次插值；输出像素值是最近 4×4 邻点中的像素的加权平均值（注意：双三次插值可能生成在原始范围之外的像素值。）[8]

这三种插值法，最近邻的规则最简单，计算也方便，但是利用的信息较少，所以图像失真较严重。而双线性和双三次插值法利用的信息更多，所以图像精度也比较高，但相应的也要付出更多的计算时间。

## （2）运行时间比较

使用 matlab 计时功能分别测试三种插值法所用时间：

最近邻：时间已过 0.250650 秒。

双线性：时间已过 0.274227 秒。

双三次：时间已过 0.364472 秒。

注：没有进行精确测试，仅运行一次利用 tic toc 来简要验证。

## （3）采用的函数简介

B = imresize(A,[numrows numcols]) 返回图像 B，其行数和列数由二

元素向量 [numrows numcols] 指定。

___ = imresize(___,method) 指定使用的插值方法。

介绍文档内给出了该函数提供的**另外几种插值核函数：**

'box' 盒形核

'triangle' 三角形核（等效于 'bilinear'）

'cubic' 三次方核（等效于 'bicubic'）

'lanczos2' Lanczos-2 核

'lanczos3' Lanczos-3 核

对于 box、lanczos2、lanczos3 三种核函数也进行了验证，图片效果和前几种差不多，放上来它们的运行时间：

**Box：时间已过 0.263135 秒。**

**lanczos2：时间已过 0.256748 秒。**

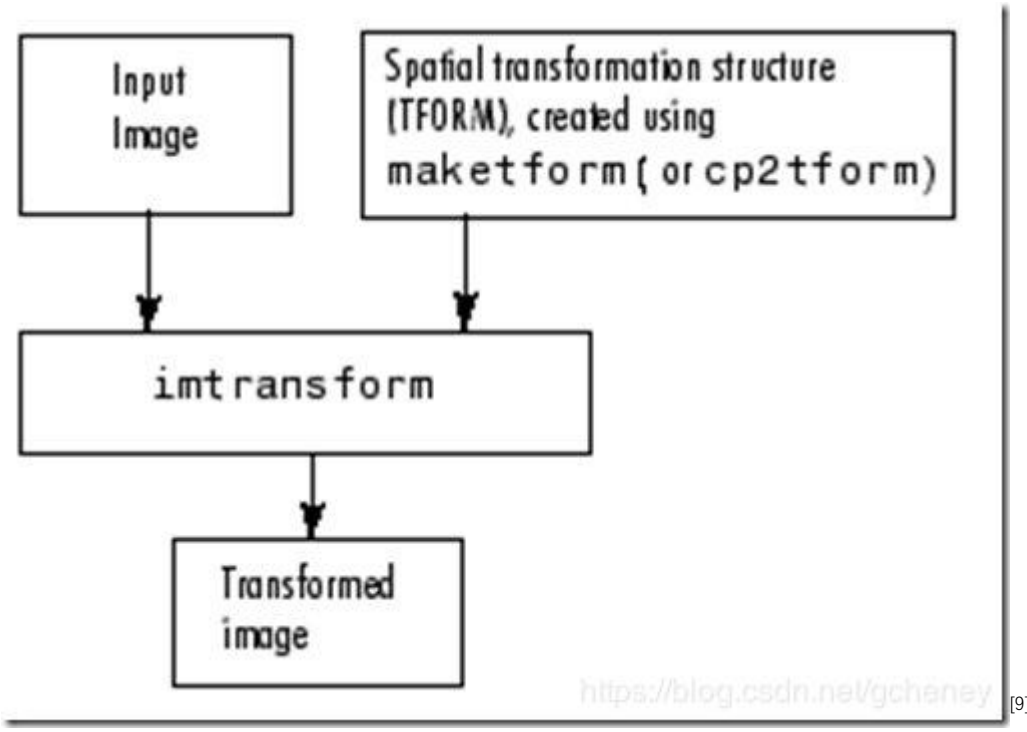
**lanczos3：时间已过 0.321961 秒。**

[8]

# 五、 把 lena 和 elain 图像分别进行水平 shear（参数可设置为 1.5，或者自行选择）和旋转 30 度，并采用用近邻、双线性和双三次插值法 zoom 到 2048*2048[9][10]

## 1. 图像变换思路

使用 maketform 函数构造 TFORM 结构体；将原图像和 TFORM 结构体导入 imtransform 函数生成变换后的图片[9]，再利用上问的方法进行插值，分析不同方法的差别。



[9]

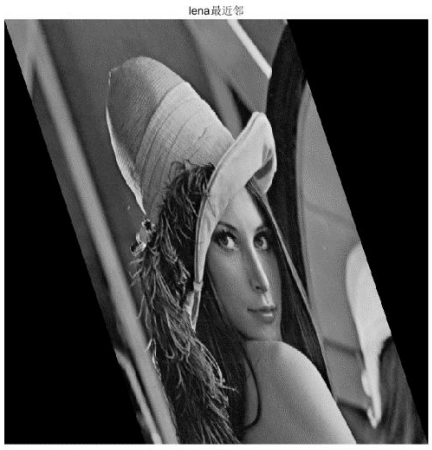
## 2. 水平 shear（参数设计为 0.5）

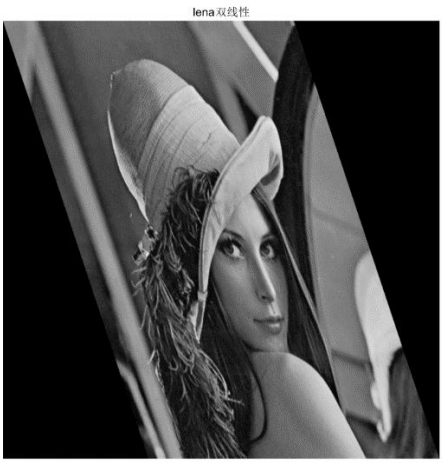
参数为 1.5 拉伸太大，看不清楚，故采用 0.5 为参数进行水平拉伸

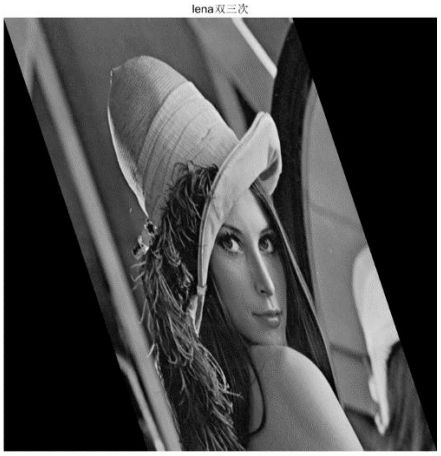
四幅图的顺序仍为：原图，最近邻，双线性，双三次

此处为 png 格式，bmp 原图见附件。

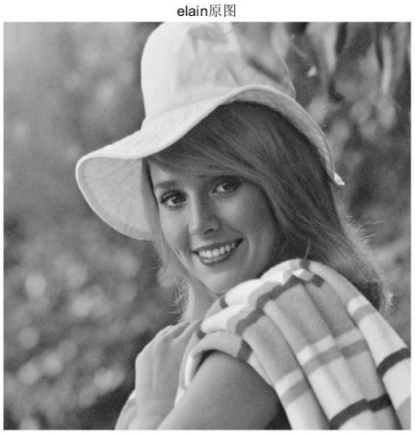
### （1）Lena

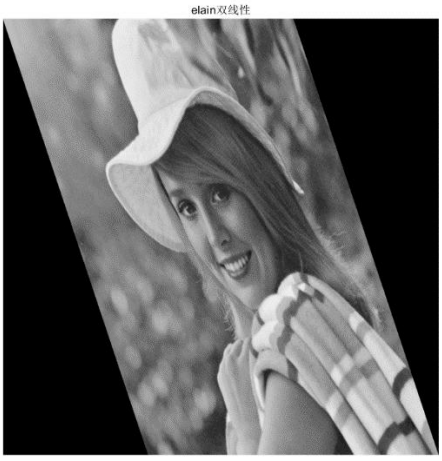



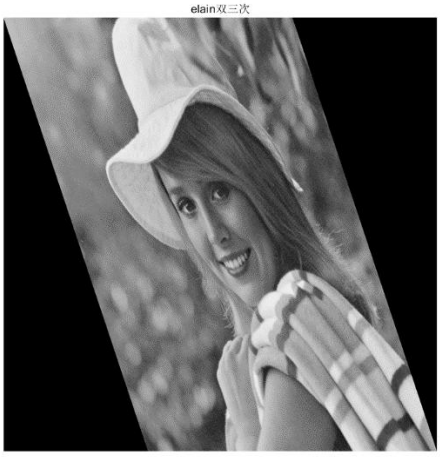



## （2）Elain



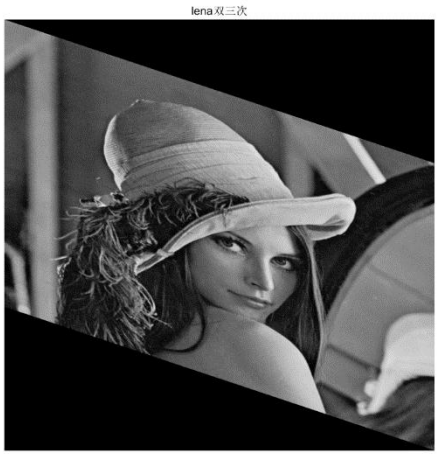


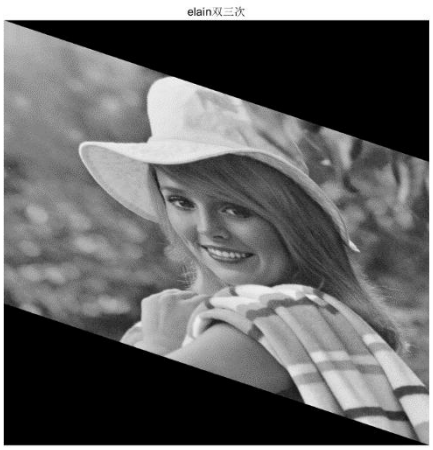



## （3）遇到的问题与分析

如下图所示，在一开始进行水平错切时，出来的图像是垂直错切的，而我利用的是书上给的仿射变换矩阵，即错切因子在（1，2）位置作为水平错切，但是结果却与事实不一样，当我改为（2，1）位置为错切因子时显示为水平错切，但在书上这是垂直错切的矩阵。




查阅 matlab 官方文档[11]之后我发现 matlab 水平错切的矩阵中错切因子确实是在（2，1）位置，这与书上给的不同。

## Examples

### Simple Transformation

Apply a horizontal shear to a grayscale image.

```
I = imread('cameraman.tif');
tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);
J = imtransform(I,tform);
imshow(J)
```

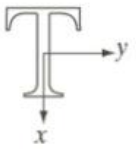
[11]

后来在查看 matlab 处理图片顺序时找到了答案：因为书上的坐标是

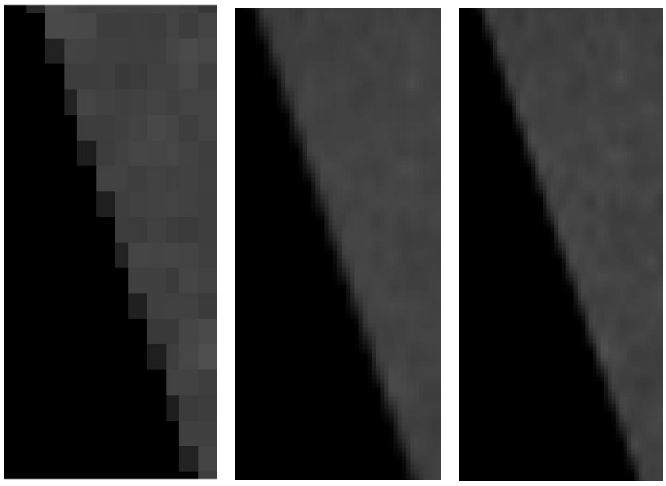
垂直为 x，水平为 y；而 matlab 中的图片处理是水平为 x，垂直为 y，

如果把 x、y 互换以后就成为了书上的公式。

Identity $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $x = v$ $y = w$




对于三种插值方式，在边缘处放大截取的图像如图所示，从左到右分别是最近邻、双线性、双三次。可以看出三种插值方式下边缘逐渐变得平滑，这与前一问也可以对应，而且对比度更明显。



## 3. 旋转 30 度

在一开始时，完全利用水平 shear 的方法，只不过把变换矩阵改成旋转变换的模样：

```
trans=[cosd(30) sind(30) 0; -sind(30) cosd(30) 0; 0 0 1];
```

四幅图的顺序仍为：原图，最近邻，双线性，双三次

此处为 png 格式，bmp 原图见附件。

### （1）Lena

lena原图

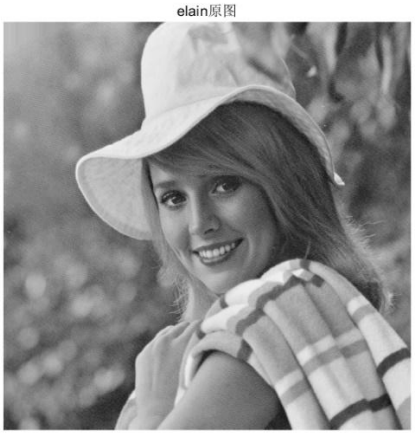
lena最近邻

lena双线性

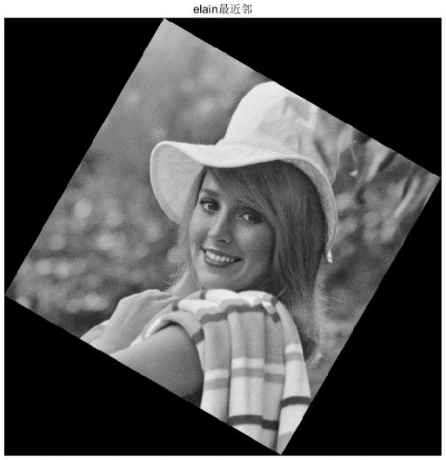
lena双三次

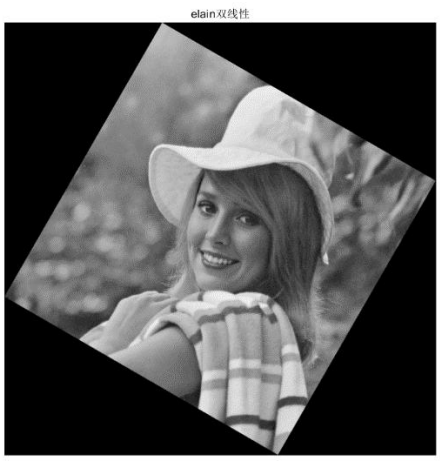
## （2）Elain

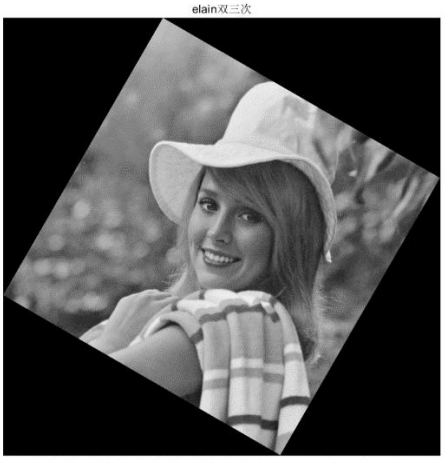
elain原图



elain最近邻

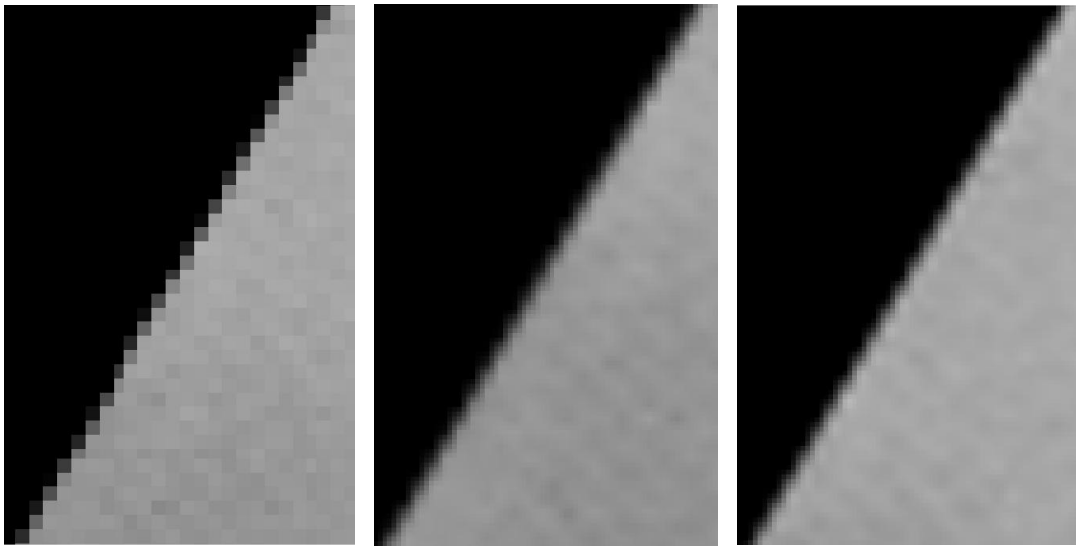





## （3）分析

对于三种插值方式，在边缘处放大截取的图像如图所示，从左到右分别是最近邻、双线性、双三次。可以看出三种插值方式下边缘逐渐变得平滑，这与前一问也可以对应，而且对比度更明显。



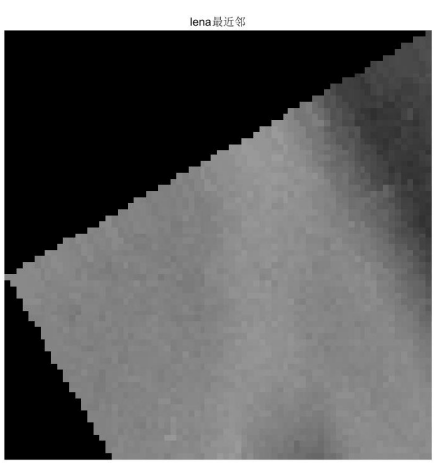
## （4）另一种旋转方法

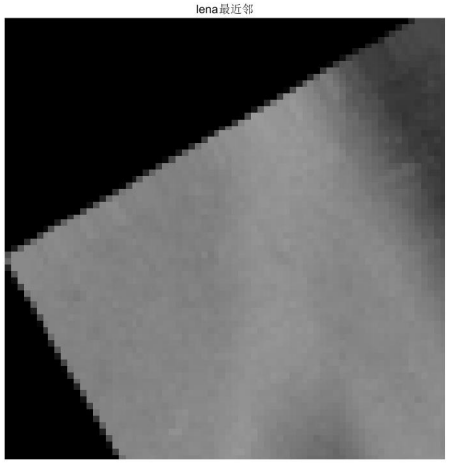
在搜集资料时也发现了 matlab 还有专门用于旋转的函数 imrotate，

输入原图片、旋转角度、插值方式，可以比较简单地实现图片旋转，而且插值也可以选择常见的几种方法。

下面分别是采用新方法得到的图片边缘和原始方法的对比：

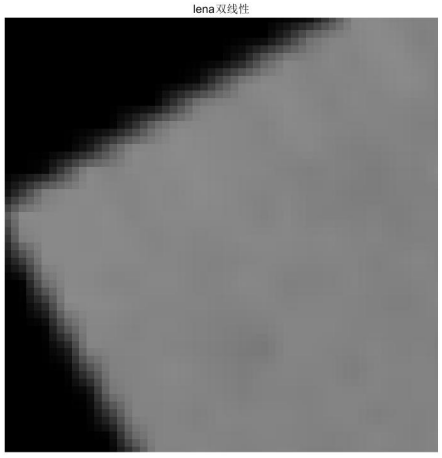
最近邻：

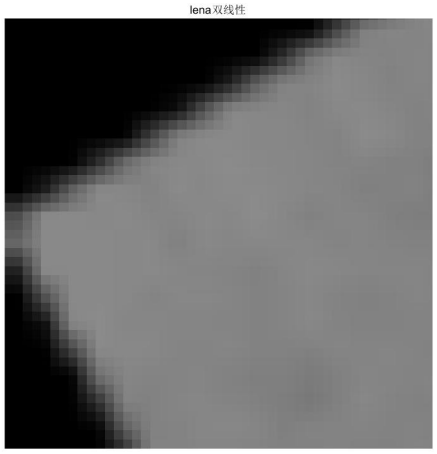



新方法　　老方法

双线性：




新方法　　老方法

对比可以看出，新方法最近邻插值下的边缘更加尖锐，双线性插值下不太明显（在放大倍数更高的情况下仍几乎看不出），双三次更看不出来，就没有进行比较。

由此可以发现：新方法由于多进行了一次插值，在效果不太好的最近邻插值下图片效果变差，而效果较好的双线性和双三次插值法二者的区别不明显，两种方法都可以采用。

# 附录

## 1. 参考文献

[1] Rafael C. Gonzalez（拉斐尔 C. 冈萨雷斯）,Richard E. Woods （理查德 E. 伍兹）. 数字图像处理(第三版)（英文版).北京：电子工业出版社. 2017 年.
[2] 位图(bmp)文件格式分析
https://blog.csdn.net/qingchuwudi/article/details/25785307
[3] 浅析 BMP 文件格式
https://jingyan.baidu.com/article/cd4c29795394c7756e6e60e9.html
[4] 关于图片存储格式的整理(BMP 格式介绍)
https://www.jb51.net/article/78186.htm
[5] BMP 文件如何区分数据记录顺序（扫描顺序）
https://zhidao.baidu.com/question/520550919.html
[6] bmp 文件颜色表
https://blog.csdn.net/A4079/article/details/24384595?locationNum=10&fps=1
[7] matlab imshow 显示图像详解
https://blog.csdn.net/qq_15295565/article/details/88410393
[8] 调整图像大小 - MATLAB imresize - MathWorks 中国
https://ww2.mathworks.cn/help/matlab/ref/imresize.html
Lanczos 插值，最邻近插值，双线性二次插值，三次插值
https://blog.csdn.net/wgx571859177/article/details/78963267
[9] 二维空间变换-了凡春秋的博客
http://blog.sina.com.cn/s/blog_6163bdeb0102du23.html
[10] MATLAB-imrotate 函数
https://blog.csdn.net/fifi0130/article/details/86009569
[11] imtransform-Apply 2-D spatial transformation to image
https://www.mathworks.com/help/images/ref/imtransform.html

## 2. 源代码

### （1）第二问

```
clear;clc;
lena=imread('lena.bmp');
imshow(lena);
title('8 灰度级');
[x,y]=size(lena);%读取图像长宽
range=256;%显示范围
for k=1:7
    range=range/2;
    p=int2str(8-k);%当前灰度级别
    for i=1:x
        for j=1:y
            lena(i,j)=floor(lena(i,j)/2);%灰度级别减小
        end
    end
    figure(k+1)
    imshow(lena,[1,range]);
    title([p '灰度级']);
end
```

### （2）第三问

```
clear;clc;
lena=imread('lena.bmp');
m=mean2(lena)%求均值
s=std2(lena);
s2=s^2%求方差
```

### （3）第四问

```
clear;clc;
```

```
lena=imread('lena.bmp');
imshow(lena);
title('原图片');
figure(2);
lena1=imresize(lena,[2048,2048],'nearest');%最近邻插值
imshow(lena1);title('最近邻');
figure(3);
lena2=imresize(lena,[2048,2048],'bilinear');%双线性插值
imshow(lena2);title('双线性');
figure(4);
lena3=imresize(lena,[2048,2048],'bicubic');%双三次插值
imshow(lena3);title('双三次');
```

## （4）第五问

1) 第一问

```
clear;clc;
lena=imread('lena.bmp');
elain=imread('elain1.bmp');
trans=[1 0 0;0.5 1 0;0 0 1]; %变换参数
T=maketform('affine',trans);%变换结构矩阵

lena0=imtransform(lena,T);%lena
figure(1);
lena1=imresize(lena0,[2048,2048],'nearest');%最近邻插值
imshow(lena1);title('lena 最近邻');
figure(2);
lena2=imresize(lena0,[2048,2048],'bilinear');%双线性插值
imshow(lena2);title('lena 双线性');
figure(3);
lena3=imresize(lena0,[2048,2048],'bicubic');%双三次插值
imshow(lena3);title('lena 双三次');

elain0=imtransform(elain,T);%elain
figure(4);
elain1=imresize(elain0,[2048,2048],'nearest');%最近邻插值
```

```
imshow(elain1);title('elain 最近邻');
figure(5);
elain2=imresize(elain0,[2048,2048],'bilinear');%双线性插值
imshow(elain2);title('elain 双线性');
figure(6);
elain3=imresize(elain0,[2048,2048],'bicubic');%双三次插值
imshow(elain3);title('elain 双三次');
```

2) 第二问

```
clear;clc;
lena=imread('lena.bmp');
elain=imread('elain1.bmp');
trans=[cosd(30) sind(30) 0; -sind(30) cosd(30) 0; 0 0 1]; %变换参数
T=maketform('affine',trans);%变换结构矩阵

lena0=imtransform(lena,T);%lena
figure(1);
lena1=imresize(lena0,[2048,2048],'nearest');%最近邻插值
imshow(lena1);title('lena 最近邻');
figure(2);
lena2=imresize(lena0,[2048,2048],'bilinear');%双线性插值
imshow(lena2);title('lena 双线性');
figure(3);
lena3=imresize(lena0,[2048,2048],'bicubic');%双三次插值
imshow(lena3);title('lena 双三次');

elain0=imtransform(elain,T);%elain
figure(4);
elain1=imresize(elain0,[2048,2048],'nearest');%最近邻插值
imshow(elain1);title('elain 最近邻');
figure(5);
elain2=imresize(elain0,[2048,2048],'bilinear');%双线性插值
imshow(elain2);title('elain 双线性');
figure(6);
elain3=imresize(elain0,[2048,2048],'bicubic');%双三次插值
imshow(elain3);title('elain 双三次');
```

3) 比较两种旋转方式代码（第二问基础上命令行输入）

```
lena0=imtransform(lena,T);%lena
lena2=imresize(lena0,[2048,2048],'bilinear');%双线性插值
imshow(lena2);title('lena 双线性');
figure
lena0=imrotate(lena,30,'bilinear');
lena2=imresize(lena0,[2048,2048],'bilinear');%双线性插值
imshow(lena2);title('lena 双线性');
```