**第一次作业**

自动化63

林子航

2160508273

2019年2月27日

摘要

本次实验深入学习了图像的基础，包括图像的基本构成及一些图像的基本变换，例如剪切，旋转，改变图像大小等等。

实验一探究了bmp格式的图像的基本构成。Bmp格式图像的存储空间由三部分组成，分别是BITMAPFILEHEADER，BITMAPINFOHEADER这两个头文件以及图像自身的灰度。图像的灰度级数确定了图像的灰度值的范围，也确定了存储图像内容所需要的位数bits。

实验二对图像进行了灰度级数的递减显示。通过调整灰度级数可以明显发现，对于细节较多的图像，并不需要过高的灰度级数，因为低的灰度级数能够给人更强的对比效果；而对于细节没有那么多的图像，灰度级数越高，给人整体的感觉是图片质量越好。

实验三求取了图像的灰度的均值和方差，直接使用matlab自带的mean和std函数求解。图像的灰度值均值表征了该图像的基本色调，而方差则表征了图像的对比度。

实验四对图像进行了三种插值方法的zoom操作。通过编写可任意变换大小的图像大小改变函数，对比三种不同的插值方法可以发现，最近邻插值法是比较粗糙但运算量最小的插值方法，适用于精度要求不那么高的情况；双三次插值法运算复杂度较高，同时其插值效果与具体参数有关，需要一定的经验；双线性插值法是比较优良也比较常用的插值方法，效果较好，运算量也适中。

实验五对图像进行了剪切，旋转的操作。通过编写剪切，旋转函数，我发现为了保证图片质量，在变换的过程中需要及时进行插值，以防止出现“窟窿”的情况。在变换结束后再使用实验四中编写的嵌入式函数，对图像进行zoom，改变图像大小至要求的2048\*2048大小。

通过本次作业，我深入探究了图片的基本结构，熟悉了图像的基本处理方法。所有处理函数均为自己编写自主探究，虽然遇到了很多障碍，并耗时较多，但锻炼了自己的编程技巧，收获颇丰。

实验一. Bmp图像格式探究（以7.bmp为例）

一.技术讨论

Bmp格式是一种标准图像文件格式，一张bmp格式的图像需存储三个维度的数据，分别是M,N,k，其中M，N表征图像的单元个数也即俗称的像素大小，分别表示图像的行像素数以及列像素数；k表征每个像素的位深度，bmp文件的图像深度可选k=1,4,8及24，每个像素的离散灰度级数即能表示的颜色数目为：

例如，常用的k=8可表示的灰度值为0至255。已知一副图像的M,N,k即可以计算得到该图像所需要的存储空间（比特）：

以7.bmp为例，给出该图像的详细信息



可以看到该图像的M=N=7，k=8，可以计算该图像所需的存储空间为392bit/8bit=49Byte。然而实际上该图片大小为1134字节，相差如此巨大，问题出在哪？

经过自主搜集资料，我找到了出现这样大的偏差的原因。Bmp格式的图像除了存储图像本身的信息以外，还需要存储BITMAPFILEHEADER，BITMAPINFOHEADER这两个头文件，需要占据54字节的存储空间；此外，还需要4\*256=1024字节存储颜色版。因此一副bmp格式的图像文件，无论其像素大小或是灰度级数是多少，都需要54+1024=1078字节的固定空间，因此7.bmp图像文件计算得到的存储空间应为1078+49=1127字节，这与实际大小是近似一致的。

以上结论也可以用其他图片得到证明。以lena.bmp文件为例，M=N=512，k=8，理论上所需要的存储空间为：

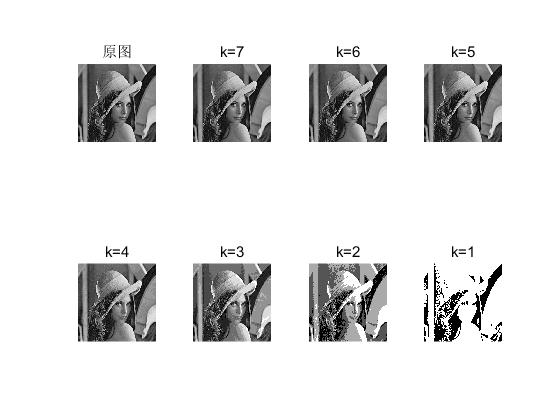
而实际大小完全一致，即263222字节。

实验二. 把lena 512\*512图像灰度级逐级递减8-1显示

一.技术讨论

灰度级数即图像详细信息里的位深度，lena图像的k=8，意味着图像的灰度值的范围为[0,255]，要求将图像的灰度级数逐级递减显示，思路是对进行除法取整操作，得到递减级数后的图像。设递减后级数为i，则有：

二.实验结果



三.结果分析

可以看到，对于lena.bmp图像而言，k在4至8之间的图像给人的主观感受变化并不大，再减小k值则会给人造成明显的图片质量下降的主观感受，这是因为这幅图像属于具有少量细节的图像，人对它的偏爱曲线将偏向右上角，即N和k越大图像质量越好。

对于一些具有大量细节的图像，k值往往不需要过大，因为小的k值可以增强对比度，给人更清晰的感觉。

实验三. 计算lena图像的均值方差

一.技术讨论

计算lena图像的均值方差只需要用到matlab自带的mean和std函数即可。若A是一个矩阵，则mean(A，dim)输出A矩阵在维度dim上的均值，要求全部灰度值的平均值只需要使用2个mean即mean(mean(A))即可。计算方差可使用std2(A)返回得到A中全体元素的标准差，再平方得到方差。

二.实验结果

实验四.使用插值法将lena图像zoom

一.技术讨论

实验要求使用三种插值方法将512\*512的图像放大至2048\*2048，即长宽均放大4倍。基本思路是先创造出任意的2048\*2048大小的矩阵T；然后将原先512\*512个像素的坐标全部放大4倍并替换T矩阵中对应的元素，这样就得到了未插值前的放大的图像；最后对各空白点分别用三种方法进行插值。

其中，需要特别注意的是双三次插值法。双三次插值法与双线性插值法思路相近，区别在于取了16个最近点的灰度值，采取一定的权重进行求和得到所求像素的灰度值。

设为变换得到的像素点在未变换的坐标系中的坐标（可能为小数），为原图像中像素点的坐标，表示P点的灰度值，表示原始图像中A像素点的灰度值。为P周围的最邻近的16个点的坐标。则每个坐标对应的权重由以下公式确定：

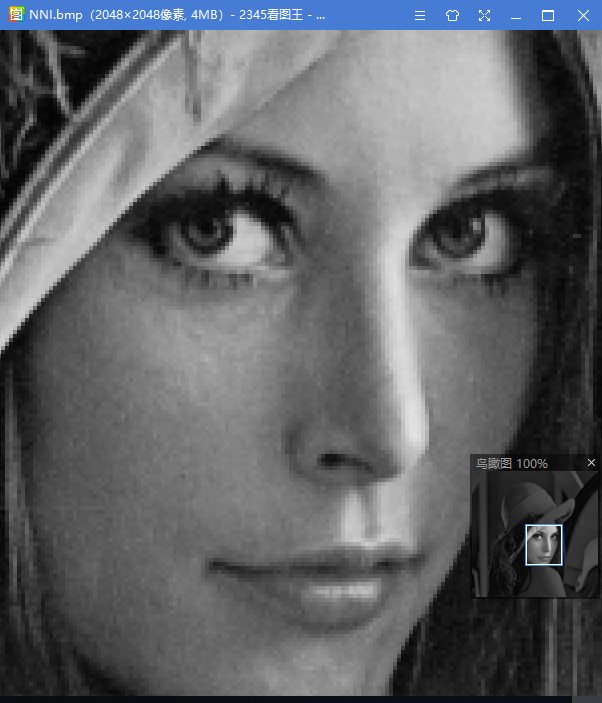
则由下式确定：

在本次实验中，上述函数中的a取0.5。

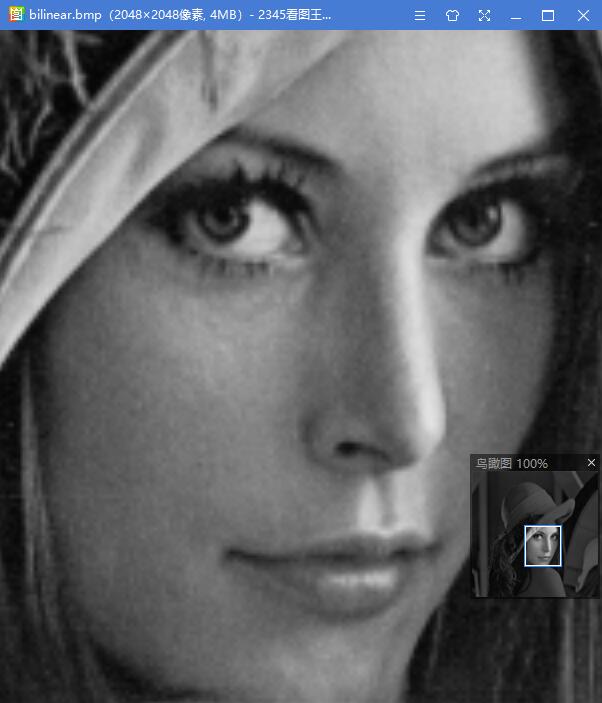
二.实验结果

由于2048\*2048图像过大，无法在电脑上100%显示，因此展示的图像均为截取三种插值方法得到的脸部局部图像，均放大至100%大小（2048\*2048），真实大小的图片已放在文件夹里以供查看。

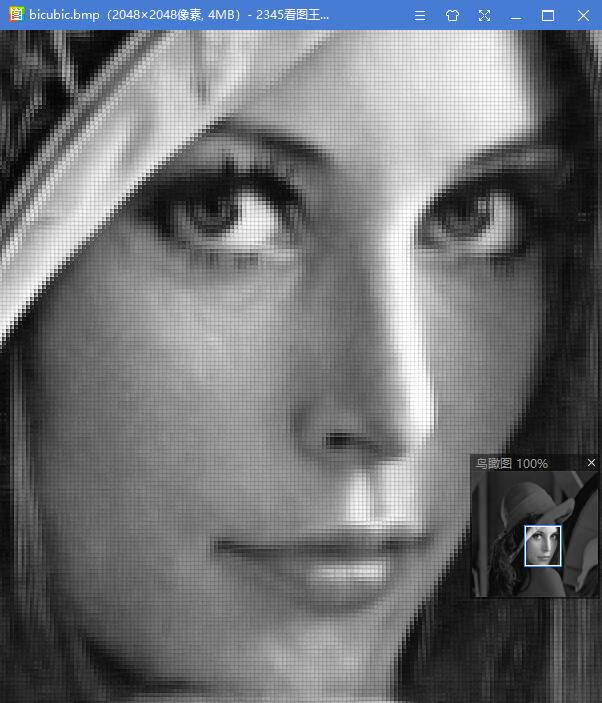
1.最近邻插值



2.双线性插值



3.双三次插值



三.结果分析

对比发现，采用最邻近插值法（NNI）的插值效果是最差的，可以清楚地看到一块一块马赛克一样的小方块；双三次线性插值效果也不好，锯齿状较明显，可能与选用的参数方程W(x)中的参数a的数值大小有关；效果最好的应该是双线性插值，整体看起来比较平滑，质量较好。

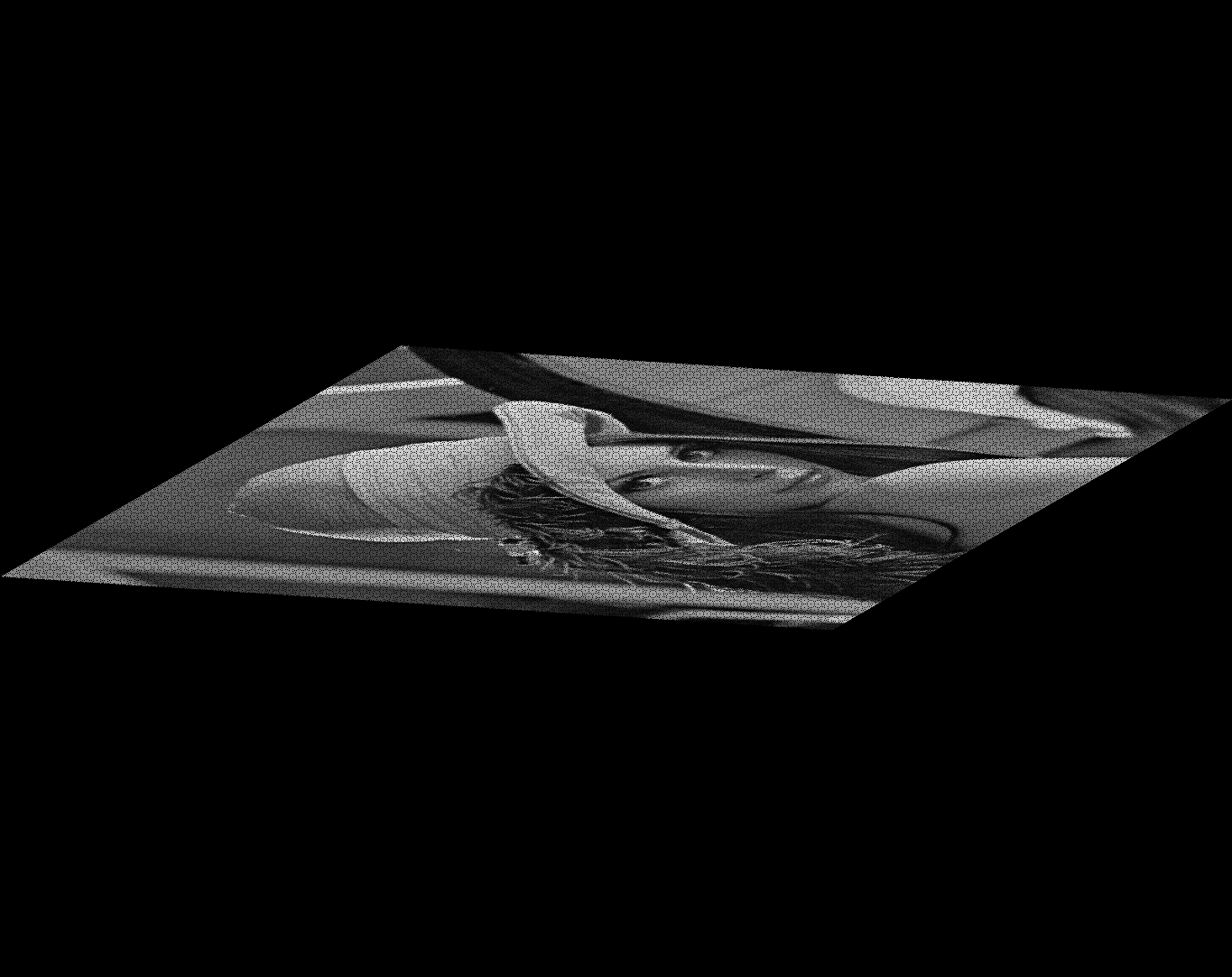
实验五.对图像进行shear,rotation并zoom

一.技术讨论

实验要求将lena.bmp和elain1.bmp两张图片分别进行参数为1.5的shear即剪切操作，再进行30°角的旋转，最后再分别利用三种插值方法zoom至2048\*2048大小。

在shear和rotation的过程中，基本思路是利用变换矩阵对原图像的坐标进行变换，变换后得到新的坐标系，然后将新的像素点的灰度值赋值为与之对应的原始图像坐标系下的像素点的灰度值。

在编写程序的过程中，我发现由于变换矩阵的存在，得到的新坐标大多数情况下是包含大量非整数点的。对于这种情况，若简单地对得到的坐标取整，则会产生大量的“窟窿”，即黑像素点：

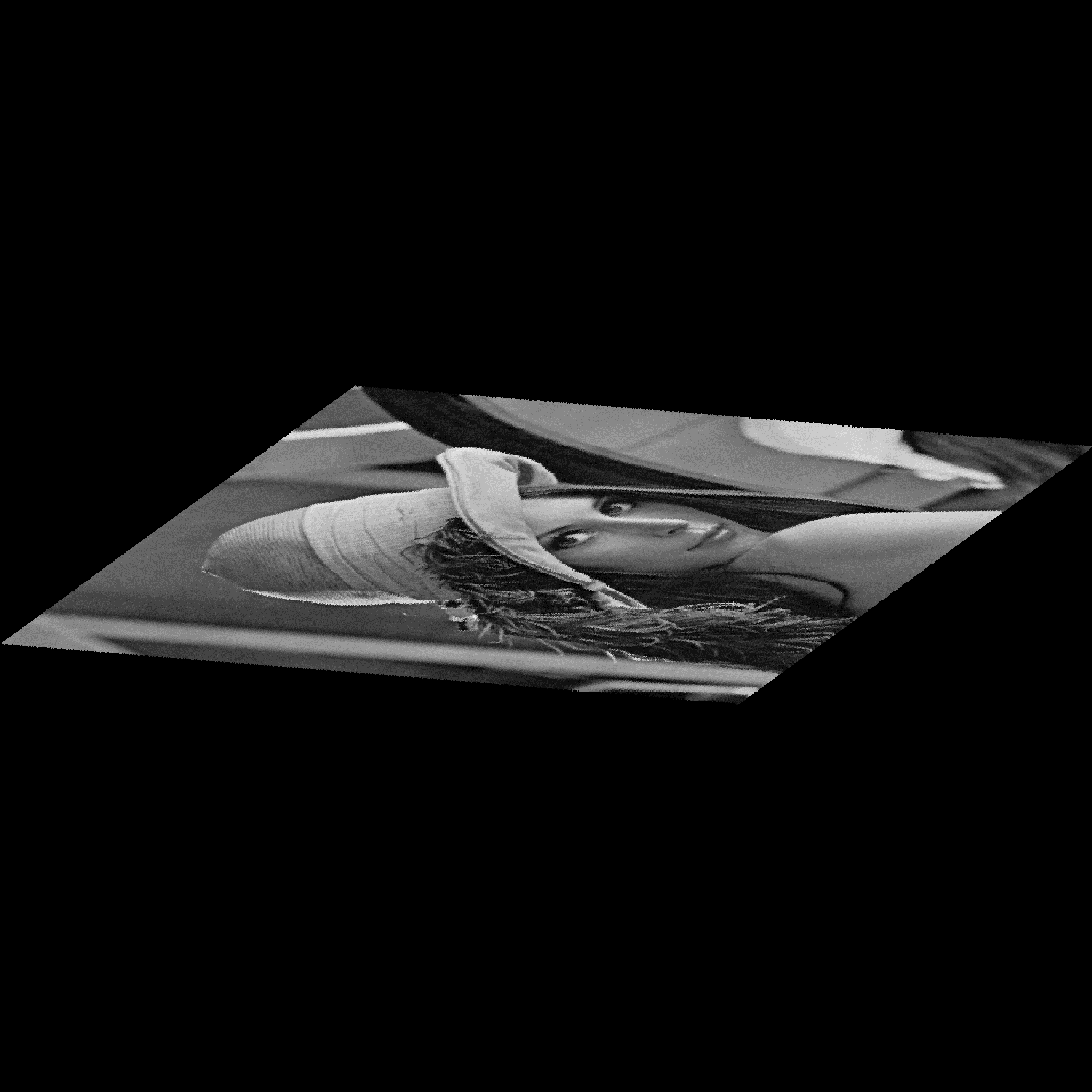


出现该现象的原因在于在坐标变换以后，图像的大小也发生了变换，而原始图像对应的点数是固定的，并不能填补上出现的窟窿。此外，在出现这种情况以后再进行插值，无论是哪种方法效果都不好，原因是窟窿太多且分布比较密集，常用的三种插值方法只能淡化它们的存在而无法彻底消除。

解决办法是在变换的过程中就对得到的新坐标系下的黑点进行插值。在本次实验中我使用了最简单的最近邻插值法，具体操作如下：

设为变换得到的像素点的坐标，其中s,v为整数部分，u,w为小数部分；为原图像中像素点的坐标，表示P点的灰度值，表示原始图像中A像素点的灰度值。则有：

则为了消除“窟窿”的存在，令点的灰度值为，得到的效果如下图：

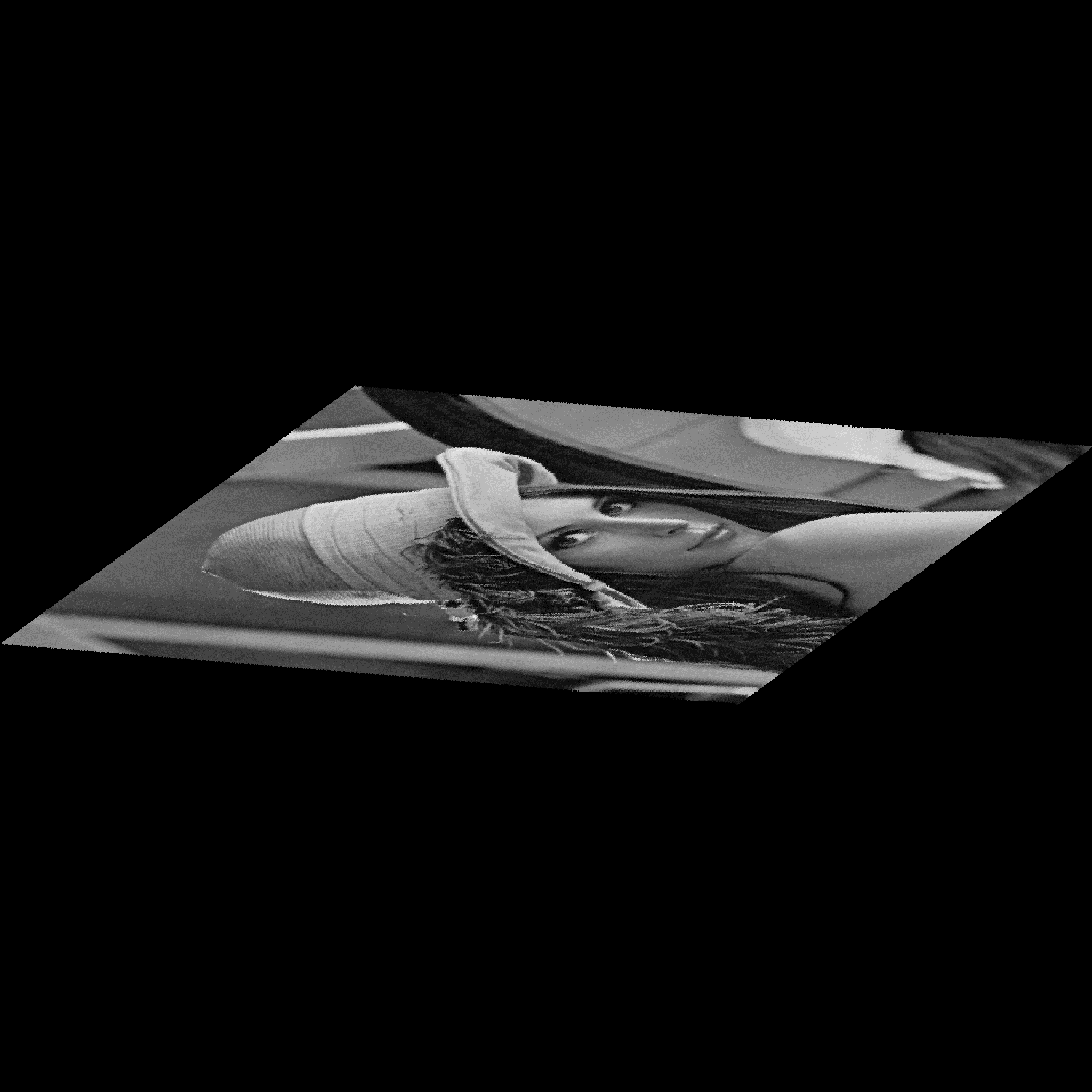


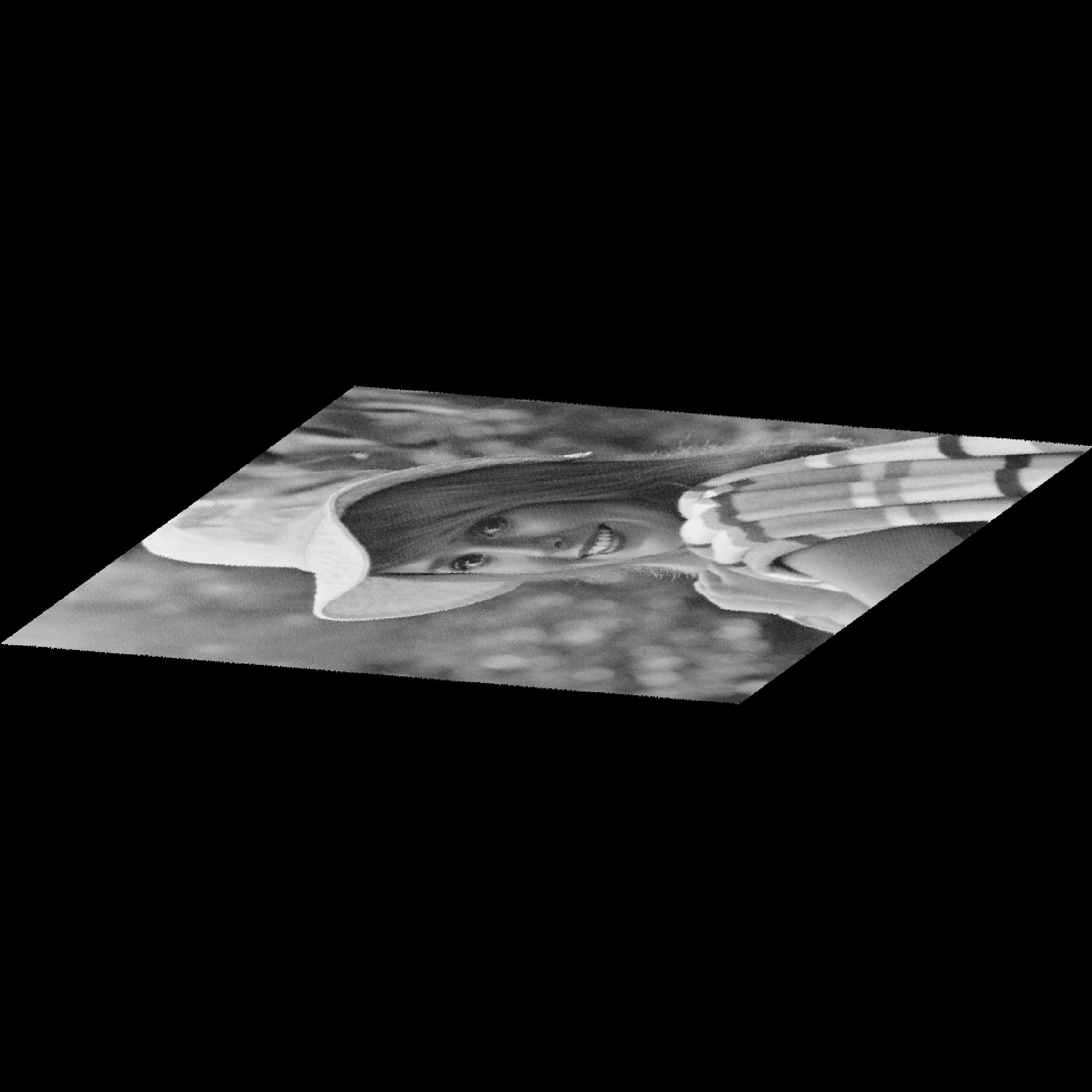
可以明显看到，经过最近邻插值的图片不再有窟窿的存在。

最后，将变换后的图像分别进行三种插值方法的插值，zoom至2048\*2048大小。

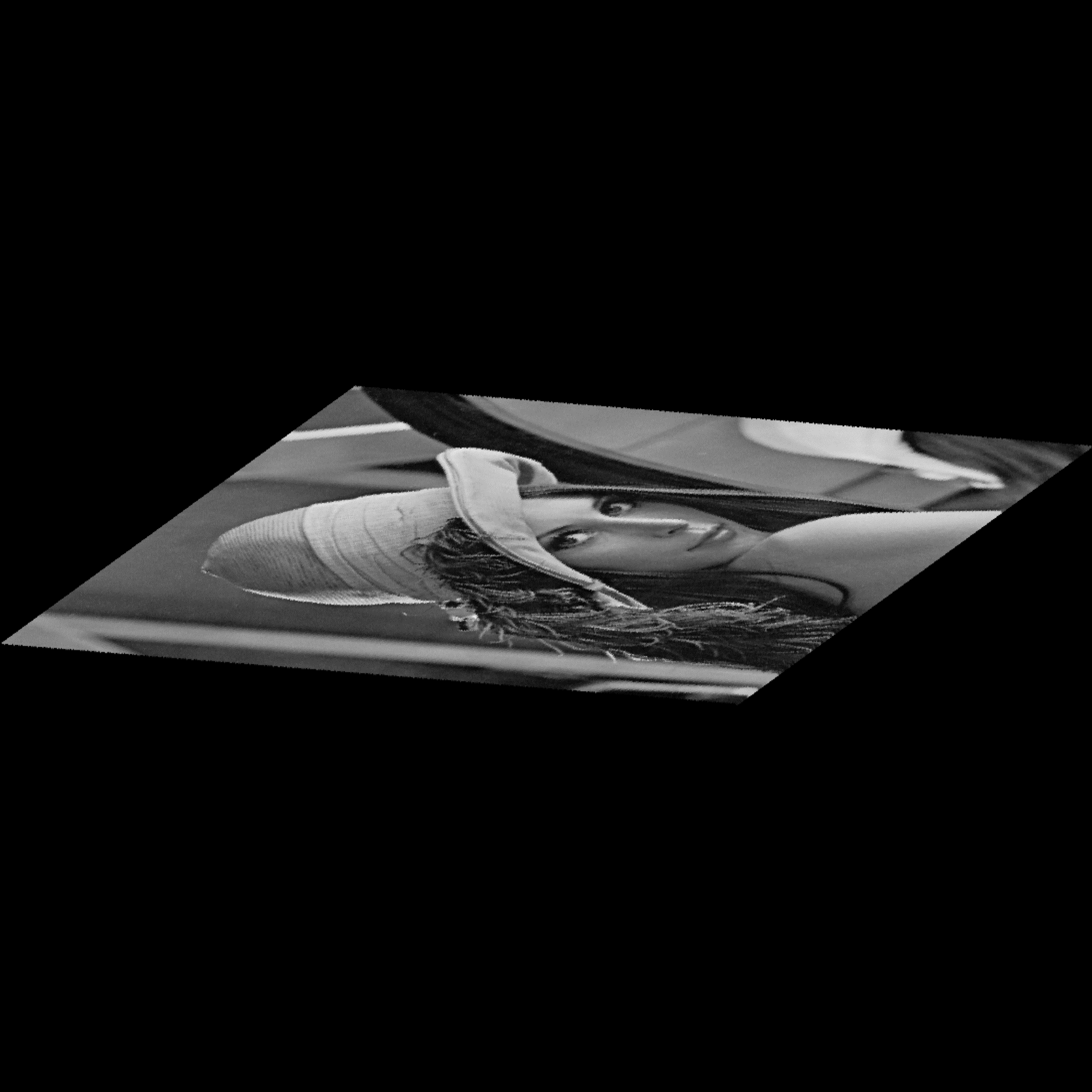
二.实验结果

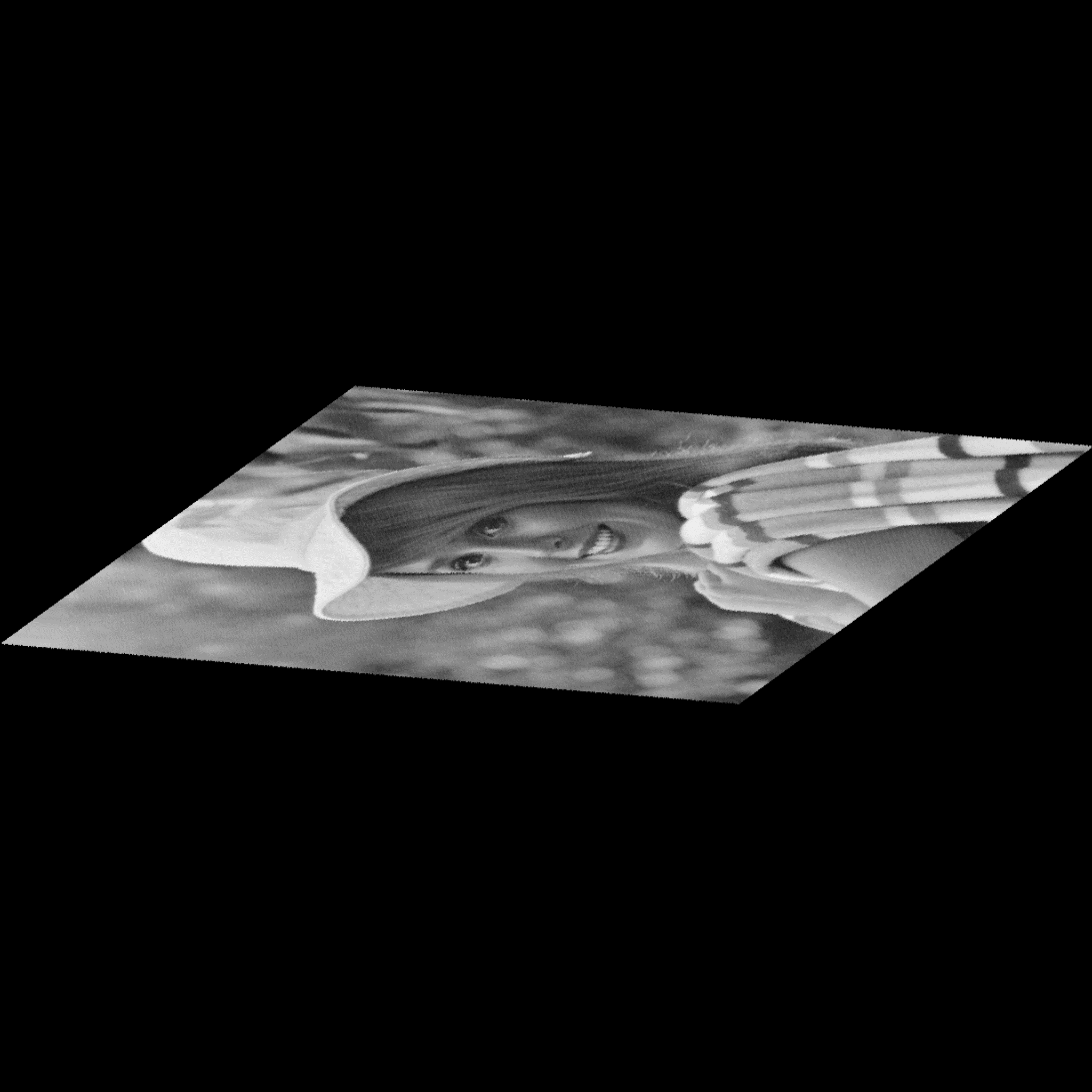
1.最邻近插值





2.双线性插值





3.双三次插值

