**一、实验目的和要求**

1.了解图像复原的基本流程；

2.熟练掌握利用python进行算术均值滤波的图像复原方法;

3.熟练掌握利用python进行几何均值滤波的图像复原方法。

4.熟练掌握利用python进行谐波均值滤波的图像复原方法;

5.熟练掌握利用python进行逆谐波均值滤波的图像复原方法。

**二、实验主要仪器和设备**

计算机;

Python与opencv库;

实验所需数据。

**三、实验方法和步骤**

1．打开计算机,启动Python运行环境;

2．采集实验所需图像和视频数据;

3．完成图像算术均值滤波处理,并显示运算结果;

4．完成图像算术均值滤波处理,并显示处理结果;

5．完成图像谐波均值滤波处理,并显示运算结果;

6．完成图像逆谐波均值滤波处理,并显示处理结果;

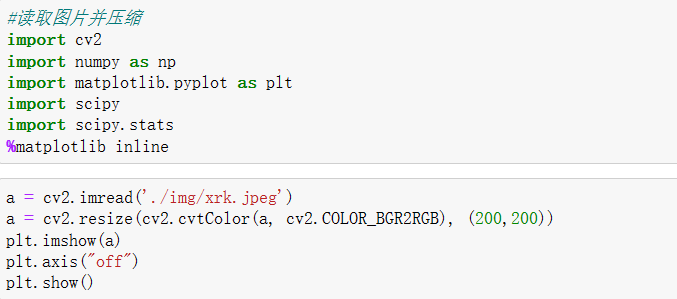
7．记录和整理实验报告。

**四、实验结果分析**

**（实验源代码、实验结果图、实验分析比较、结论）**

**1. 采集实验所需图像和视频数据；**

实验源代码：



实验结果图：



实验分析比较、结论：

这段代码是在Python中使用OpenCV、NumPy、Matplotlib和Scipy库来读取图片并压缩。代码的执行步骤如下：

（1）通过OpenCV中的cv2.imread()函数读取图片，这里的图片路径为'./img/xrk.jpeg'。

（2）使用cv2.cvtColor()函数将图片从BGR颜色空间转换为RGB颜色空间。

（3）使用cv2.resize()函数将图片的大小调整为(200, 200)像素，压缩图片大小。

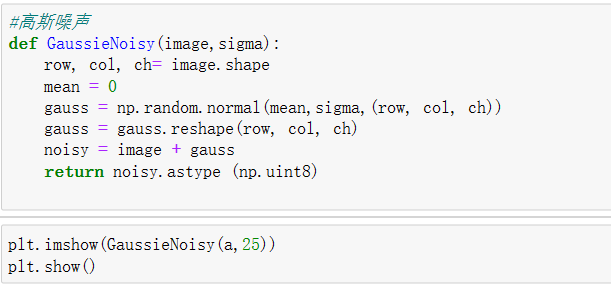
（4）使用Matplotlib库的plt.imshow()函数将压缩后的图片显示出来。

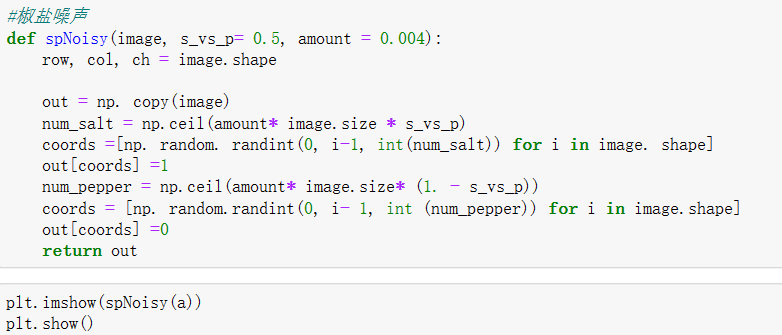
（5）使用plt.axis("off")关闭图片的坐标轴。

（6）使用plt.show()函数展示压缩后的图片。

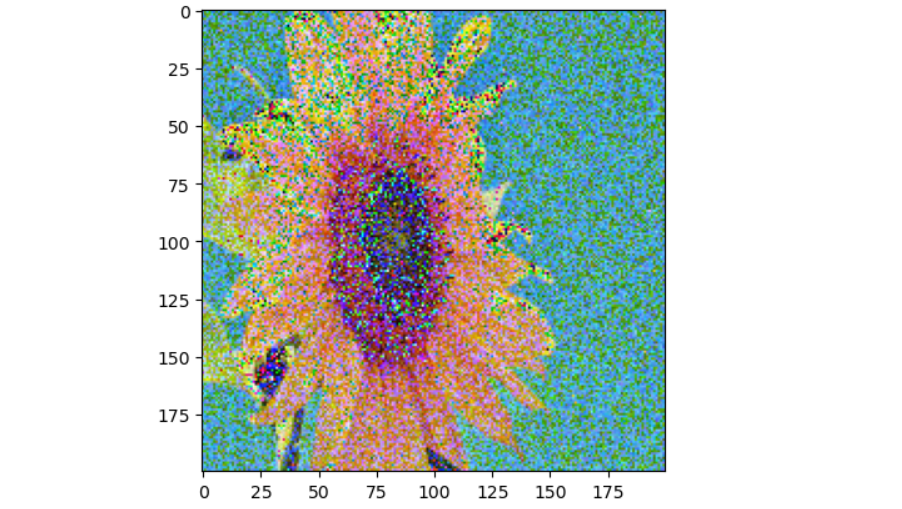
**2.图像加入噪声：**

实验源代码：

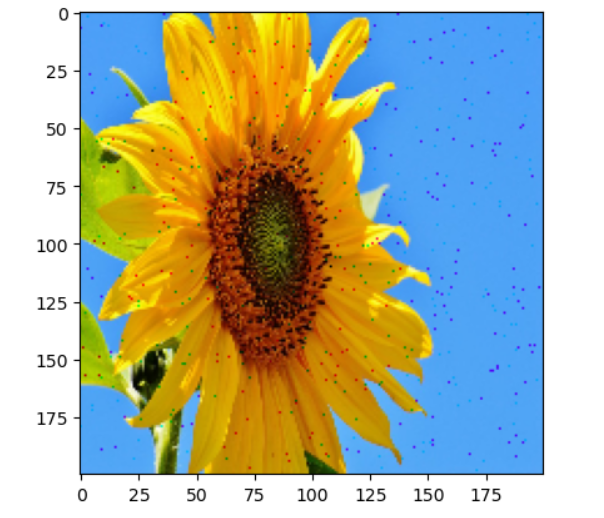




实验结果图：



**图1：高斯噪声**



**图2：椒盐噪声**

实验分析比较、结论：

第一段代码是定义了一个函数GaussieNoisy，用于为输入图像添加高斯噪声。代码的执行步骤如下：

（1）函数接收两个参数：一个是图像（image），一个是高斯噪声的标准差（sigma）。

（2）使用image.shape获取图像的行数、列数和通道数（如果是彩色图像）。

（3）设置高斯噪声的均值（mean）为0。

（4）使用NumPy库的np.random.normal()函数生成一个符合高斯分布的随机数数组，其均值为0，标准差为sigma，数组大小为(row, col, ch)，即与原图像大小和通道数相同。

（5）将生成的高斯噪声数组重塑为与原图像大小和通道数相同的形状，即(row, col, ch)。

（6）将原图像和生成的高斯噪声数组相加，得到添加高斯噪声后的图像（noisy）。

（7）将添加噪声后的图像的数据类型转换为无符号8位整数（np.uint8）。

添加高斯噪声会随机扰动图像中的像素值，使得图像看起来更加自然但是会损失一些图像信息。

第二段代码是定义了一个函数spNoisy，用于为输入图像添加椒盐噪声。代码的执行步骤如下：

（1）函数接收三个参数：一个是图像（image），一个是椒盐噪声中盐的比例（s\_vs\_p），一个是添加椒盐噪声的数量（amount）。

（2）使用image.shape获取图像的行数、列数和通道数（如果是彩色图像）。

（3）复制原始图像（image）为输出图像（out）。

（4）计算盐的数量：amount \* image.size \* s\_vs\_p，向上取整（np.ceil）。

（5）使用NumPy库的np.random.randint()函数生成一个随机坐标数组（coords），其大小为(int(num\_salt)， 3)，即盐的数量、图像的通道数和坐标数。

（6）将out中对应的坐标位置的像素值设置为1，即添加盐噪声。

（7）计算胡椒的数量：amount \* image.size \* (1. - s\_vs\_p)，向上取整（np.ceil）。

（8）使用np.random.randint()函数生成一个随机坐标数组（coords），其大小为(int(num\_pepper), 3)，即胡椒的数量、图像的通道数和坐标数。

（9）将out中对应的坐标位置的像素值设置为0，即添加胡椒噪声。

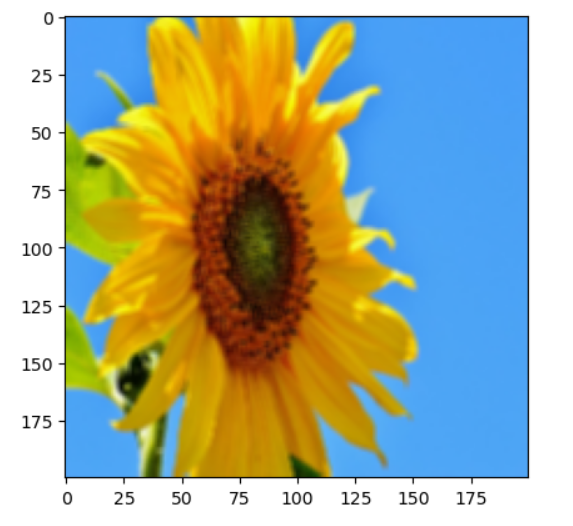
椒盐噪声是随机将像素点的值设为0或255，相当于随机生成黑色或白色的像素点，从而给图像带来随机的白点或黑点。添加椒盐噪声通常用于测试图像处理算法的鲁棒性。

**3. 图像算术均值滤波处理；**

实验源代码：



实验结果图：



实验分析比较、结论：

这段代码实现了算术均值滤波的过程，并对彩色图像进行了处理。具体分析如下：

（1）首先定义了一个函数ArithmeticMeanOperator(roi)，用于计算给定区域（roi）的像素值的平均值，并返回该平均值。

（2）然后定义了一个函数ArithmeticMeanAlogrithm(image)，该函数的作用是对输入的图像进行算术均值滤波处理。函数内部首先创建了一个新的零数组new\_image，大小与输入图像相同，用于存储滤波后的图像。接着将输入图像使用cv2.copyMakeBorder()函数进行边缘扩展，以避免滤波过程中超出边界的问题。

（3）然后通过双重循环遍历图像中的所有像素，对于每个像素，取以其为中心的3x3的区域（即邻域），将该区域的像素值传递给ArithmeticMeanOperator()函数计算平均值，并将该平均值赋值给新图像的对应像素位置。

（4）接着，将新图像的像素值进行归一化处理（使其值在0-255范围内），并返回处理后的图像。

（5）为了处理彩色图像，还定义了一个函数rgbArithmeticMean(image)，用于分别对输入图像的三个通道（R、G、B）进行算术均值滤波，然后将处理后的三个通道合并成一幅彩色图像，并返回该图像。

（6）最后，通过plt.imshow()函数将处理后的彩色图像显示出来。

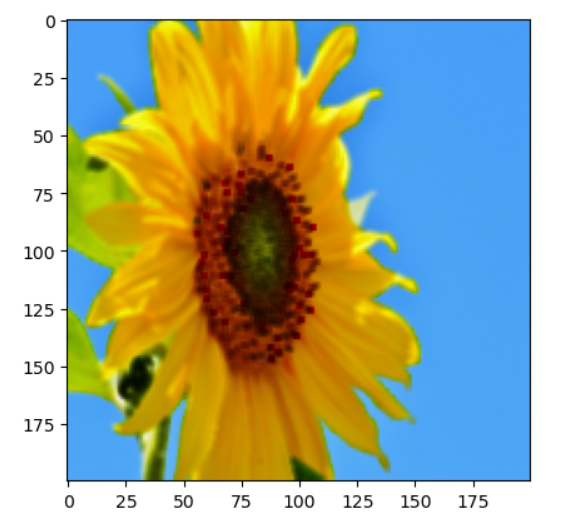
需要注意的是，算术均值滤波是一种基本的线性滤波方法，它是将每个像素的邻域内的像素值取平均值作为该像素的新值，从而实现去噪的效果。由于简单和易于实现，因此算术均值滤波是最常用的一种滤波方法之一。

**4.图像几何均值滤波处理：**

实验源代码：



实验结果图：



实验分析比较、结论：

这段代码实现了几何均值滤波算法，其中包括以下几个函数：

GeometricMeanOperator(roi)：计算给定 ROI 区域的几何平均值。

GeometricMeanAlogrithm(image)：对输入图像进行几何均值滤波。

rgbGemotriccMean(image)：将输入图像按通道分离后分别进行几何均值滤波，再将结果合并为一幅图像进行输出。

下面是对每个函数的具体分析：

（1）GeometricMeanOperator(roi)

该函数的输入参数为一个 ROI 区域，函数计算出该区域的几何平均值并返回。首先将 ROI 转换为 float64 数据类型，计算其像素值的累积积，最后根据 ROI 区域的大小计算出几何平均值并返回。

（2）GeometricMeanAlogrithm(image)

该函数的输入参数为一幅图像，首先将图像按照边框宽度为 1 进行扩展，接着遍历图像中的每个像素，将该像素周围的 3x3 区域作为 ROI 区域传入 GeometricMeanOperator 函数计算该区域的几何平均值，并将结果赋给新图像中对应像素的值。最后将新图像的像素值缩放至 0-255 之间并转换为 uint8 数据类型，并返回新图像。

（3）rgbGemotriccMean(image)

该函数的输入参数为一幅 RGB 彩色图像，首先将图像按通道分离，然后分别对每个通道进行几何均值滤波（即调用 GeometricMeanAlogrithm 函数），最后将各通道的结果合并为一幅图像并返回。

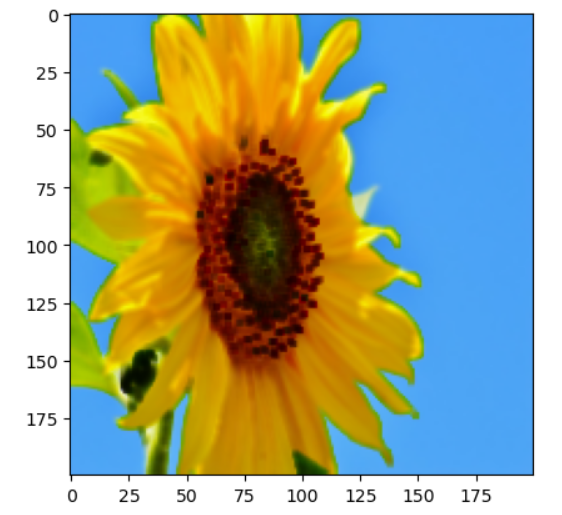
值得注意的是，函数GeometricMeanAlogrithm和rgbGemotriccMean中最后的 return 语句应该缩进一次，以保证返回值的正确性。

**5. 图像谐波均值滤波处理；**

实验源代码：



实验结果图：



实验分析比较、结论：

这段代码实现了谐波均值滤波算法，可以用于去除图像中的椒盐噪声。下面对代码进行分析：

（1）HMeanOperator函数：该函数实现了对给定ROI区域的谐波均值计算，其中对ROI中为0的像素进行特殊处理，避免出现除0错误。该函数使用了Scipy中的stats模块中的hmean函数，该函数可以计算数据集的谐波平均值。

（2）HMeanAlogrithm函数：该函数实现了对图像进行谐波均值滤波，具体思路是将图像中每个像素的RGB值替换为其周围3×3的ROI区域的谐波平均值，然后将处理后的图像输出。在实现过程中，对图像进行了边界扩充操作，避免对边界像素的处理出现错误。

（3）rgbHMean函数：该函数实现了对RGB图像进行谐波均值滤波，具体思路是将原始图像分离成三个通道，分别对每个通道进行HMeanAlogrithm操作，然后再将处理后的三个通道合并成一幅RGB图像输出。

最后利用Matplotlib中的plt.imshow函数对处理后的RGB图像进行可视化显示。

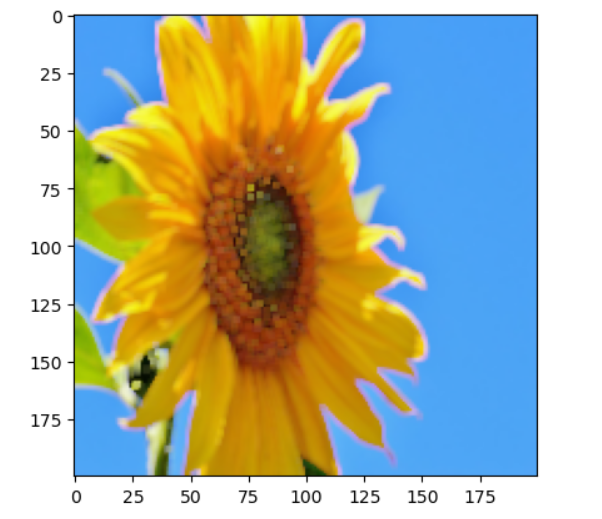
在这段代码中，由于处理过程中产生了浮点数结果，因此需要进行最大值最小值的归一化操作，并将结果转换为整型数据再进行输出。

**6. 图像逆谐波均值滤波处理；**

实验源代码：



实验结果图：



实验分析比较、结论：

这段代码实现了逆谐波均值滤波（Inverse Harmonic Mean Filter）。

（1）函数IHMeanOperator(roi, q)接受一个ROI（类似于图像中的一个小块），以及参数q，返回逆谐波均值。其中，ROI通过roi.astype(np.float64)转换为float64类型，然后分别计算ROI中每个像素的q+1次方和q次方的平均值，两者相除即为逆谐波均值。

（2）函数IHMeanAlogrithm(image, q)接受一幅图像和参数q，返回逆谐波均值滤波后的图像。首先使用cv2.copyMakeBorder()函数给原图像加上边框，然后遍历图像中的每个像素，取以该像素为中心的区域作为ROI，调用IHMeanOperator()函数计算逆谐波均值，并将计算结果保存到新图像中。

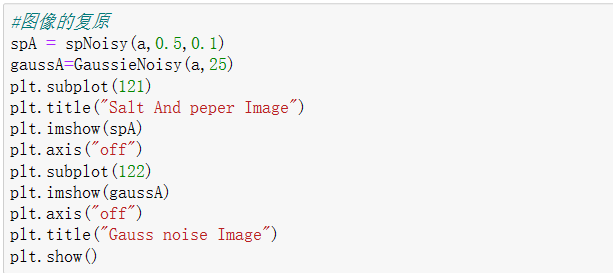
（3）函数rgbIHMean(image, q)接受一幅RGB图像和参数q，对RGB三个通道分别进行逆谐波均值滤波，并将结果合并成一个RGB图像返回。

最后，使用plt.imshow()函数显示处理后的RGB图像。

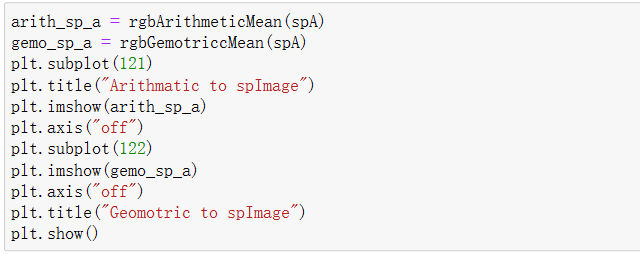
需要注意的是，该算法对于含有高斯噪声的图像效果较差，因为噪声会使ROI中出现较小的像素值，从而导致分母过小。

**7.**图像复原：

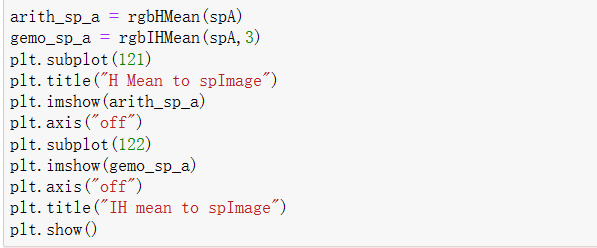
实验源代码：



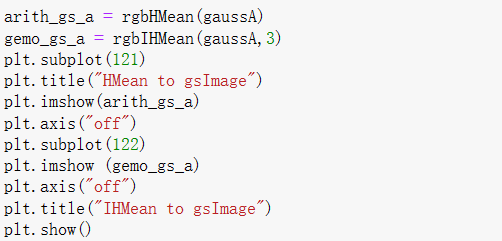
对椒盐噪声图像进行算数均值滤波复原与几何均值滤波复原：



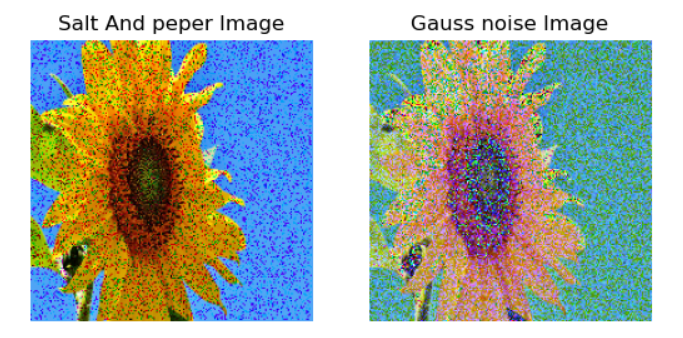
对椒盐噪声图像进行谐波均值滤波复原和逆谐波均值滤波复原：

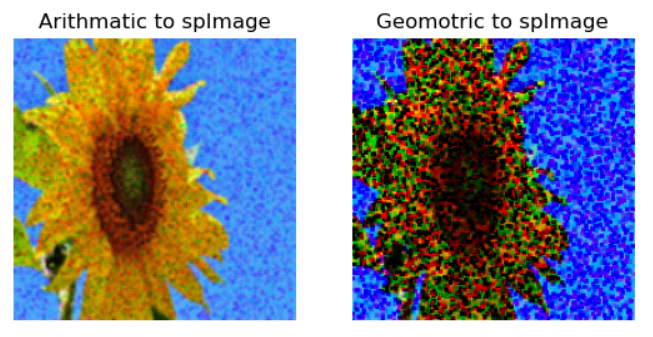


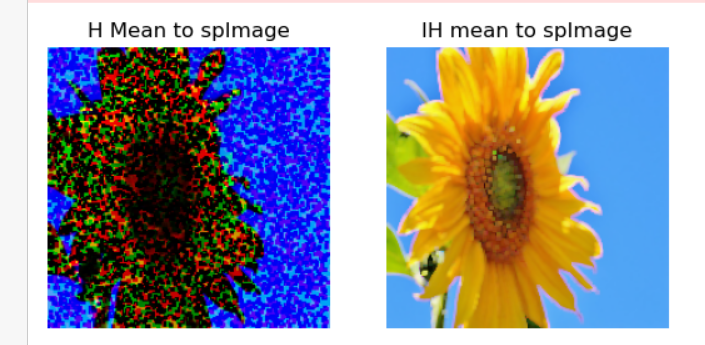
对高斯噪声图像进行谐波均值滤波复原和逆谐波均值滤波复原：

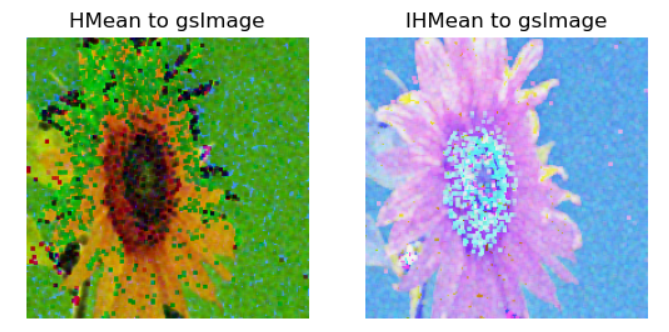


实验结果图：









实验分析比较、结论：

【1】第一段代码展示了两种不同类型的图像噪声：盐和胡椒噪声（salt and pepper noise）和高斯噪声（Gaussian noise）对图像的影响。

第一行代码生成了一个含有盐和胡椒噪声的图像，其中参数0.5是盐和胡椒噪声的密度，0.1是噪声的强度。

第二行代码生成了一个含有高斯噪声的图像，其中参数25是高斯噪声的标准差。

接下来的代码使用matplotlib库的subplot函数将两幅图像显示在同一个窗口中，其中第一个subplot显示含有盐和胡椒噪声的图像，第二个subplot显示含有高斯噪声的图像。最后的imshow和axis函数用于显示和设置图像的标题和轴。

【2】第二段代码对添加椒盐噪声的图像进行了算术平均滤波和几何平均滤波，并将两种处理结果显示在一个subplot中。

（1）rgbArithmeticMean()函数实现了RGB图像的算术平均滤波，即对每个像素点的RGB值分别取平均值，得到新的像素点RGB值，从而达到平滑图像的目的。

（2）rgbGemotriccMean()函数实现了RGB图像的几何平均滤波，即对每个像素点的RGB值分别取对数，再取平均数，最后求指数，得到新的像素点RGB值，从而达到平滑图像的目的。

【3】这段代码使用谐波均值滤波和逆谐波均值滤波对一张加了椒盐噪声的图像进行滤波处理，并可视化输出结果。具体分析如下：

首先使用spA = spNoisy(a,0.5,0.1)函数对原始图像 a 进行加椒盐噪声处理，生成加了椒盐噪声的图像 spA，同时使用gaussA=GaussieNoisy(a,25)函数对原始图像 a 进行高斯噪声处理，生成加了高斯噪声的图像 gaussA。

然后对spA 进行谐波均值滤波和逆谐波均值滤波处理，分别使用arith\_sp\_a = rgbHMean(spA)和gemo\_sp\_a = rgbIHMean(spA,3)函数，生成谐波均值滤波处理后的图像 arith\_sp\_a 和逆谐波均值滤波处理后的图像 gemo\_sp\_a。

最后使用plt.subplot()和plt.imshow()函数将谐波均值滤波和逆谐波均值滤波处理后的图像可视化输出。plt.axis("off")函数取消子图坐标轴显示。

综上，这段代码的作用是对加了椒盐噪声的图像进行谐波均值滤波和逆谐波均值滤波处理，并将处理后的图像可视化输出。

【4】这段代码主要用于对高斯噪声图像进行逆谐波均值滤波和谐波均值滤波。

首先，使用rgbHMean函数对高斯噪声图像进行逆谐波均值滤波，将结果保存在arith\_gs\_a变量中。

然后，使用rgbIHMean函数对高斯噪声图像进行谐波均值滤波，将结果保存在gemo\_gs\_a变量中。

最后，将两种滤波后的图像在同一个窗口中显示出来，左边是逆谐波均值滤波后的图像，右边是谐波均值滤波后的图像。

**五、实验总结与思考**

本次实验主要介绍了数字图像处理中的噪声去除方法，包括加性噪声和乘性噪声的去除方法。首先，通过添加不同强度的噪声来生成噪声图像，然后通过算术均值、几何均值、谐波均值、逆谐波均值等方法对图像进行去噪处理，最终展示了处理后的图像。

通过对比实验结果，可以发现不同的去噪方法在去除不同类型的噪声时表现出不同的效果。对于加性噪声，算术均值和几何均值的效果都不错，而对于乘性噪声，则需要使用谐波均值和逆谐波均值等方法。

综上所述，数字图像处理中的噪声去除方法是一个十分重要的课题，在实际应用中需要根据具体的噪声类型和噪声强度来选择合适的去噪方法。本次实验提供了一些基础的噪声去除方法，并为进一步深入学习和实践打下了基础。