**一、实验目的和要求**

（1）掌握灰度直方图的概念及其计算方法；

（2）熟练掌握直方图均衡化过程；

（3）熟练掌握傅里叶变换过程；

（4）熟练掌握频域滤波中低通、高通滤波过程；

（5）熟练掌握空域滤波中均值、中值、锐化等滤波过程。

**二、实验主要仪器和设备**

计算机

Python与OpenCV库

**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）完成图像直方图运算，并显示运算结果；

（4）完成直方图均衡化处理，并显示处理结果；

（5）完成图像傅里叶变换，并显示计算结果；

（6）完成频域滤波中低通、高通滤波过程；

（7）完成空域滤波中均值、中值、锐化等滤波过程；

（8）记录和整理实验报告。

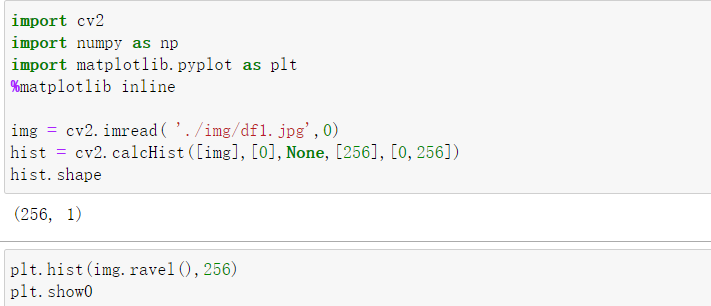
**四、实验结果分析**

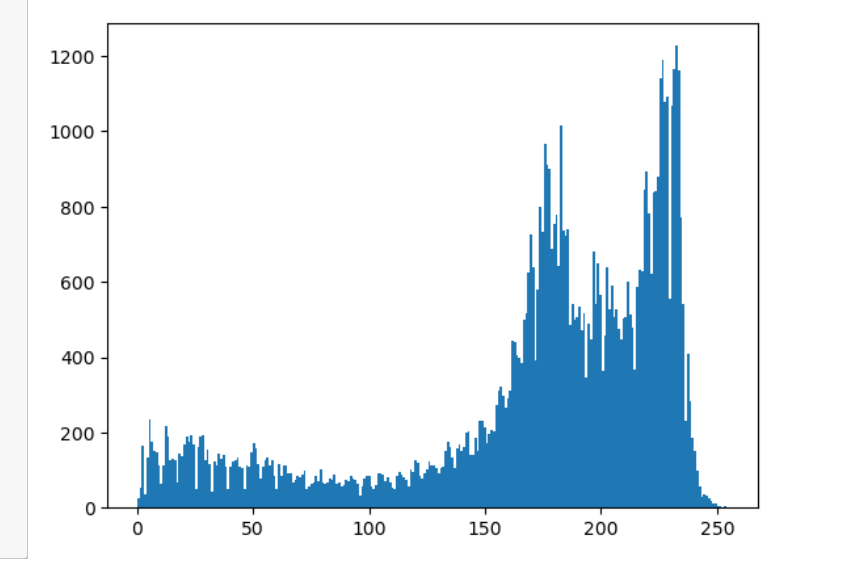
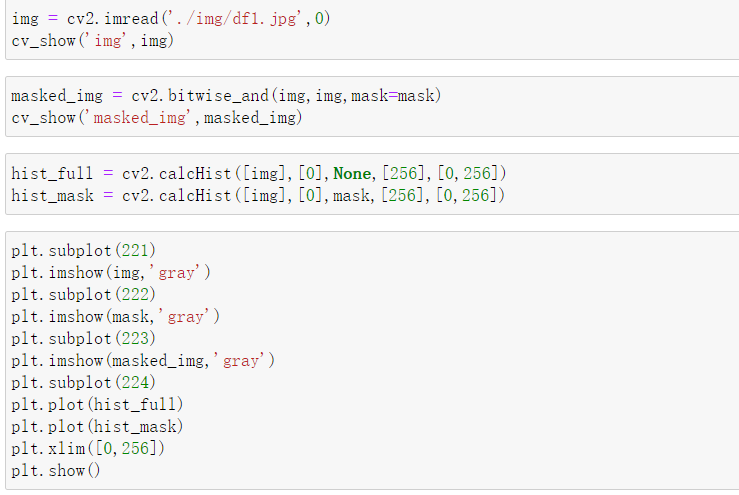
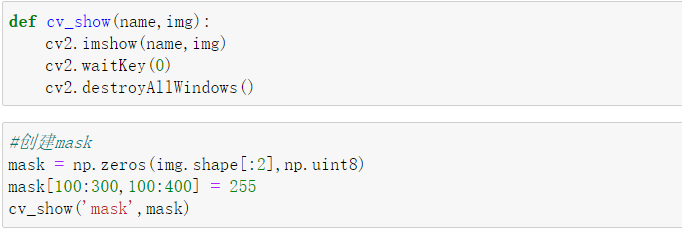
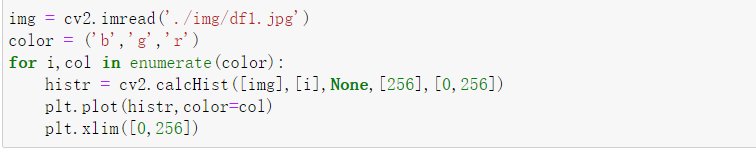
**（实验源代码、实验结果图、实验分析比较、结论）**

**1.图像直方图运算：**

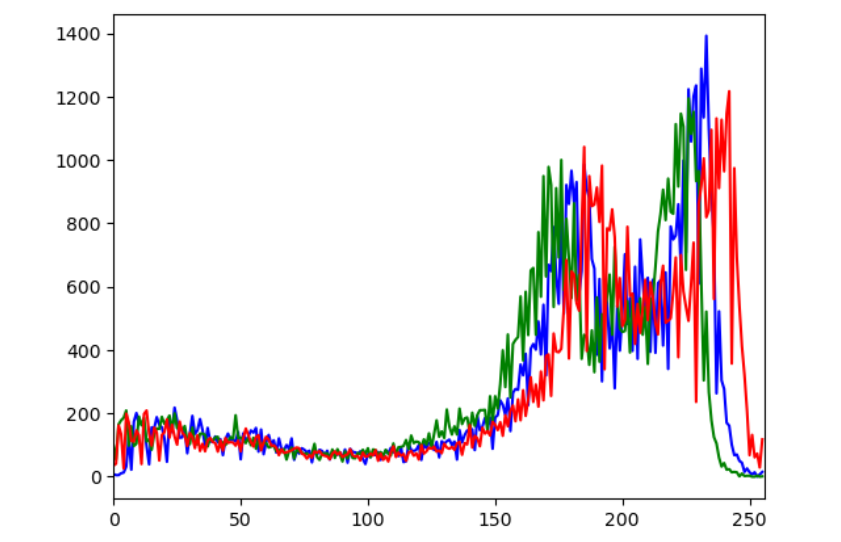
实验源代码：

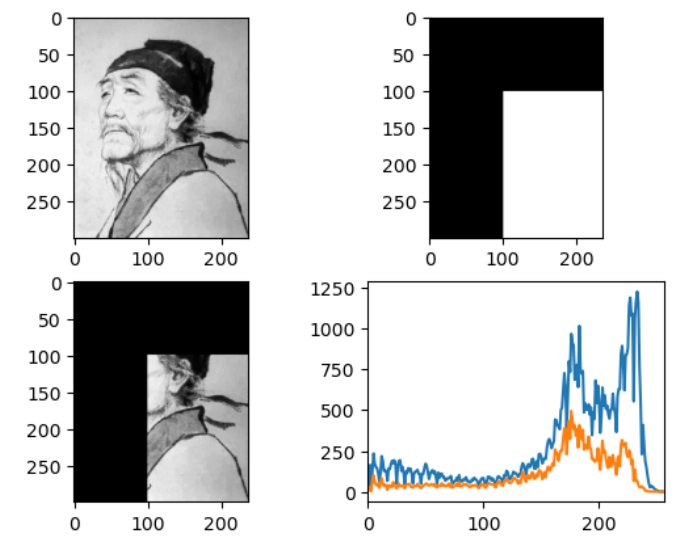
绘制图像像素点直方图：





**图像像素点直方图**





实验分析比较、结论：

这段代码主要实现了以下几个功能：

（1）读取图像：使用cv2.imread函数读取一张灰度图像，并显示直方图。

（2）计算直方图：使用cv2.calcHist函数计算图像的灰度直方图，并使用plt.hist函数显示直方图。

（3）绘制颜色直方图：使用cv2.calcHist函数计算图像的颜色直方图，并使用plt.plot函数绘制三个通道的颜色直方图。

（4）创建掩膜：使用np.zeros函数创建一个和图像大小相同的全零矩阵，然后在矩阵的指定位置赋值为255，以创建一个矩形掩膜。

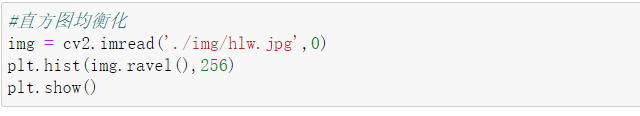
（5）显示图像：使用cv\_show函数显示原始图像、掩膜和掩膜后的图像。

（6）计算带掩膜的直方图：使用cv2.calcHist函数计算原始图像和掩膜后的图像的灰度直方图，并使用plt.subplot函数将原始图像、掩膜、掩膜后的图像和两个直方图显示在一个窗口中。

总体来说，这段代码展示了如何使用OpenCV和Matplotlib库对图像进行处理和分析。其中包括读取、显示、计算直方图、创建掩膜等基本操作。同时，该代码还演示了如何使用Matplotlib库绘制图像和直方图，以及如何使用掩膜来限制对图像的处理。

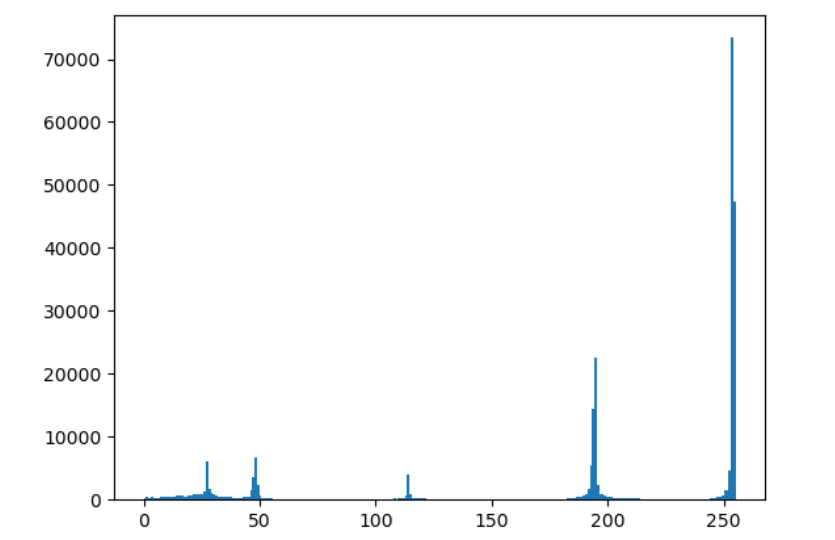
**2.图像直方图均衡化处理：**

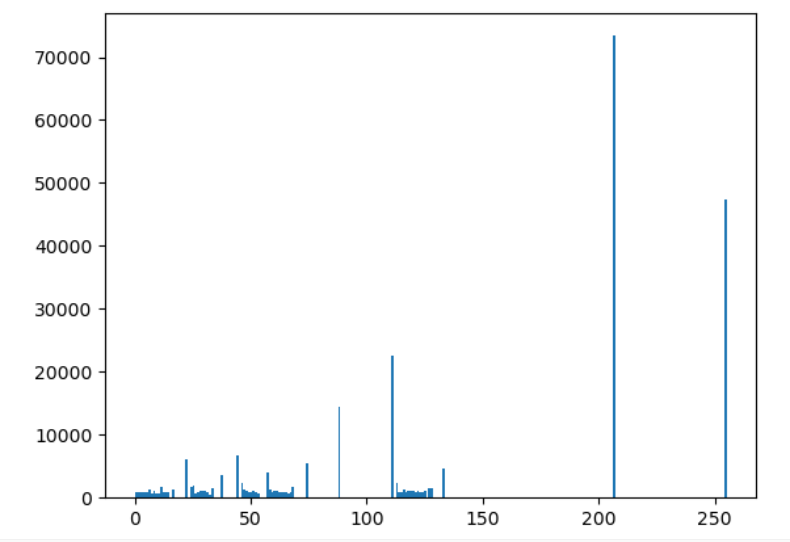
实验源代码：





实验效果图：







实验分析比较、结论：

这段代码展示了如何使用OpenCV对图像进行直方图均衡化的操作。代码主要实现了以下几个步骤：

（1）使用cv2.imread函数读取一张灰度图像，并使用plt.hist函数绘制该图像的灰度直方图。

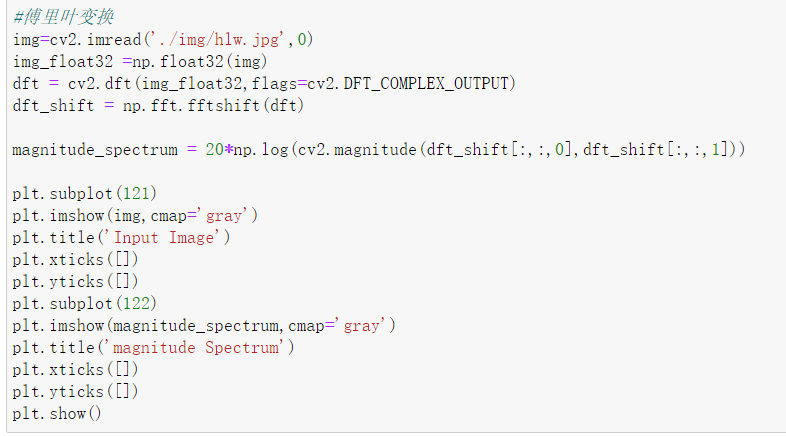
（2）使用cv2.equalizeHist函数对图像进行直方图均衡化，并使用plt.hist函数绘制均衡化后的直方图。直方图均衡化可以使图像的像素值更加均匀，从而提高图像的对比度和亮度。

（3）使用np.hstack函数将原始图像和均衡化后的图像在水平方向拼接，并使用cv\_show函数显示结果图像。

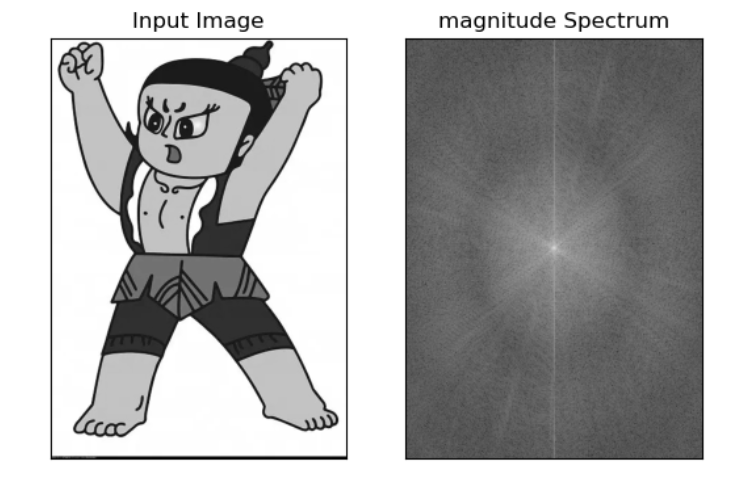
总体来说，这段代码是一个简单的图像增强示例，它展示了如何使用直方图均衡化方法来增强图像的对比度和亮度。

**3.图像傅里叶变换：**

实验源代码：



实验效果图：



实验分析比较、结论：

这段代码展示了如何使用OpenCV和NumPy对图像进行傅里叶变换并绘制频率谱的操作。代码主要实现了以下几个步骤：

（1）使用cv2.imread函数读取一张灰度图像，并使用np.float32将图像转换为浮点型数据。

（2）使用cv2.dft函数对图像进行傅里叶变换，并使用np.fft.fftshift函数将频率域的直流分量移到频谱的中心。

（3）计算频谱的幅度谱，使用cv2.magnitude函数计算幅度谱的值，并使用20\*np.log函数对值进行对数变换。

（4）使用plt.subplot函数创建一个2x1的子图窗口，分别绘制原始图像和幅度谱图像。在绘制幅度谱图像时，使用cmap='gray'参数设置为灰度图像。

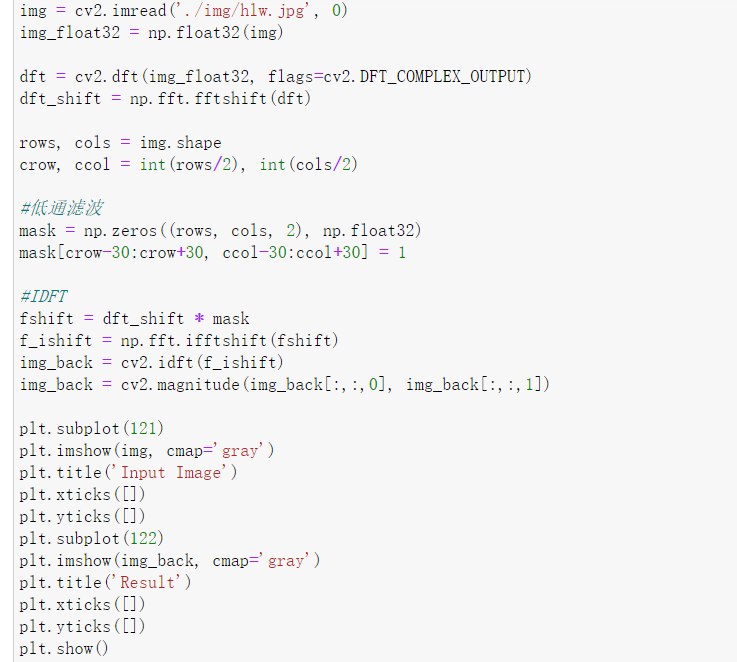
（5）使用plt.xticks和plt.yticks函数去除坐标轴刻度。

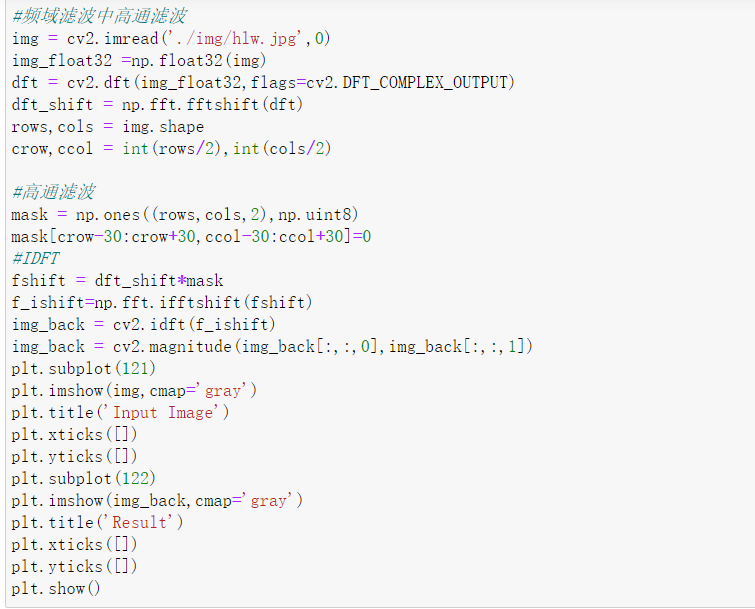
总体来说，这段代码是一个简单的傅里叶变换，它展示了如何将图像转换到频域，并绘制幅度谱图像。

**4.完成频域滤波中低通、高通滤波过程：**

实验源代码：

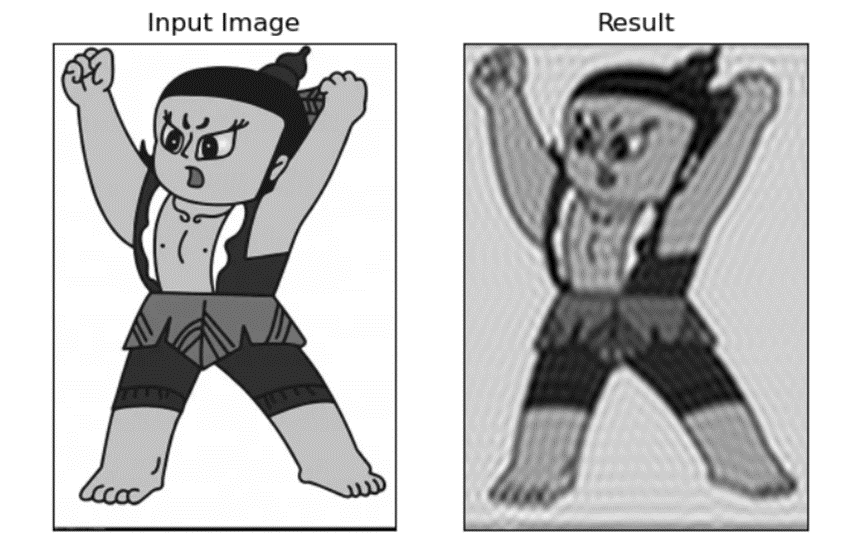
低通滤波：



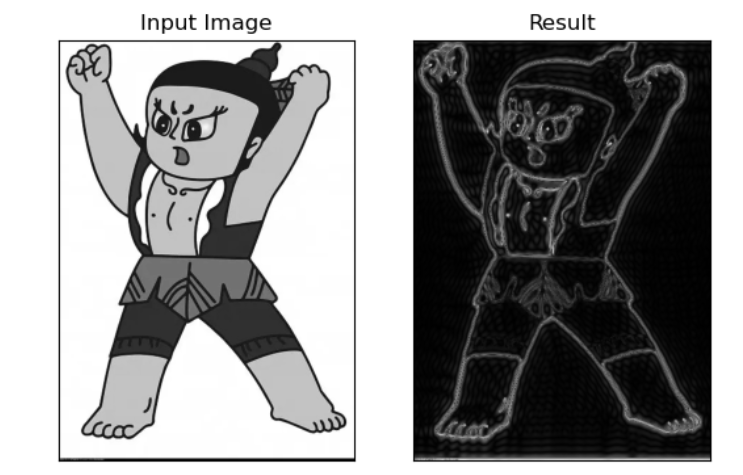


实验效果图：

低通滤波：



高通滤波：



实验分析比较、结论：

【1】这段代码实现了傅里叶变换下的频域滤波中的低通滤波。具体分析如下：

（1）读入一张灰度图像hlw.jpg。

（2）将图像转换成浮点型并进行傅里叶变换，得到其复数形式的频域表示dft。

（3）将频域表示进行移位，使得低频中心在频谱图像的中心位置。

（4）根据图像的大小，找到图像中心点的坐标(crow,ccol)。

（5）创建一个大小和输入图像一样的全零数组mask，在其中心区域设为1，实现低通滤波。

（6）将mask与频域表示dft\_shift相乘，得到低通滤波后的频域表示fshift。

（7）将fshift进行反移位，得到原始频域表示dft。

（8）对dft进行反傅里叶变换，得到空域的滤波结果img\_back。

（9）显示原始图像和滤波结果。

可以看出，低通滤波通过在频域中心附近设置1值区域，实现了对低频分量的保留，对高频分量的抑制。该滤波器可用于去除图像中高频噪声，平滑图像。

【2】这段代码是进行频域滤波中的高通滤波，主要步骤如下：

（1）读入灰度图像；

（2）进行傅里叶变换，并进行频谱中心化；

（3）定义高通滤波掩码，即中心为1，周围为0的矩阵；

（4）将滤波掩码和傅里叶变换后的图像进行乘积；

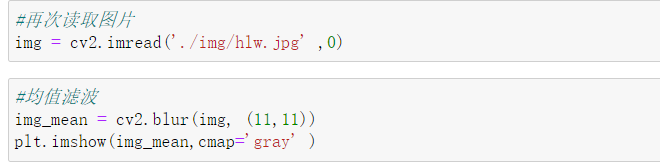
（5）进行傅里叶反变换，并取幅值，得到滤波后的图像；

（6）绘制原图和滤波后的图像。

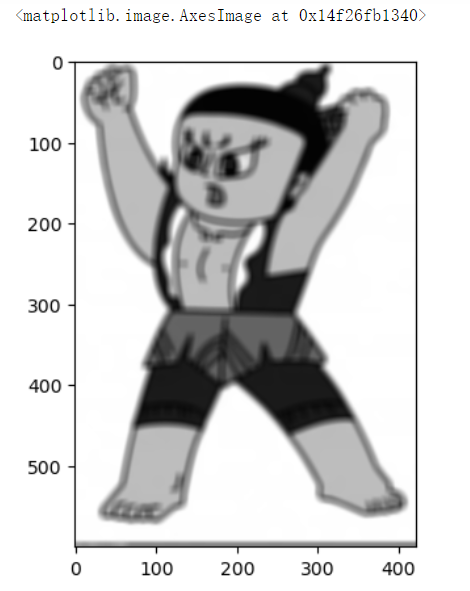
与低通滤波相比，高通滤波的滤波掩码取值相反，即中心为1，周围为0，这意味着在频域中去除低频分量，保留高频分量。这里的高通滤波掩码大小为60x60，即去除中心60x60的低频分量。滤波后的图像呈现出边缘、细节等高频特征更明显的效果。

**5.完成空域滤波中均值、中值、锐化等滤波过程：**

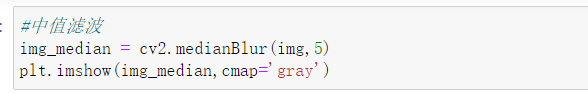
均值滤波实验源代码：



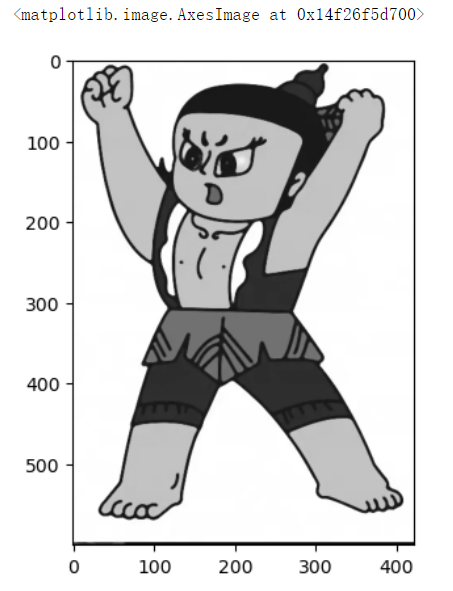
均值滤波实验效果图：



中值滤波实验源代码：



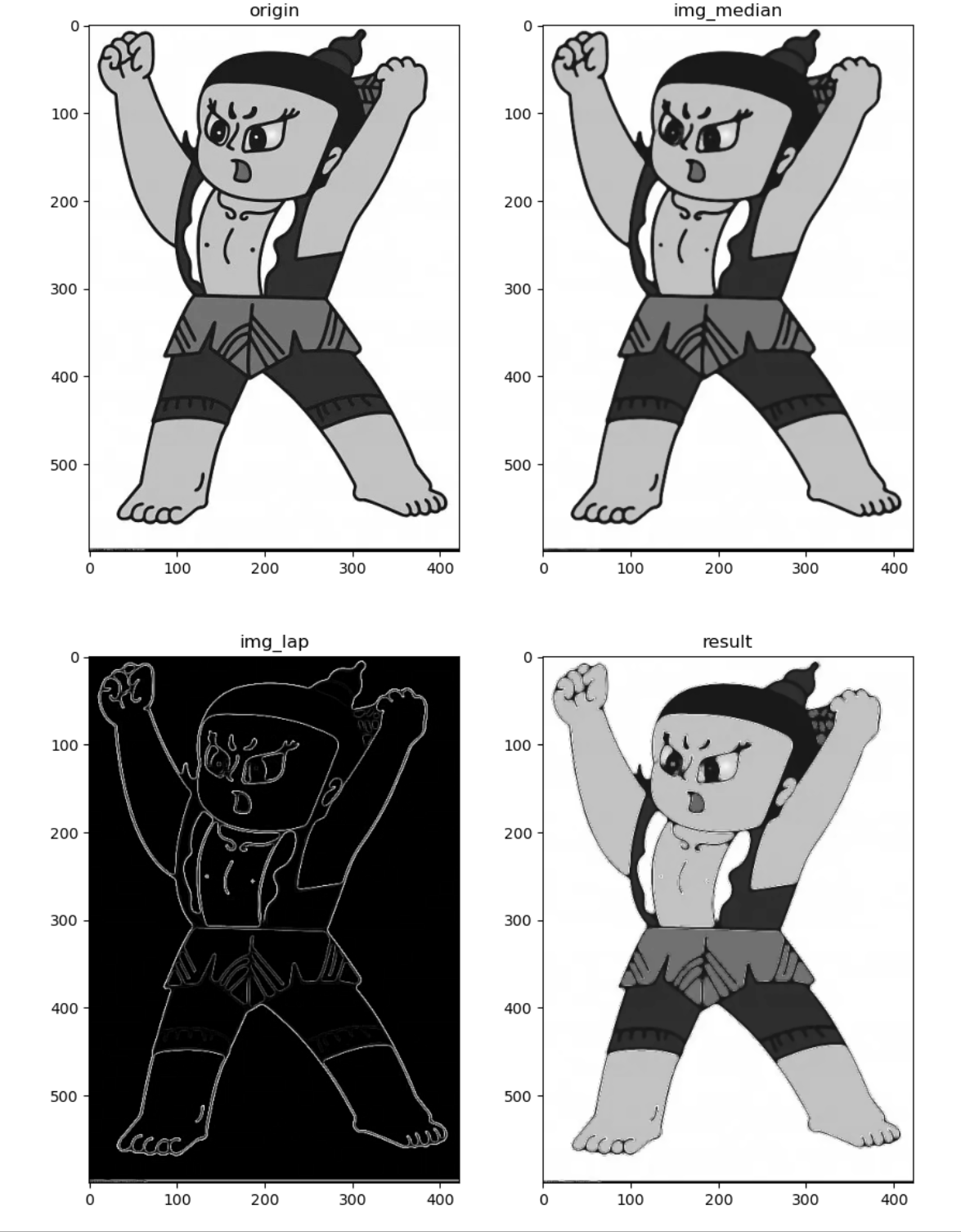
中值滤波实验效果图：



空域滤波中锐化源代码：



锐化实验效果图：



实验分析比较、结论：

第一个代码是进行均值滤波，使用了OpenCV中的cv2.blur()函数，该函数接受两个参数：输入图像和卷积核大小。在这里，输入图像是img，卷积核大小为(11,11)。均值滤波是一种线性滤波器，通过对图像中每个像素周围的像素进行求平均值的操作，来实现平滑处理，以去除图像中的噪声。

第二段代码实现了中值滤波，可以用来消除图像中的噪声。具体实现步骤如下：

（1）读取图像：通过cv2.imread()函数读取灰度图像。

（2）中值滤波：使用cv2.medianBlur()函数进行中值滤波，其中第二个参数指定了滤波器的大小（必须是奇数）。

（3）显示图像：使用plt.imshow()函数显示滤波后的图像，其中cmap='gray'指定了灰度图像。

与均值滤波相比，中值滤波更适合用于处理图像中的脉冲噪声或椒盐噪声等随机噪声，因为它可以更好地保留图像的边缘信息，同时不会使图像模糊。

第三段代码，是一段实现拉普拉斯锐化处理的代码。该代码的具体实现过程如下：

（1）定义一个Laplacian核，使用cv2.filter2D函数将该核与经过中值滤波处理的图像进行卷积，得到拉普拉斯锐化后的图像img\_lap。

（2）设定阈值threshold，将img\_lap中大于阈值的像素值保留，小于阈值的像素值置为0，得到img\_lap\_thresh。

（3）将中值滤波后的图像img\_median与img\_lap\_thresh相加，得到最终的锐化处理结果result。

（4）使用matplotlib库中的subplot和imshow函数将原始图像、中值滤波处理后的图像、拉普拉斯锐化处理后的图像以及最终的锐化处理结果在一个窗口中展示。

总体来说，该代码实现了一种简单的图像锐化处理方法，通过对经过中值滤波处理后的图像进行拉普拉斯锐化处理，可以增强图像的边缘信息，使图像看起来更加清晰锐利。

**五、实验总结与思考**

本次实验我们学习了数字图像处理中的常见滤波技术，包括均值滤波、高斯滤波、中值滤波、拉普拉斯锐化处理等。通过实践代码的演示，我们对这些滤波技术的原理、使用方法及效果进行了深入了解。

均值滤波可以平滑图像并消除噪声，但同时也会导致图像失真。高斯滤波则是一种常用的平滑滤波方法，可以有效地平滑图像，但也会导致图像细节损失。中值滤波能够有效地去除椒盐噪声等离群点，而不会损失图像细节。拉普拉斯锐化处理可以增强图像的轮廓和边缘，但也容易引起图像的噪声和失真。

在实践中，我们还了解了OpenCV库中的一些常用函数，例如cv2.filter2D()进行卷积滤波、cv2.medianBlur()进行中值滤波、cv2.add()实现图像加法等。同时，我们也学习了一些常用的可视化工具，例如matplotlib库中的imshow()和subplot()函数，用于显示和排列图像。

最后，我们还应用这些技术进行图像处理实践，例如去除噪声、增强图像细节、增强图像轮廓等。通过实践，我们深入理解了这些滤波技术的应用场景和实际效果，也能更好地应用这些技术进行数字图像处理。