图像处理作业-刘继东

# 第一周作业：打开图像，显示图像，存储图像，对一张图片进行缩放，观察其分辨率，降低灰度处理

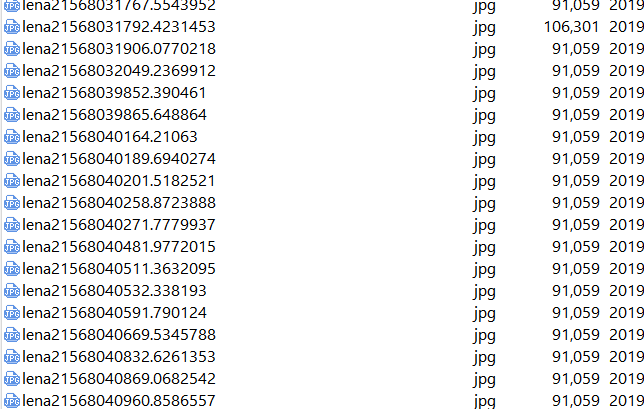
处理步骤：

1. 打开一副jpg文件，读取head部分，获得hight和width，然后读取data中的数据，将data数据放置到三维数组中，每一个点有r，g，b三个数值标识该点的颜色。
2. 使用opencv的imshow显示图片，imsave保存图片，
3. 使用opencv的resize缩放图片，放大或缩小，显示图片
4. 降低灰度级：对灰度图像的每个点的像素值/4,从8bit位降低到6bit位（即从256级降低到64灰度级）

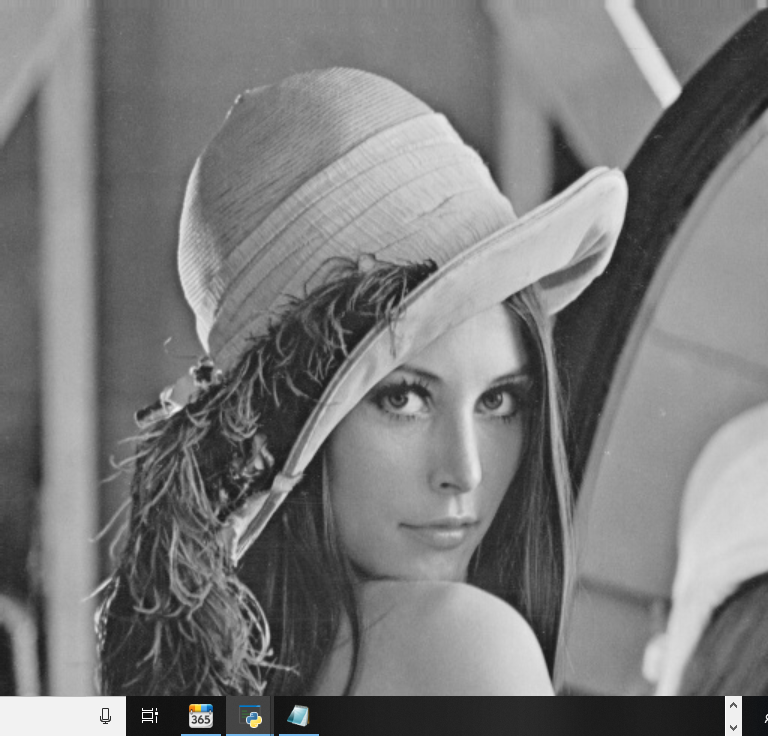
显示图片：



存储：增加时间戳作文文件名，保存如下：



放大1.5倍：



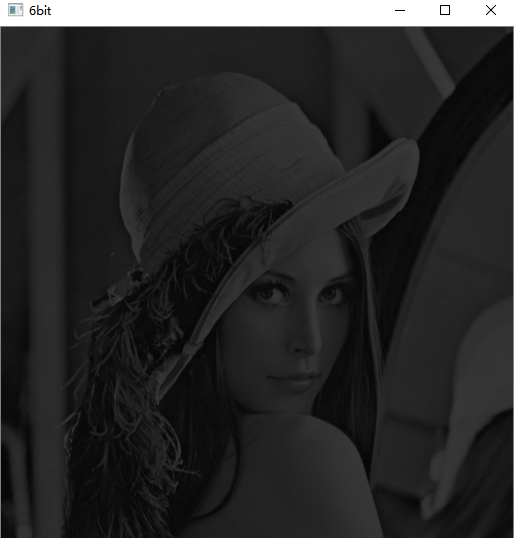
缩小0.5倍：



降低灰度值（每个像素灰度值-100）：



降低灰度级，6bit



源码：

###2019年9月9日 第一周作业：打开图像，显示图像，存储图像，对一张图片进行缩放，观察其分辨率，降低灰度处理  
import cv2 as cv  
import time  
import numpy as np  
  
img = cv.imread("./source/lena.jpg",cv.IMREAD\_GRAYSCALE) #读入做灰色处理  
print(img.shape) #打印大小  
(height,width) = img.shape  
  
cv.imshow("lena",img) #显示  
rename = str(time.time())  
print(rename,height,width)  
cv.imwrite("./dest/lena2"+rename+".jpg",img) #存错  
  
#放大一倍  
largedim = (int(height\*1.5),int(width\*1.5))  
img\_enlarge=cv.resize(img,largedim) #1data 2 (高，宽)  
cv.imshow("large",img\_enlarge);  
print(img\_enlarge.shape)  
  
###缩小一半,方法一  
img\_enMin=cv.resize(img,None,0.5,0.5,cv.INTER\_AREA) #1data 2 (高，宽)  
cv.imshow("minor1",img\_enMin);  
print(img\_enMin.shape)  
  
###缩小一半,方法二  
mindim = (int(height\*0.5),int(width\*0.5))  
img\_enMindim=cv.resize(img,mindim,cv.INTER\_AREA) #1data 2 (高，宽)  
cv.imshow("minor2",img\_enMindim);  
print(img\_enMindim.shape)  
  
  
##@改变每个像素的灰度值，每个像素降低100, 生成同大小的二维数组  
subval = np.full(img.shape,100,dtype=np.uint8)  
subimg = cv.subtract(img,subval)  
cv.imshow("subimg",subimg)

###降低灰度级数8bit图像降为6bit。  
destImg = img//4  
cv.imshow("6bit",destImg )  
print(destImg)  
  
cv.waitKey(0)

# 第二周作业：对低对比度的图像进行拉伸和用直方图均值化进行优化

处理步骤：

1. 使用伽马变换进行拉伸：

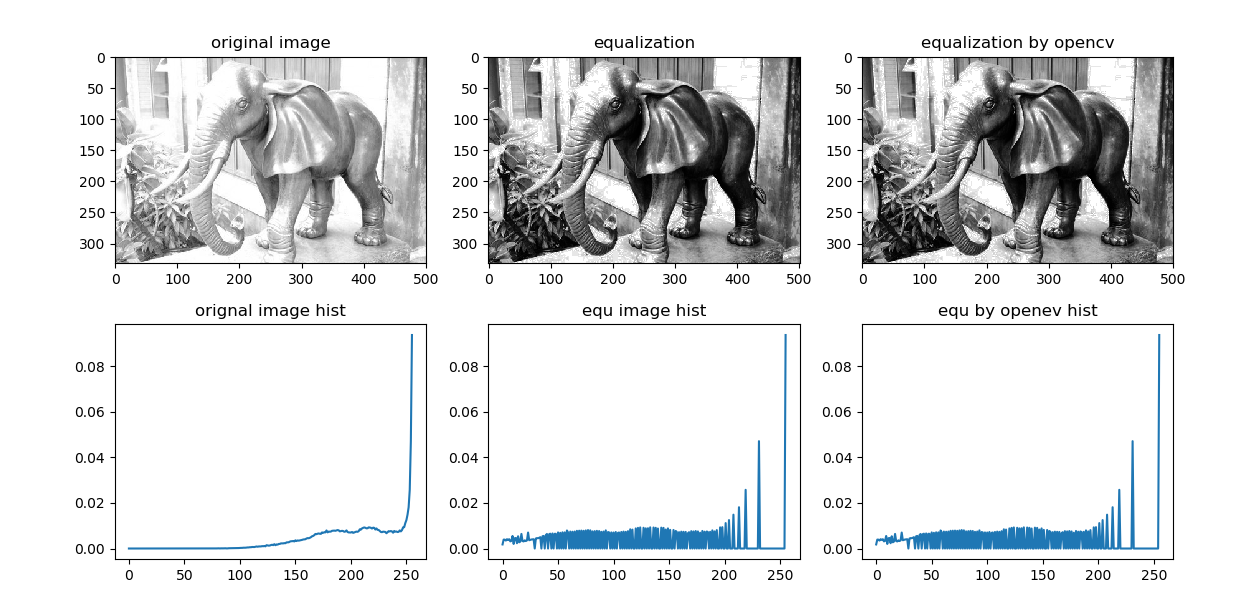
选取一幅对比度图像偏亮（曝光过度）的照片，将其变暗，γ>1，分别取2，3，4，5进行拉伸变换，效果如下。结果完成后，与skimage的伽马变换函数进行效果比对，基本相同。根据显示效果，当γ=4时效果最佳，实验结束。

1. 对图像进行直方图均衡优化

首先计算p(rk),然后求出累计直方图sk，并进行取整操作skInt。建立转换函数s=skInt[rk]然后使用map函数，对原像素值进行转换。效果如下。使用opencv的直方图均衡方法，比较两者差异，效果基本相同，实验结束。

对比度拉伸

对图像进行直方图均衡优化



代码如下：

###2019年9月14日 打开一幅低对比度图像，拉伸其图像，直方图均衡。  
#使用skimage开发包 和 根据原理自行开发，并对比显示其效果。  
import cv2 as cv  
import numpy as np  
import matplotlib.pyplot as plt  
from skimage import exposure  
  
  
img = cv.imread("./source/02.lowcontrast.jpg",cv.IMREAD\_GRAYSCALE) #读入做灰色处理  
print(img.shape)  
  
  
  
#使用skimage的伽马变换  
gamma\_img = exposure.adjust\_gamma(img, 5)  
#根据原理开发的伽马变换：对图像拉伸，先需做归一化(/255去除量纲），然后进行变换，最后复原回图像空间（需做无符号转换）  
#曝光过度，调黑。尝试γ取2，3，4，5，检查效果  
stretch2 = np.uint8(np.power(img / 255.0, 2) \* 255.0)  
stretch3 = np.uint8(np.power(img/255.0,3)\*255.0)  
stretch4 = np.uint8(np.power(img/255.0,4)\*255.0)  
stretch5 = np.uint8(np.power(img/255.0,5)\*255.0)  
print(stretch2)  
#对比skimage与自行开发的伽马变换。  
#cv.imshow("dev and skimage",np.hstack((stretch15,gamma\_img)))  
figure,axs =plt.subplots(2,3)  
axs[0,0].set\_title("original image")  
axs[0,0].imshow(img,cmap='gray') #原图  
  
axs[0,1].set\_title("stretch 2")  
axs[0,1].imshow(stretch2, cmap='gray')  
  
axs[0,2].set\_title("stretch 3")  
axs[0,2].imshow(stretch3,cmap='gray')  
  
axs[1,0].set\_title("stretch 4")  
axs[1,0].imshow(stretch4,cmap='gray')  
  
axs[1,1].set\_title("stretch 5")  
axs[1,1].imshow(stretch5,cmap='gray')  
  
axs[1,2].set\_title("package skimage stretch 5")  
axs[1,2].imshow(gamma\_img,cmap='gray')  
  
  
#skimage的直方图均衡函数  
equ = cv.equalizeHist(img)  
#cv.imshow('equalization by opencv', np.hstack((img, equ))) # 并排显示  
  
#按照公式计算：  
#原始图像灰度级统计  
rk = np.bincount(img.ravel(), minlength=256) #图像二维数组先转换为一维，然后使用bincount求出不同像素级数的个数  
prk=rk/img.size #各级像素的概率  
sk=np.zeros(256)  
pos = 0  
sum = 0.0  
for val in prk:  
 sum = sum + val #累计  
 sk[pos] = sum  
 pos = pos+1  
  
#展示处理后的直方图  
#plt.figure()  
#plt.plot(sk)  
  
print(sk,pos,sum)  
  
#sk取整扩展  
skInt = np.uint8(255\*sk+0.5)  
  
print(skInt)  
  
def fn\_change(x):  
 return skInt[x]  
  
equImg2=np.array(list(map(fn\_change,img)),dtype=np.uint8)  
equFigure,equAxs =plt.subplots(2,3)  
equAxs[0,0].set\_title("original image")  
equAxs[0,0].imshow(img,cmap='gray') #原图  
  
equAxs[0,1].set\_title("equalization")  
equAxs[0,1].imshow(equImg2, cmap='gray')  
  
equAxs[0,2].set\_title("equalization by opencv")  
equAxs[0,2].imshow(equ, cmap='gray')  
  
#展示图像直方图  
# plt.hist(img.ravel(), 256, [0, 256])#等价于 bincount先求出每个级数的个数，然后画直方图。  
hist = np.bincount(img.ravel(), minlength=256)/img.size  
equAxs[1,0].set\_title("orignal image hist")  
equAxs[1,0].plot(hist) #原图  
  
hist2 = np.bincount(equImg2.ravel(), minlength=256)/img.size  
equAxs[1,1].set\_title("equ image hist")  
equAxs[1,1].plot(hist2)  
  
hist3 = np.bincount(equ.ravel(), minlength=256)/img.size  
equAxs[1,2].set\_title("equ by openev hist")  
equAxs[1,2].plot(hist3)   
  
plt.show()  
  
cv.waitKey(0)

# 第三周作业：对一副图像进行加噪声，平滑，锐化处理。

处理步骤：

1.噪声处理：

1）.高斯噪声：生成正态分布的随机数，期望为0，方差=0.04（方差太大，图像噪声太多，太小，噪声不明显），矩阵大小与图像大小相同；随机矩阵+原图（归一化）后范围限制在0~1之间，然后进行还原为0~255的灰度值。

2）.添加椒盐噪声：产生x%的椒盐噪声：设总的像素点m，则x%的椒盐量，总共需要点数量为n= m\*x; 随机生成n个点坐标，随机设置该处的值为0或255.

2.平滑处理：

对椒盐噪声图像使用均值平滑与中值平滑

3.锐化处理：

梯度法为：newPixel = abs(fn[i, j+1] - fn[i, j]) + abs( fn[i+1, j] - fn[i, j])

[-1, 1],

[1, 0]

roberts: newPixel = abs(fn[i+1,j+1] - fn[i,j]) + abs(fn[i+1,j] - fn[i, j+1])

[-1, -1]

[1, 1]

sobel算法：检测边缘较平滑，光洁

newPixel = max(HorizontalDifference , VerticalDifference)

HorizontalDifference = abs( fn[

结构：horizontalDifference：

[-1, 0, 1]

[-2, 0, 2]

[-1, 0, 1]

verticalDifference： 将HorizontalDifference 顺时针旋转90度

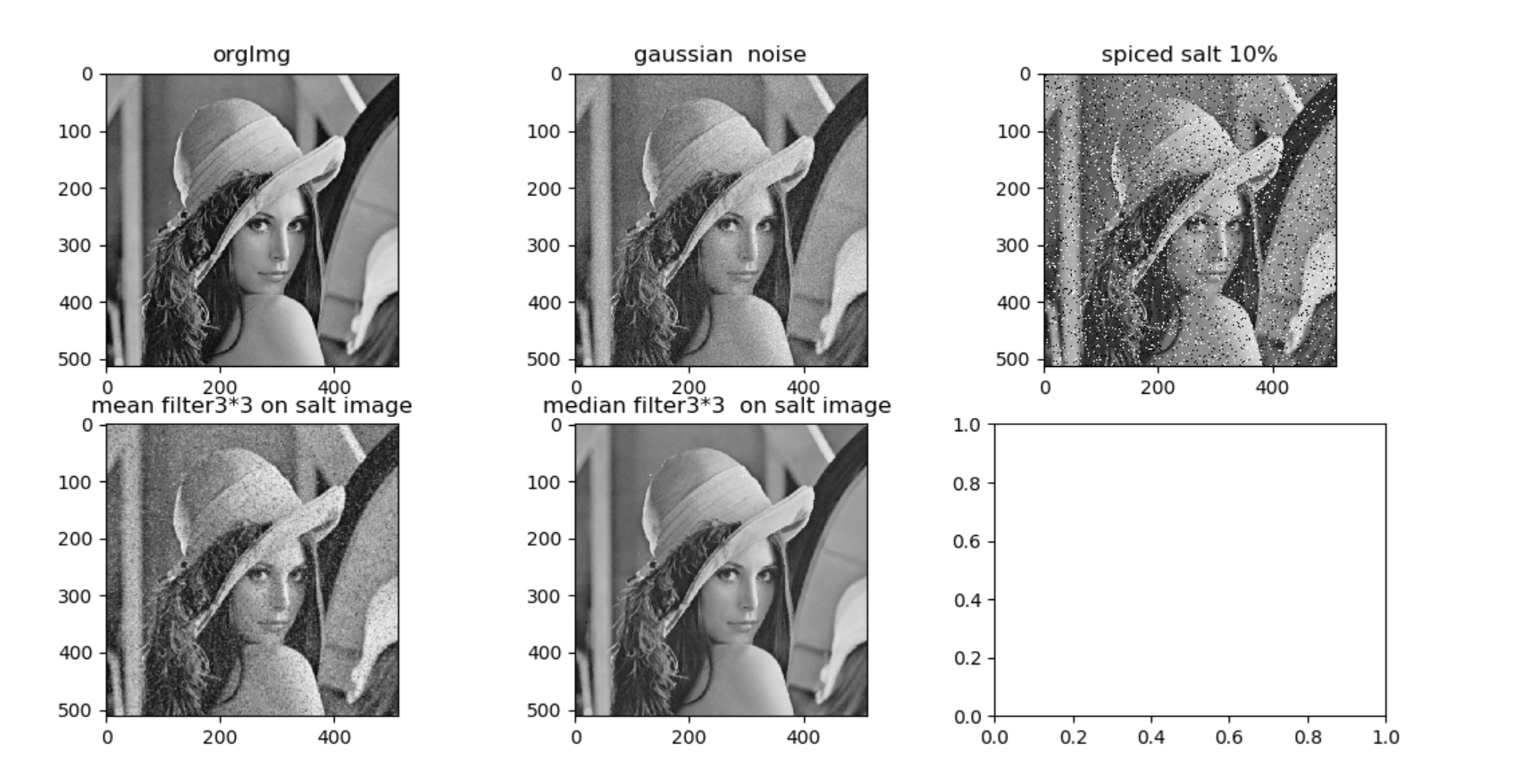
Prewitt算子：

结构：

[-1, 0, 1]

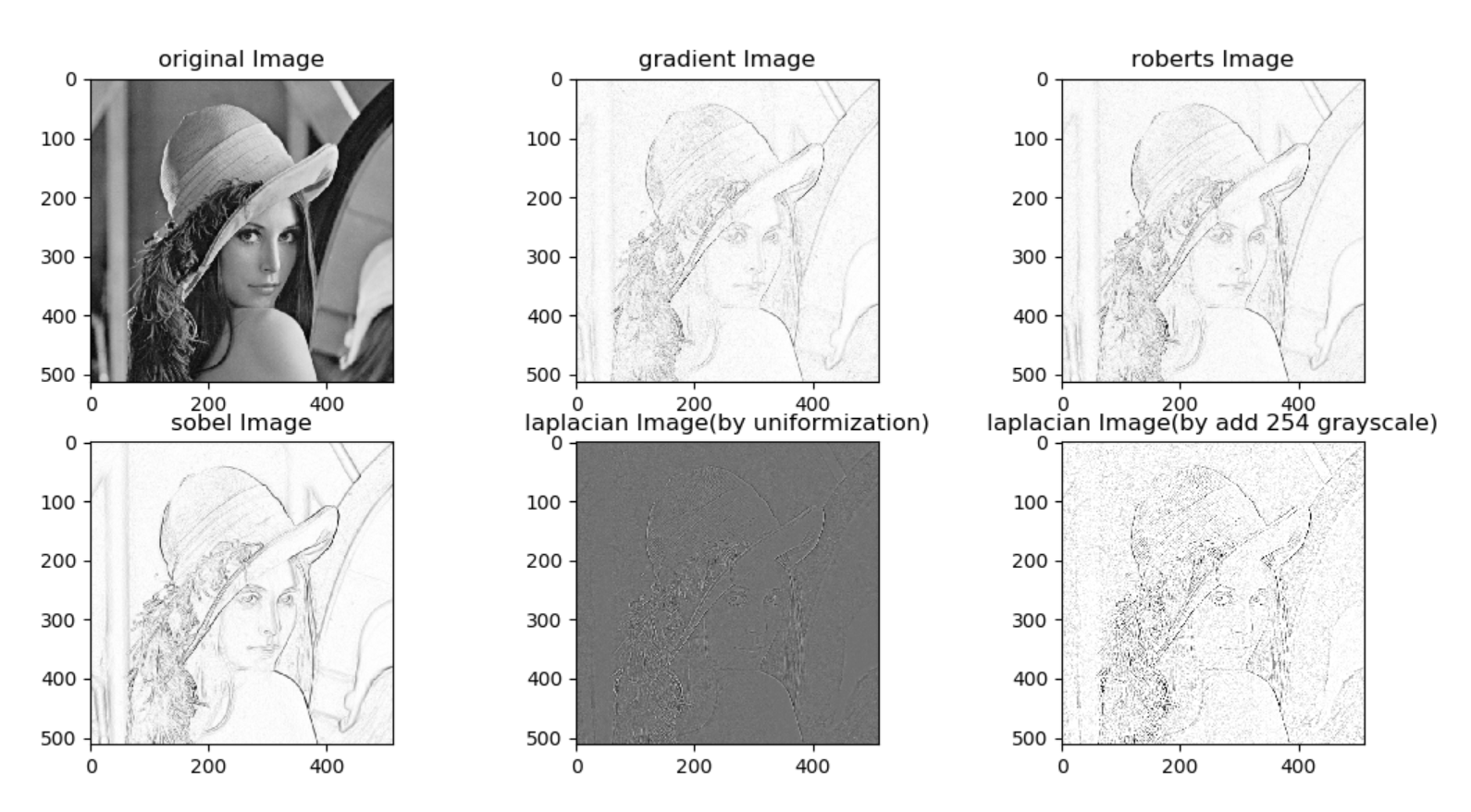
[-1, 0, 1]

[-1, 0, 1]



锐化：

梯度锐化，roberts锐化，sobel，Laplace。



# 第四周作业：对一副图像进行傅立叶变换，显示频谱，取其5，50，150为截至频率，进行频率域平滑，锐化，显示图像

处理步骤：

1）傅里叶变换：直接调用numpy的fft的2维傅里叶变换fft2,并作居中处理

2）显示频谱：取幅度谱，求对数，并通过线性变换至返回【0，255】，显示该图像

（求对数的原因是幅度谱的值比较大，超出显示范围，线性变换的原因是取完对数，范围比较小【0，20】，图像显示不易观察）

3）取D0={5，50，150}，设置滤波器H（理想，butterworth ，gauss），显示图像。步骤：

HUV = 1/（1+(DUV/D0)^2n) butterworth 低通滤波器公式，

HUV = exp(-DUV^2/2D0^2) guass 低通滤波公式

高通 HP = 1- HL

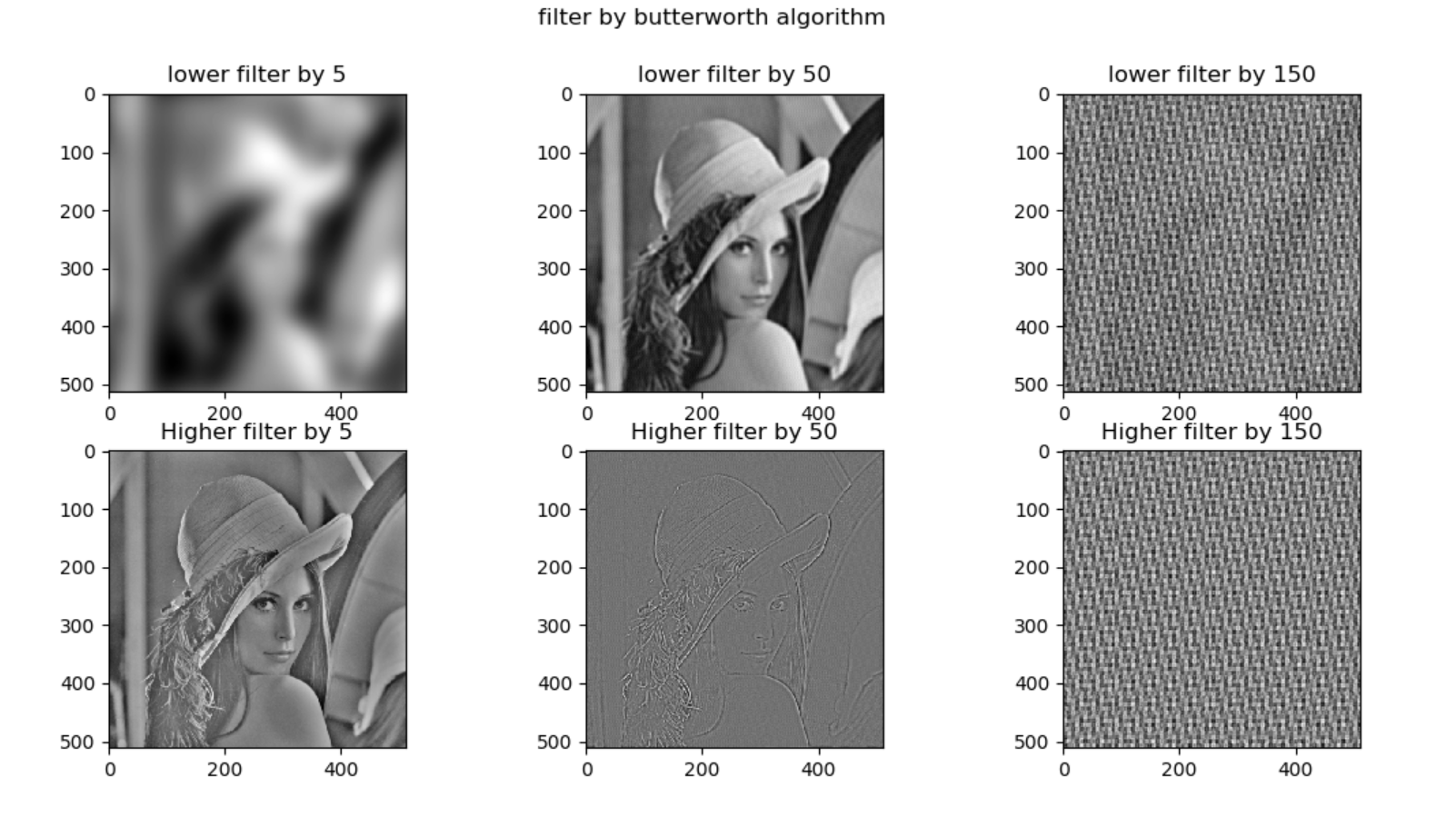
滤波操作 GUV = HUV\*FUV

傅里叶反变换（先将中心点切换回左端点），取出实部

对实部进行范围变换，线性到【0，255】范围，显示图片。

参考：https://blog.csdn.net/cjsh\_123456/article/details/79342300





# 第五周作业：对一副图像加噪，进行几何均值，算术均值，谐波，逆谐波处理

处理步骤：

1）给图像增加10%的椒盐噪声；添加方差为0.4的高斯噪声的叠加图片。（此部分同第三章，为做区别，本次做高斯与椒盐的叠加噪声）

2）使用几何平均，f(x,y) = （prod(g(s,t))^(1/mn)

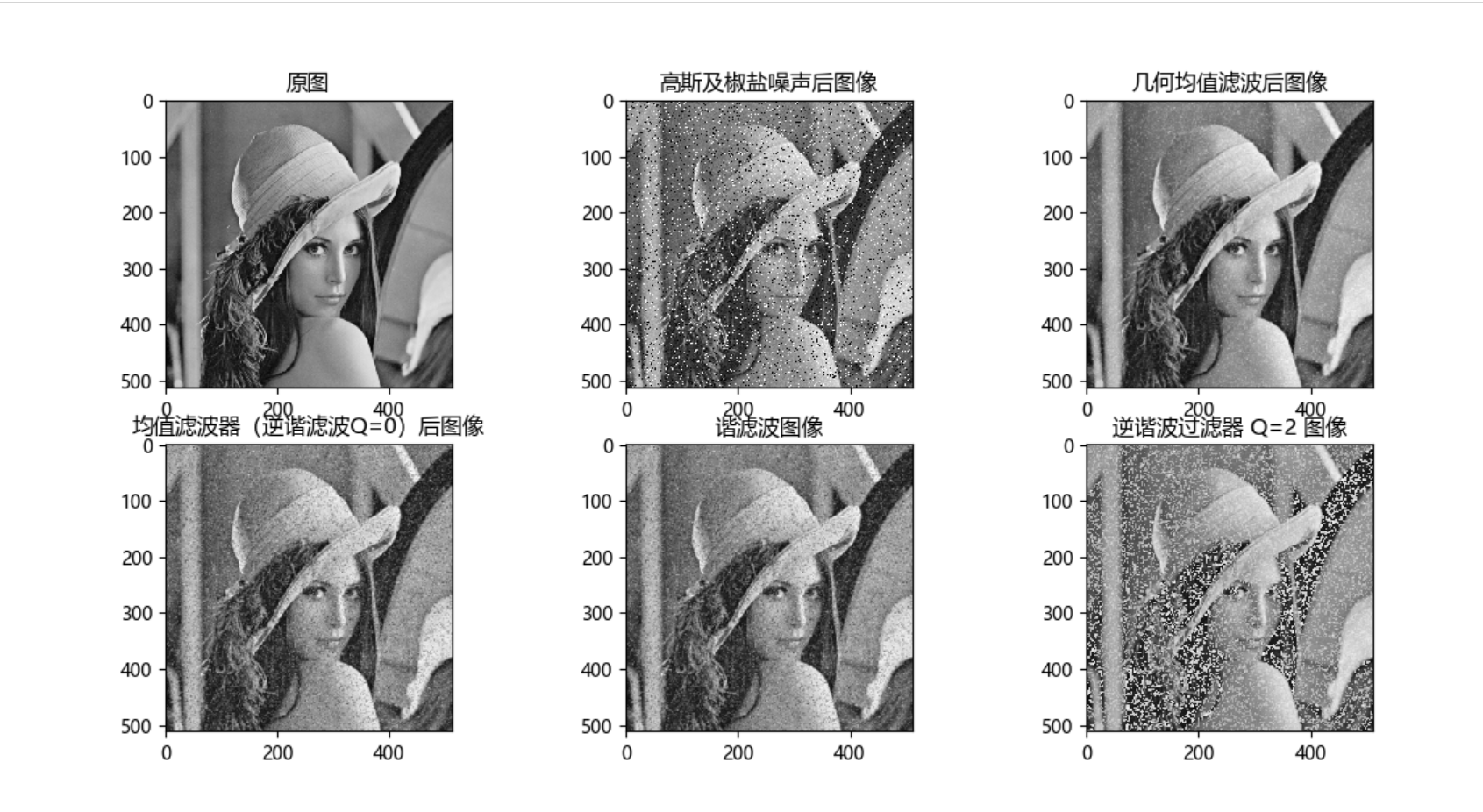
3）编制逆谐波函数，当Q = 0 退化为算术均值，当Q=-1时，退化为谐波均值滤波器

(s,t) ∈ Sxy f(x,y)= sum g(s,t)^(Q+1) / sum g(s,t)^Q

取Q=0进行算术均值计算

取Q=-1 进行谐波计算

取Q=2进行逆谐波计算



# 第六周作业：提取一副图像中的红色，用HSI方式。

步骤：

1）使用opencv的cvtColor方法将RGB图像转换成HSL图像，并提取其中的H，S分量，从H中获取红色（范围在【0，10】，【156，180】设置为255，其他为0，

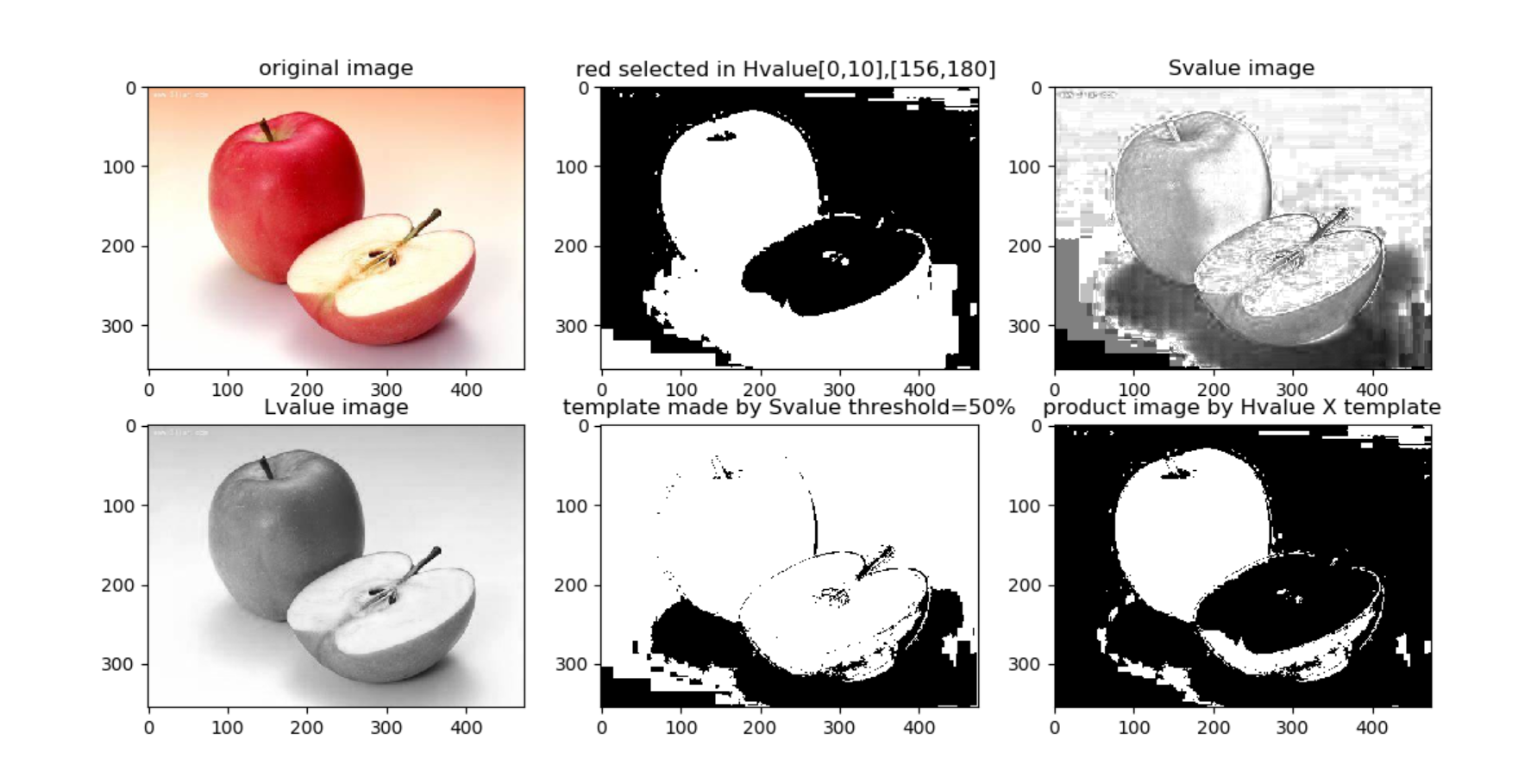
色相范围参考：

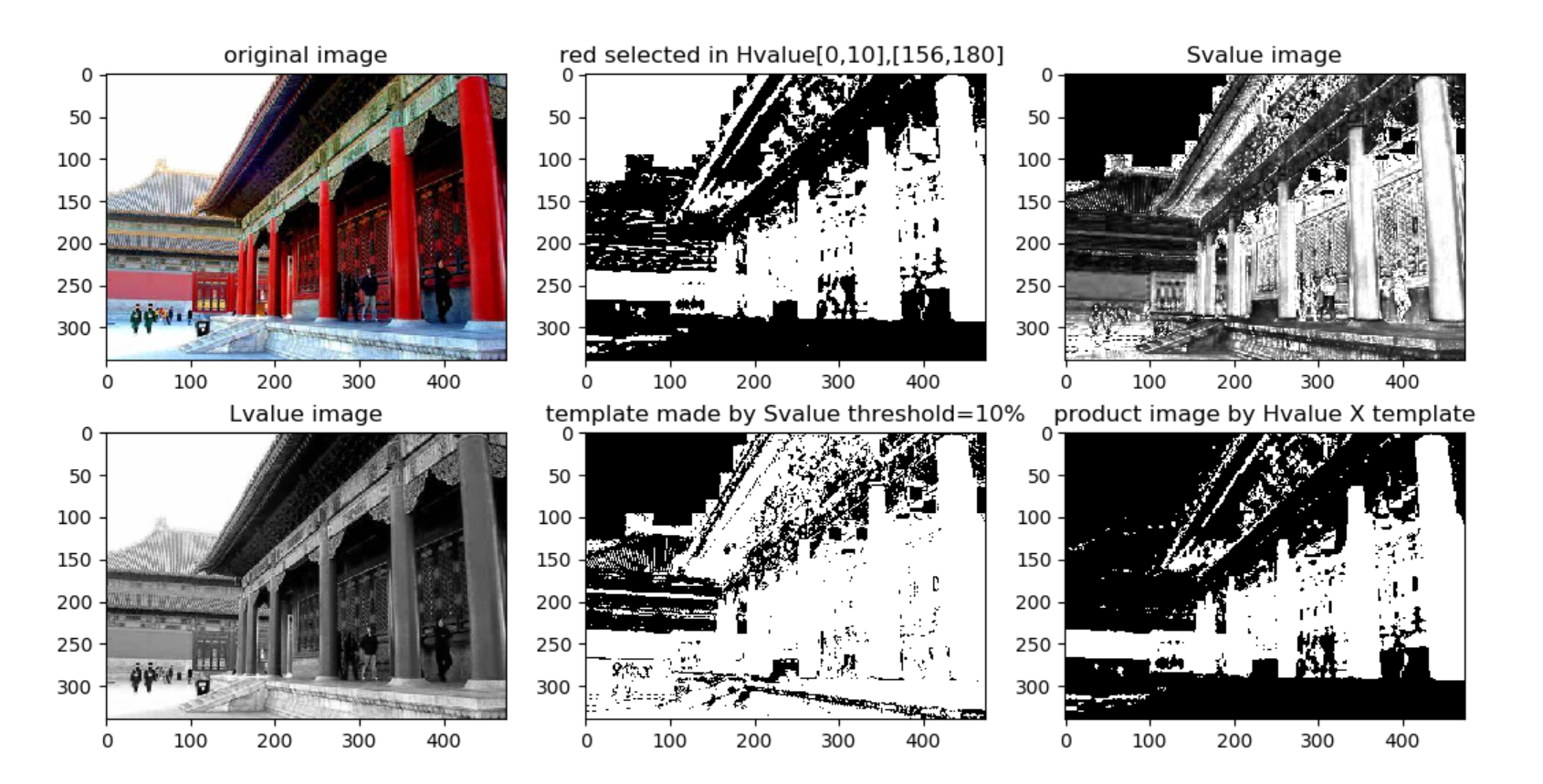
https://zh.wikipedia.org/wiki/%E8%89%B2%E7%9B%B8#%E8%89%B2%E7%9B%B8%E7%8E%AF, https://www.cnblogs.com/wangyblzu/p/5710715.html）。

注：H的正常范围是【0，360】，opencv为了适配到【0，255】范围，H的范围减半。详见：https://docs.opencv.org/master/de/d25/imgproc\_color\_conversions.html#color\_convert\_rgb\_hls

2) 使用饱和度分量制作掩模template（二值化，阈值取最大饱和度的50%，饱和度低于50%的点舍去）

3) H分量与template进行卷积，得到提取的图片。





# 第七周：对一副二值图像进行膨胀、腐蚀、开、闭操作

步骤：

1. 生成5X5的核，对图像进行膨胀操作，外围变大，与操作，全为0的为0，否则为1
2. 生成3X3的核，对图像进行腐蚀操作，剔除毛刺，与操作，存在0的为0，否则为1
3. 开操作：先腐蚀后膨胀 (A-B)+B 剔除毛边后复原
4. 闭操作：先膨胀后腐蚀 (A+B)-B

参考：https://blog.csdn.net/Eastmount/article/details/83581277

