**2016/10/17 数字图像处理第二次作业**

**1 Exercises**

**1.1 Linearity**

Histogram equalization isn't a linear operator.

Linear contrast enhancement referred a contrast stretching, linearly expands the original digital values of the remotely sensed data into a new distribution.
In contrast, histogram equalization is one of the most useful forms of nonlinear contrast enhancement. It employs a non-linear mapping which re-assigns the intensity values of pixels in the input image such that the output image contains a uniform distribution of intensities (and we get a flat histogram).[1]

**Proof by contradiction:**

A linear operator F on an image is a mapping from one image to another$I' = F(I)$, that satisfies:

1. $\mathrm{F(cI) = cF(I)}$,
2. $\mathrm{F}(I_1 + I_2) = \mathrm{F}(I_1) + \mathrm{F}(I_2)$,

whereI, $I_1$, $I_1$ are images, and c is a constant.

Since the process of histogram equalization is applying a transformation $\mathrm{g = F(f)}$ to each pixel of the input image $\mathrm{f[x,y]}$, such that a uniform distribution of gray levels results for the output image $\mathrm{g[x,y]}$. [2]

$$\mathrm{g = F(f)} = \int_0^f P_f(\alpha)d\alpha \quad (0 \le f \le 1)$$

Suppose g is linear, then

$$\mathrm{F}(f_1 + f_2) = \int_0^{f_1+f_2} P_{f_1+f_2}(\alpha)d\alpha = \mathrm{F}(f_1) + \mathrm{F}(f_2) = \int_0^{f_1} P_{f_1}(\alpha)d\alpha + \int_0^{f_2} P_{f_2}(\alpha)d\alpha$$

And suppose $P_f(\alpha) = \alpha$, then $\int_0^f P_f(\alpha)d\alpha = \frac{1}{2}\alpha^2$

$$\mathrm{F}(f_1 + f_2) = \frac{1}{2}(\alpha_1 + \alpha_2)^2$$

$$\mathrm{F}(f_1) + \mathrm{F}(f_2) = \frac{1}{2}{\alpha_1}^2 + \frac{1}{2}{\alpha_2}^2$$

thus

$$\mathrm{F}(f_1 + f_2) \neq \mathrm{F}(f_1) + \mathrm{F}(f_2)$$

It contradicts with the definition of linear operator, so we conclude histogram equalization is a non-linear operator.

**1.2 Spatial Filtering**

[1] http://www.cs.utah.edu/~lavanyat/ip/report1/report.htm
[2] http://web.stanford.edu/class/ee368/Handouts/Lectures/2013_Spring/4-Histograms_16x9.pdf

1. Padded f: $\begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&1&2&2&1&0&0 \\ 0&0&2&3&3&2&0&0 \\ 0&0&2&3&3&2&0&0 \\ 0&0&1&2&2&1&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}$

   Full convolution result: $\frac{1}{16}\begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 0&1&4&7&4&1&1&0 \\ 0&4&15&25&25&15&4&0 \\ 0&7&25&40&40&25&7&0 \\ 0&7&25&40&40&25&7&0 \\ 0&4&15&25&25&15&4&0 \\ 0&1&4&7&4&1&1&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}$

   Cropped convolution result: $\frac{1}{16}\begin{bmatrix} 15&25&25&15 \\ 25&40&40&25 \\ 25&40&40&25 \\ 15&25&25&15 \end{bmatrix}$

2. The given filter is weighted average filter. By replacing the value of every pixel in an image by the average of the intensity levels in the neighborhood defined by the filter mask, this process results in an image with reduced "sharp" transitions in intensities. However, edges also are characterized by sharp intensity transitions, so averaging filters have the undesirable side effect that they blur edges.
   Since applying the same filter twice doubles its effect, the limiting effect of repeatedly applying the given filter to an image is blurring edges of the image repeatedly at the same time of noise reduction.
3. The mechanics of convolution are the same as correlation, except that the filter is first rotated by $180°$.
4. The weighted average filter is a kind of smoothing linear filter whose application are noise reduction, smoothing of false contours that result from using an insufficient number of intensity levels and reduction of "irrelevant" detail in an image. In addition to that, the basic strategy behind weighing the center point the highest and then reducing the value of the coefficients as a function of increasing distance from the origin is simply an attempt to reduce blurring in the smoothing process.

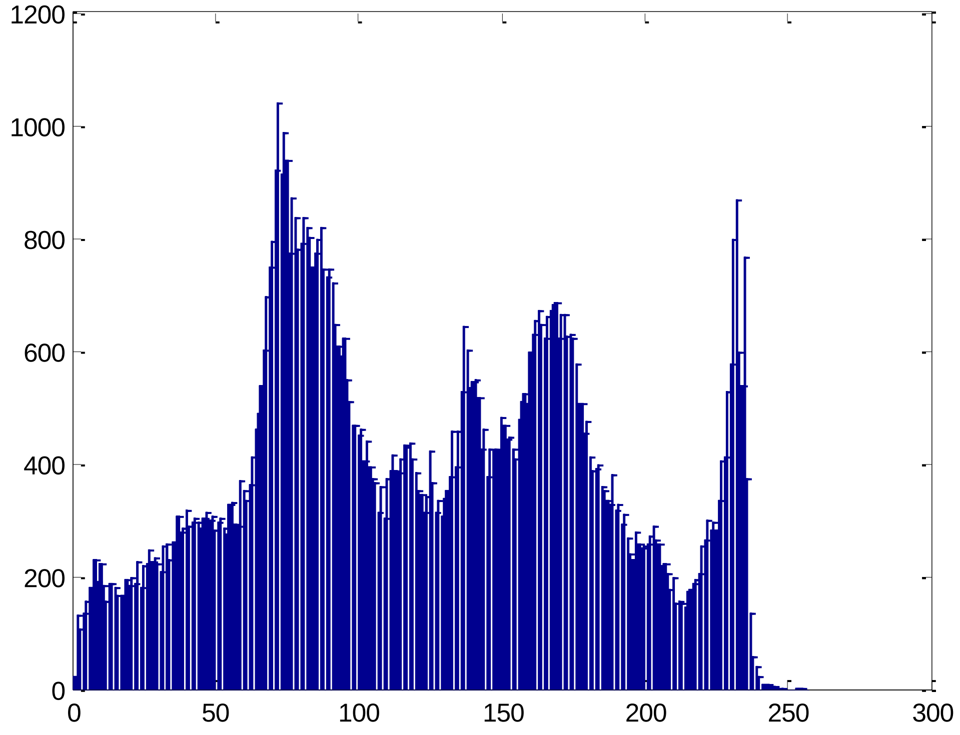
**2 Programming Tasks**

**2.1 Pre-requirement**

**2.2 Histogram Equalization**

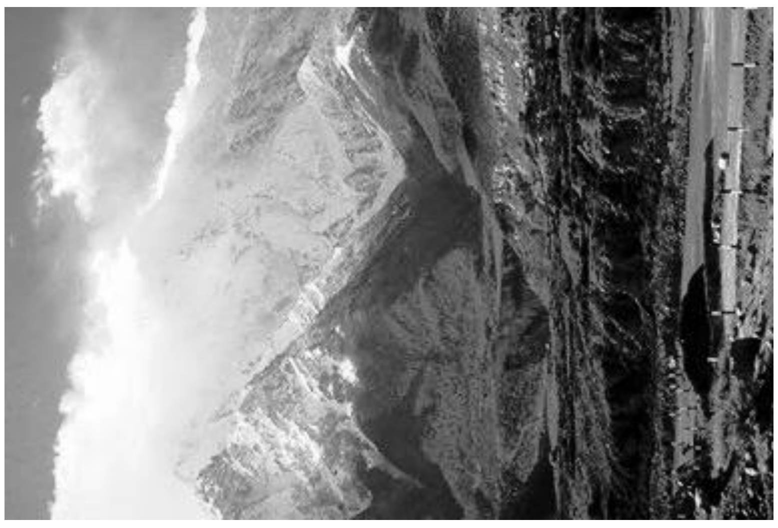
1. **Histogram of the input image.**

Origin Histogram

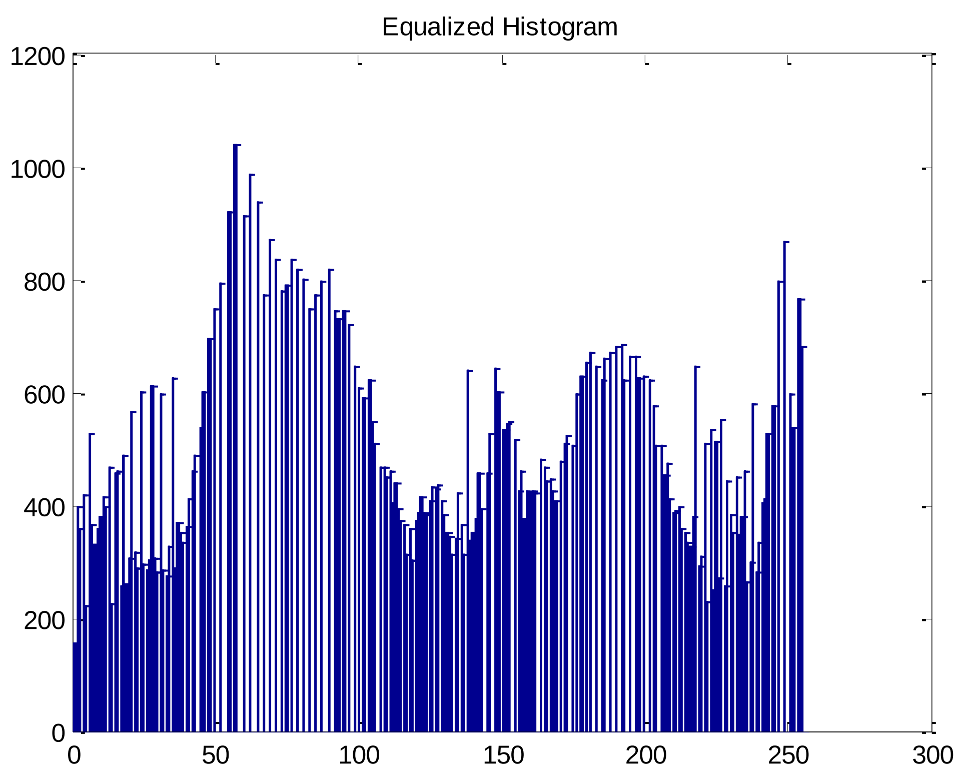


**2. Histogram-equalized image.**

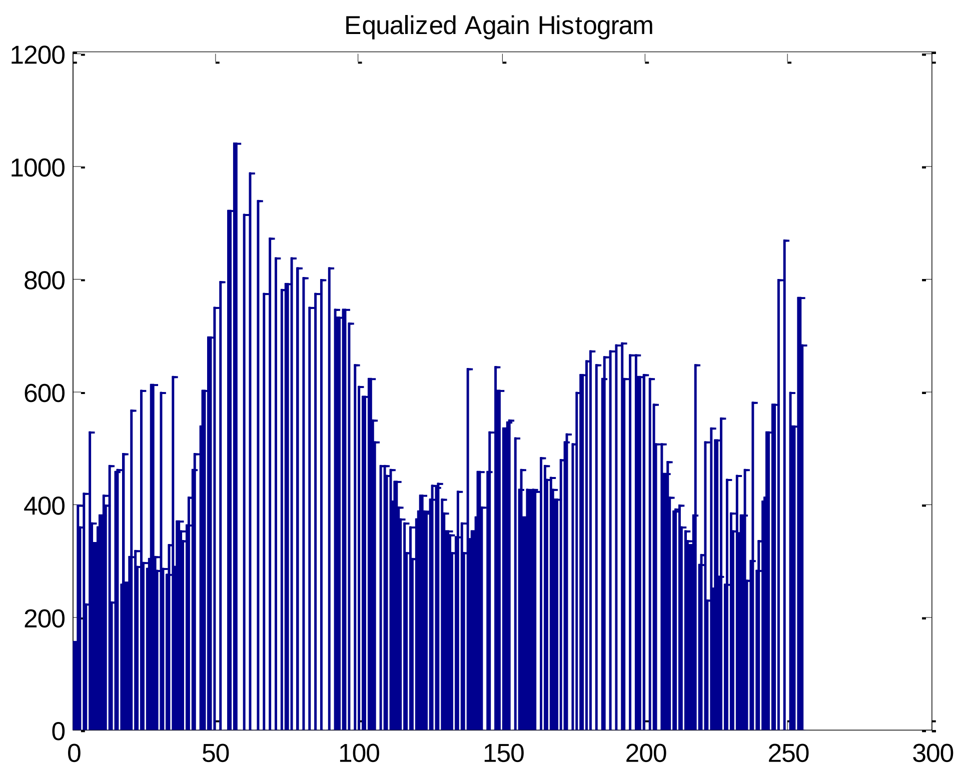
Output Image



**Corresponding histogram:**



**3. The resulting histogram for equalizing the histogram again.**



肉眼上看不出两图的差别，我通过 MATLAB 的 isequal 方法来比较两个直方图，返回的结果为 1，标明两者是相同的。

**原因：**

直方图均衡化的过程是通过转换函数将原图中的每个像素对应的灰度值转换为对应的灰度值，这个转换是一一对应的，输入图像中为某灰度值的像素个数和输出图像中为对应灰度值的像素个数相等且不会随着多次进行直方图均衡化变化。这说明每个灰度的像素个数的累积分布函数不变，即转换函数不变。

**数学证明：**

设$n$是输入图像的总像素数，$L$是输入图像总的灰度级别数，$n_{r_j}$代表输入图像中灰度值为$r_j$的像素数，直方图均衡化的转换函数为：

$$s_k = T(r_k) = round(cdf(r_j) \times (L-1))$$

$$cdf(r_j) = \sum_{j=0}^{k} \frac{n_{r_j}}{n}$$

由于每个灰度值为$r_k$的像素的灰度值都被转换为灰度值$s_k$，因此输入图像中灰度值为$r_j$的像素数等于输出图像中灰度值为$s_k$的像素数，即

$$n_{r_k} = n_{s_k}$$

进行第二次直方图均衡化时，第一次的输出图像的灰度值$s_k$被转换为第二次的输出图像的灰度值$v_k$

$$v_k = T(s_k) = round(cdf(s_j) \times (L-1))$$

$$cdf(s_j) = \sum_{j=0}^{k} \frac{n_{s_j}}{n}$$

联立上面各式，得

$$v_k = s_k$$

即证明了第二次直方图均衡化的结果和第一次的完全一样。

4. **The process of implementing the histogram equalization operation.**

**计算直方图：**

由于灰度图像的灰度值通常在[0,255]，创建一个有 256 行 1 列的矩阵 histogram 来表示直方图，用来存储图像中某灰度值的总数。

$$\text{histogram(i)} = \frac{n_{r_j}}{n}, 0 \le i < L$$

$n$代表输入图像的总像素数，$L$代表输入图像总的灰度级别数，$n_{r_j}$代表输入图像中灰度值为$r_j$的像素数。

注意因为下标的范围是[1,256]，灰度值对应到下标时要加 1。例如$\text{histogram}(200) = 100$表示图像中共有 100 个像素的灰度值为 199. 通过遍历图像中的每一个像素可以完成这个矩阵。

**直方图均衡化[3]：**

首先通过前文所述步骤获得输入图像的直方图。

再计算直方图的累积分布函数$cdf$

$$cdf(r_j) = \sum_{j=0}^{k} \frac{n_{r_j}}{n}$$

通过转换函数将输入图像的灰度值$r_k$转换为输出图像的灰度值$s_k$

$$s_k = T(r_k) = round(cdf(r_j) \times (L-1))$$

并存入数组 equalized。

[3] http://www.math.uci.edu/icamp/courses/math77c/demos/hist_eq.pdf

最后遍历输入图像每个像素对应的灰度值，在 equalized 数组中找到对应的输出图像该像素的灰度值。注意，输入的灰度值先加 1 得到 equalized 的对应下标，即

$$\mathrm{output\_img(r,c)} = \mathrm{equalized(input\_img(r,c)} + 1)$$

完毕。

**转换函数的由来：**

上面转换函数的思想来自于将$s_k$的分布看作连续的随机变量X,Y在$[0, \mathrm{L}-1]$区间定义的函数

$$\mathrm{Y} = \mathrm{T(X)} = (\mathrm{L}-1)\int_0^X P_X(x)dx$$

其中$P_X$输入图像的概率分布函数。T是X的累积分布函数乘上$(\mathrm{L}-1)$。

简便起见，假设T可微和可逆。可以证明通过T(X)定义的Y均匀分布在$[0, \mathrm{L}-1]$上，即$P_Y(y) = \frac{1}{L-1}$

$$\int_0^y P_Y(z)dz = probability\ that\ 0 \le Y \le y = probability\ that\ \ 0 \le X \le T^{-1}(y)$$

$$= \int_0^{T^{-1}(y)} P_X(\omega)d\omega$$

$$\frac{d}{dy}\left(\int_0^y P_Y(z)dz\right) = P_Y(y) = P_X\left(T^{-1}(y)\right)\frac{d}{dy}(T^{-1}(y))$$

又因为$\frac{d}{dy}T\left(T^{-1}(y)\right) = \frac{d}{dy}y = 1$，因此

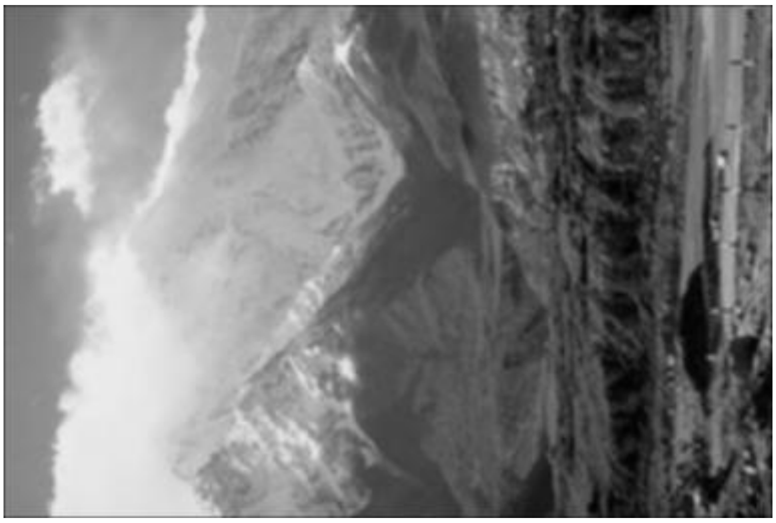
$$\frac{\mathrm{dT}}{\mathrm{dx}}|_{x=T^{-1}(y)}\frac{d}{dy}\left(T^{-1}(y)\right) = (L-1)P_X\left(T^{-1}(y)\right)\frac{d}{dy}\left(T^{-1}(y)\right) = 1$$

即证$P_Y(y) = \frac{1}{L-1}$

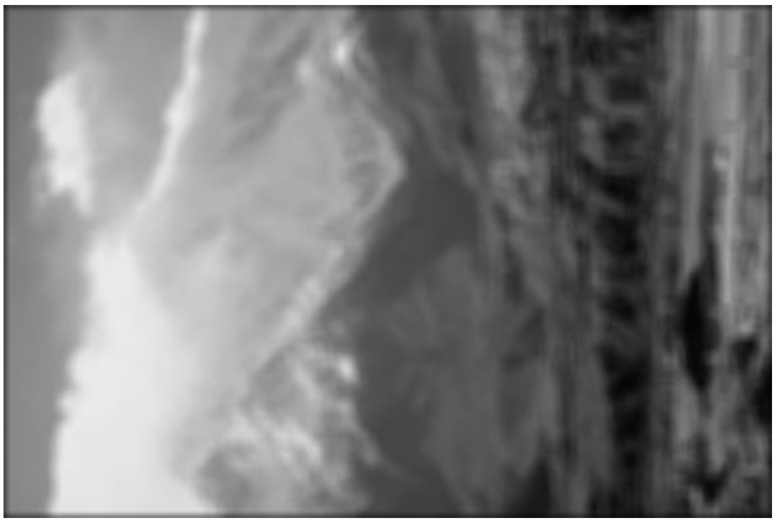
**2.3 Spatial Filtering**

1.

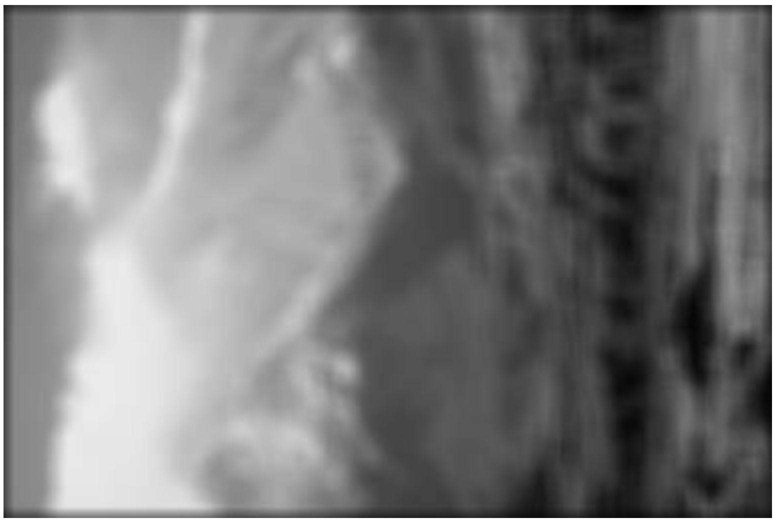
3*3 Averaging Filter Image



7*7 Averaging Filter Image



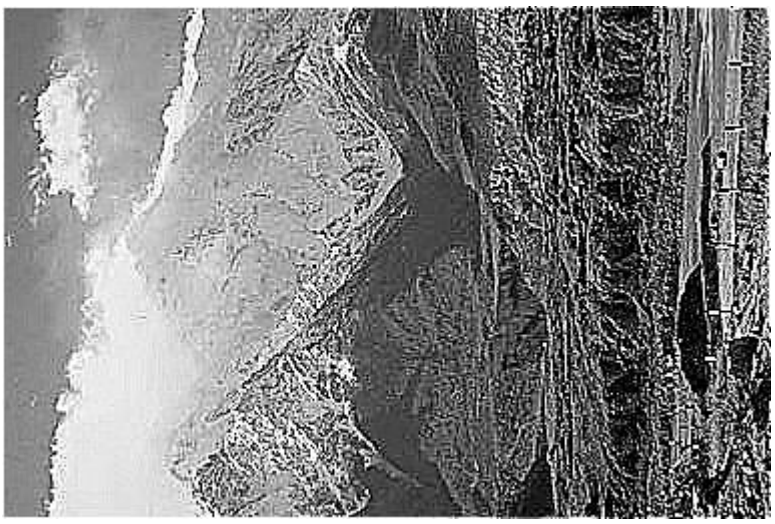
11*11 Averaging Filter Image



2.

4.

3*3 Laplacian Filters Image, c = -1



Why Laplacian filter can be used for sharpening.

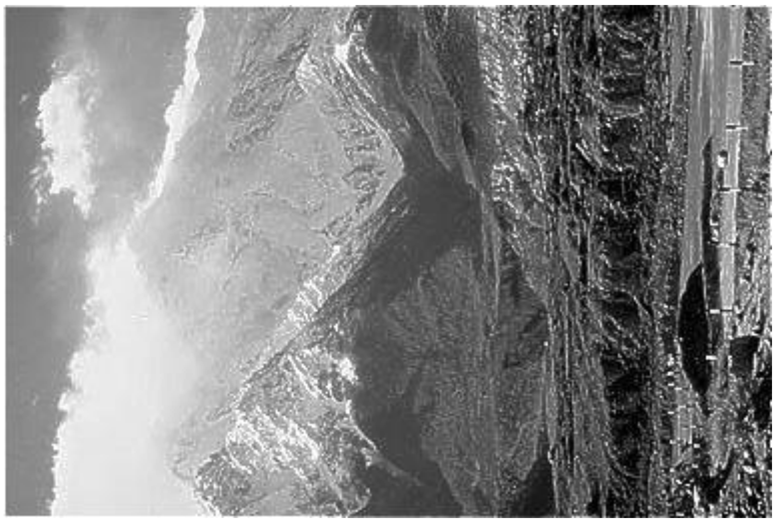
Reference: Section 3.6 Sharpening Spatial Filters of the textbook.

Because the Laplacian is a derivative operator, its use highlights intensity discontinuities in an image and deemphasizes regions with slowly varying intensity levels. This will tend to produce images that have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background. Background features can be “recovered” while still preserving the sharpening effect of the Laplacian simply by adding the Laplacian image to the original.

3.

By comparing the result of applying different k, I select 3 for k since it performs better.

3*3 High-boost Filters Image, k = 3



4.

空间滤波器的工作原理是对图像的每个像素进行处理，处理的范围是以这个像素为中心的矩形（正方形），处理的内容是对这个矩形内的像素根据滤波器进行加权求值并通过一定的变换赋值给输出图像的对应中心像素。

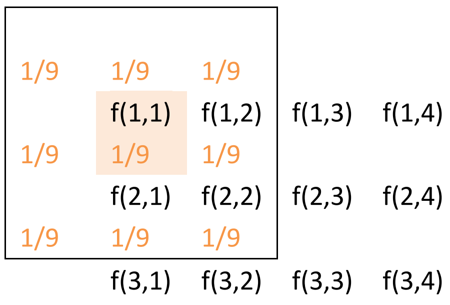
要注意的是在图像靠近边界的像素以它为中心不能构成一个矩形，此时课本的做法是先将输入图像进行扩充，向输入图像的四个边界填充 0 直到边界像素也能作为中心点处理。我的做法是省略了这个“扩充”的过程，直接对输入图像的每个像素进行处理，同时判断以处理像素形成的“虚拟”矩形是否越界，若越界，则越界的点不加以计算，这和填充 0 的效果是一样的。

题目围绕空间滤波器展开，三个小问分别用到的滤波器分别是均值空间滤波器、拉普拉斯空间滤波器、高增益空间滤波器，由于他们的整体做法一致，我用同一个函数`filter2d`处理，并通过第三个参数k进行区分。均值空间滤波器对应k为0，拉普拉斯空间滤波器对应k为-1，其他值表示用的是高增益空间滤波器。

算法首先求出上述的加权值。具体做法是按照每行每列的顺序遍历每一个像素，对这一个像素，滤波器以它为中心，计算滤波器矩形范围内的权重和对应位置的像素的灰度值的乘积和。如下图是$3\times3$的平均滤波器（用黑色框表示）的例子，$\mathrm{f(r,c)}$表示第r行第c列的像素对应的灰度值，浅橘色阴影对应的是此刻计算的像素点

$$\mathrm{intensity}(1,1)=\mathrm{f}(1,1)\times\frac{1}{9}+\mathrm{f}(1,2)\times\frac{1}{9}+\mathrm{f}(2,1)\times\frac{1}{9}+\mathrm{f}(2,2)\times\frac{1}{9}$$

以此类推。



接下来要根据不同的滤波器进行不同的操作。对于均值滤波器，计算得到的加权值即输出图像的灰度值：

$$\mathrm{output\_img}(r,c)=\mathrm{intensity}(r,c)$$

对于拉普拉斯滤波器，输出图像的灰度值等于输入图像的灰度值加上 c 倍的加权值：

$$\mathrm{output\_img}(r,c)=\mathrm{input\_img}(r,c)+c\times\mathrm{intensity}(r,c)$$

对于题目中应用的拉普拉斯滤波器，取$c=-1$.

对于高增益滤波器，首先输入图像的灰度值减去加权值得到 mask，输出图像的灰度值为输入图像的灰度值加k倍的 mask：

$$\mathrm{mask}=\mathrm{input\_img}(r,c)-\mathrm{intensity}(r,c)$$

$$\mathrm{output\_img}(r,c)=\mathrm{input\_img}(r,c)+k\times\mathrm{mask}$$

对于高增益滤波器，k为正数，经试验，取$k=3$效果较好。

完毕。