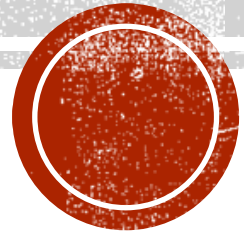
遥感数字图像处理实验课

图像增强

李荣昊

2021年12月



实验内容与目的

- 图像直方图的拉伸与均衡
- 频率滤波器

图像增强

- 是什么?

  将原来不清晰的图像变得清晰或**强调**某些关注的特征，**抑制**非关注的特征，使之改善图像质量、丰富信息量，加强图像判读和识别效果的图像处理方法。

- 目的

  - 改善图像的视觉效果，提高图像的清晰度
  - 使图像转化成一种更适合人或机器进行分析处理的形式，

图像增强

- 空间域增强

直接对图像像素灰度操作

- 点运算
  - ✓ 灰度变换
  - ✓ 直方图修正法
    - 均衡化
    - 规定化
  - ✓ 局部统计法
- 局部运算
  - ✓ 图像平滑
  - ✓ 图像锐化

- 频率域增强

对图像经傅里叶变变换后的频谱成分进行操作，然后经傅里叶逆变换获得所需要的结果。

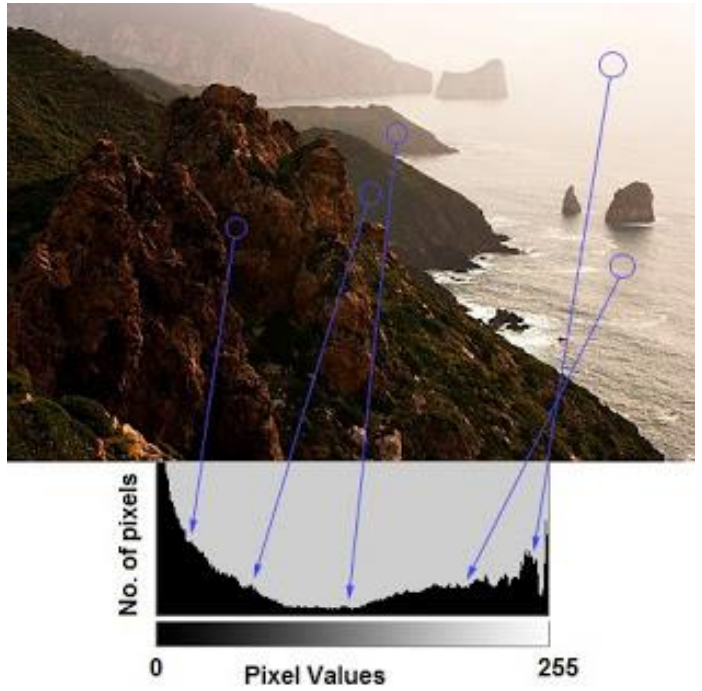
- 高通滤波
- 低通滤波
- 同态滤波增强

空间域增强——图像灰度直方图

- 查看直方图信息
- 直方图拉伸
- 直方图均衡

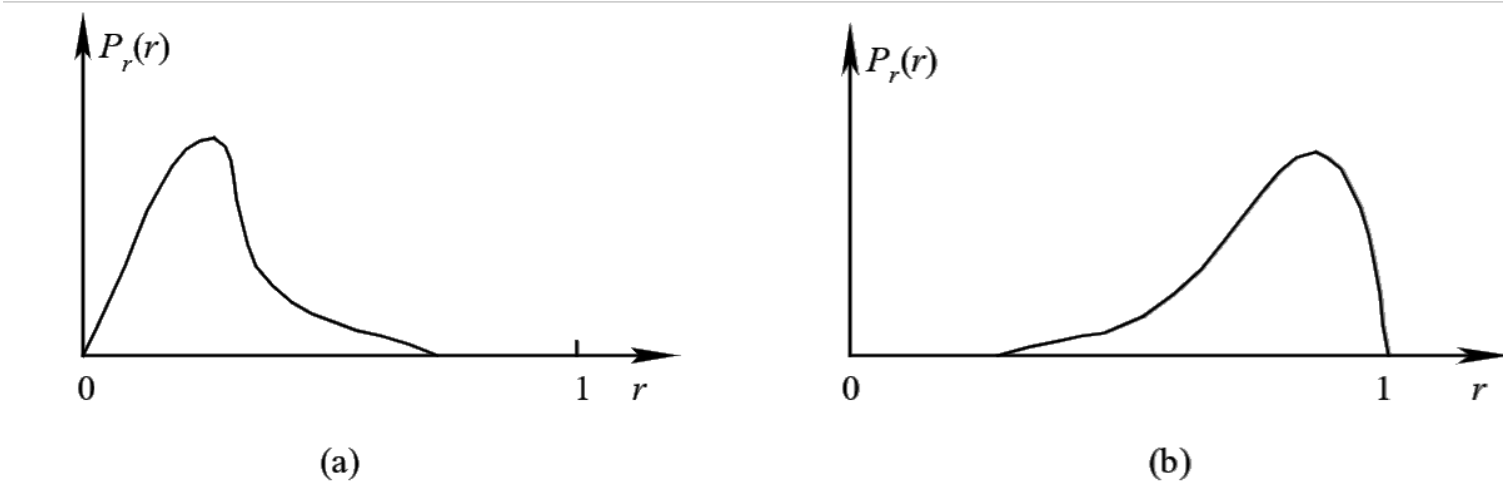
空间域增强——图像灰度直方图

- 灰度直方图
  - 灰度值的函数，描述的是图像中具有该灰度值的像素的个数或者比率。
    - 横坐标表示像素的灰度级别；
    - 纵坐标是该灰度出现的频率（像素的个数）或者概率.
  - 对离散图像来说，H(D)描述的是图像中具有该灰度级D的像素的个数。



空间域增强——图像灰度直方图

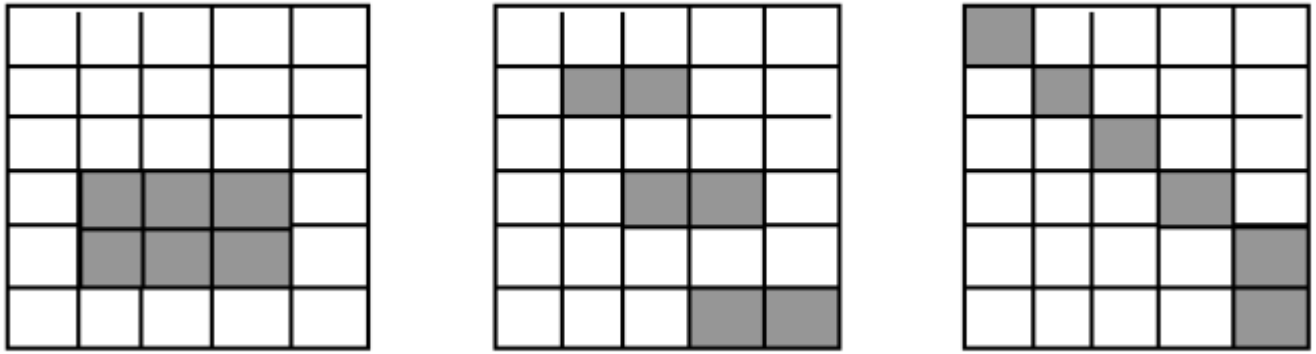
- 灰度直方图性质



- a：大多数像素集中在较暗的区域，所以这幅图一定比较暗，可能是曝光过弱
- b：像素集中在较亮的区域，所以b图偏亮

空间域增强——图像灰度直方图

- 灰度直方图性质



- 每张图都能唯一地确定一副与它对应地直方图
- 图像与直方图之间是多对一的映射关系

空间域增强——图像灰度直方图

- 灰度直方图应用

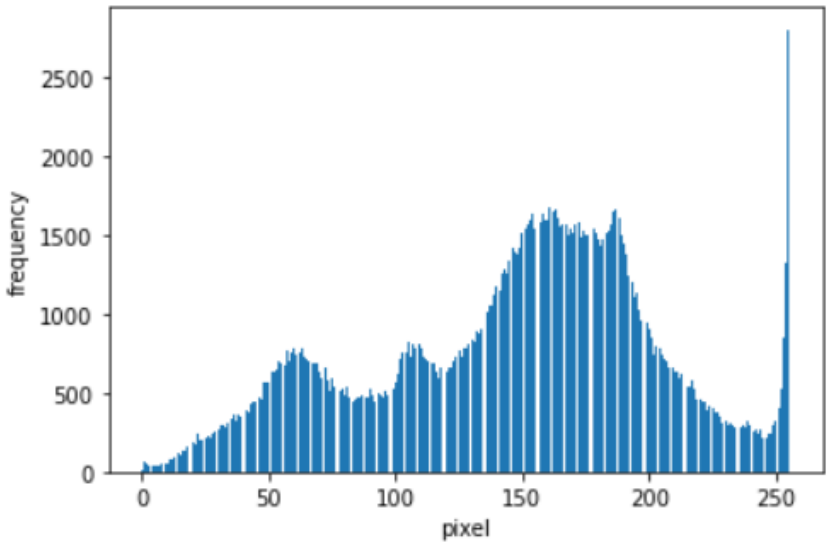
  - 判断图像量化是否恰当
  - 确定图像二值化阈值
  - 利用图像直方图统计图像中物体的面积
  - 计算图像的信息H

空间域增强——图像灰度直方图

- 查看灰度直方图

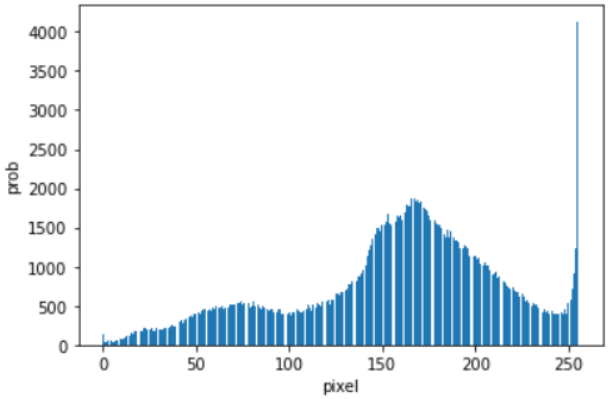
```python
def get_histogram(image):
    #frequency:频数
    frequency = np.zeros((256))
    h,w = image.shape
    #遍历图像计算灰度出现次数
    for i in range(h):
        for j in range(w):
            data = image[i][j]
            frequency[data] += 1
    return np.int32(frequency)
```

```python
def draw_hist(frequency):
    plt.bar([i for i in range(len(frequency))],frequency)
    plt.xlabel("pixel")
    plt.ylabel("frequency")
    plt.show()
```

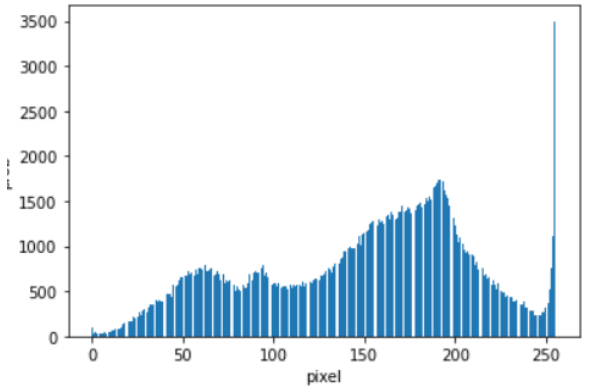


空间域增强——图像灰度直方图

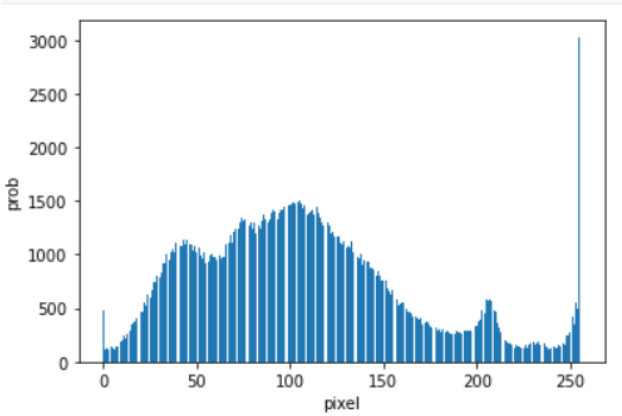
- 查看灰度直方图



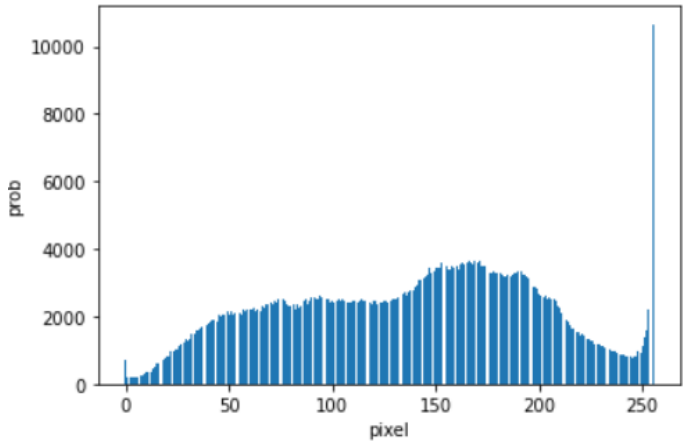
R通道



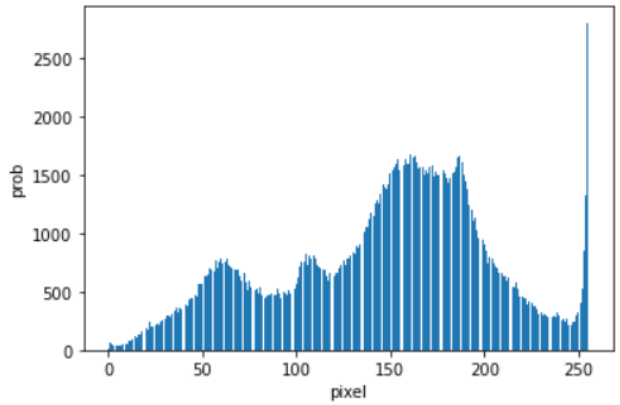
G通道



B通道



真彩色图



灰度图

空间域增强——图像灰度直方图

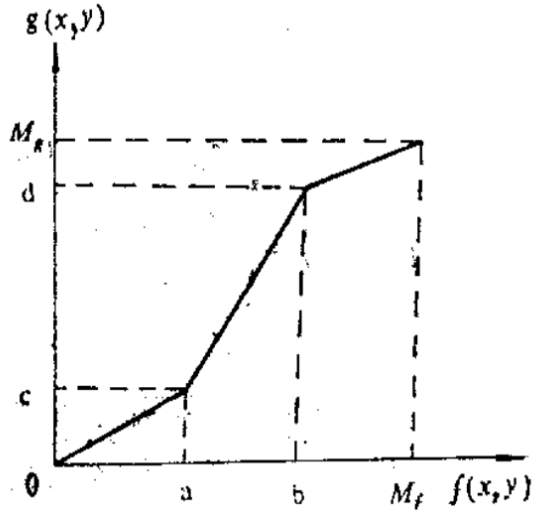
- 灰度直方图拉伸
  - 点运算：对每个像素点的灰度信息作变换，输出图像的灰度值仅由原像素灰度值以及变换函数决定。
  - 分段进行线性变换，灵活控制直方图的分布

若图像灰度在0–M 范围内，其中大部分像素的灰度级分布在区间$[x_1, x_2]$内，很小部分像素的灰度级超出此区间，则映射关系为：

$$f(x)=\begin{cases}\dfrac{y_1}{x_1}x, & 0 \le x < x_1 \quad \text{灰度区间被压缩} \\ \dfrac{y_2-y_1}{x_2-x_1}(x-x_1)+y_1, & x_1 \le x \le x_2 \quad \text{灰度区间进行拉伸} \\ \dfrac{255-y_2}{255-x_2}(x-x_2)+y_2, & x_2 < x \le M \quad \text{灰度区间被压缩}\end{cases}$$

空间域增强——图像灰度直方图

- 灰度直方图拉伸



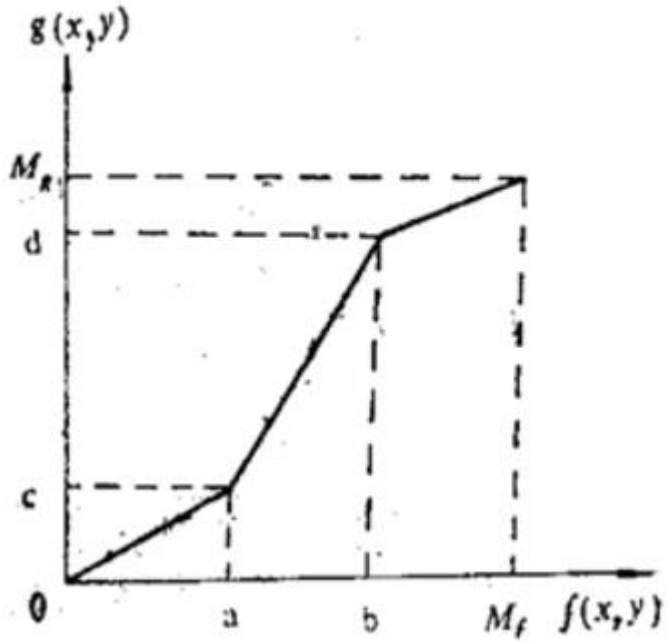
➢ 为了突出感兴趣目标所在的灰度区间，相对抑制那些不感兴趣的灰度区间，可采用分段线性变换

空间域增强——图像灰度直方图

- 灰度直方图拉伸

```python
def hist_stretch(image,a,b,c,d):
    h,w = image.shape
    out = np.zeros((h,w),dtype = np.uint8)
    for i in range(h):
        for j in range(w):
            data = image[i][j]
            #分情况讨论
            if data in range(0,a):
                out[i][j] = int(data*(c/a))
            elif data in range(a,b):
                out[i][j] = int((d-c)/(b-a)*(data-a)+c)
            else:
                out[i][j] = int((255-d)/(255-b)*(data-b)+d)
    return out
```

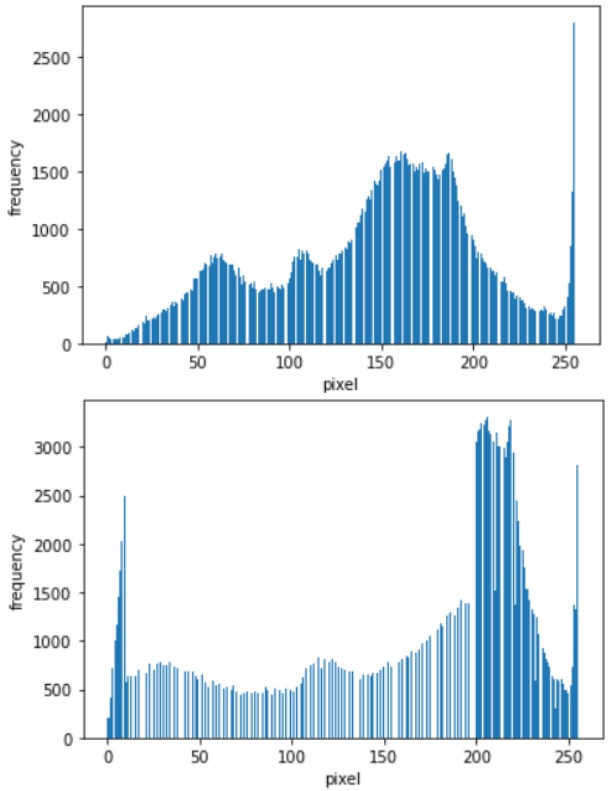
.4s Python

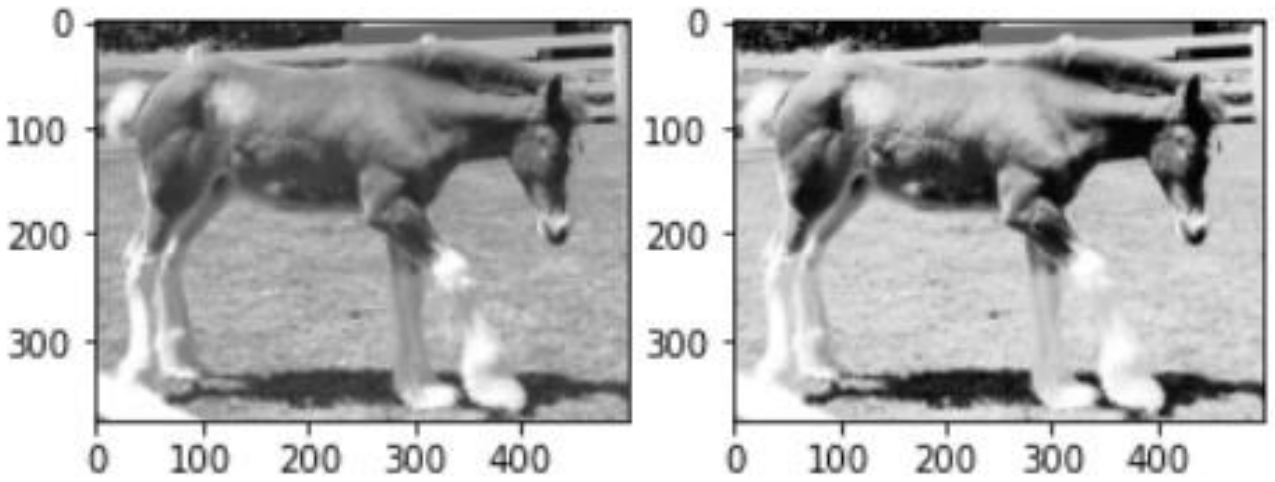


空间域增强——图像灰度直方图

- 灰度直方图拉伸

```
image_np_2 = hist_stretch(image_np,50,150,10,200)
freq2 = get_histogram(image_np_2)
draw_hist(freq2)
```

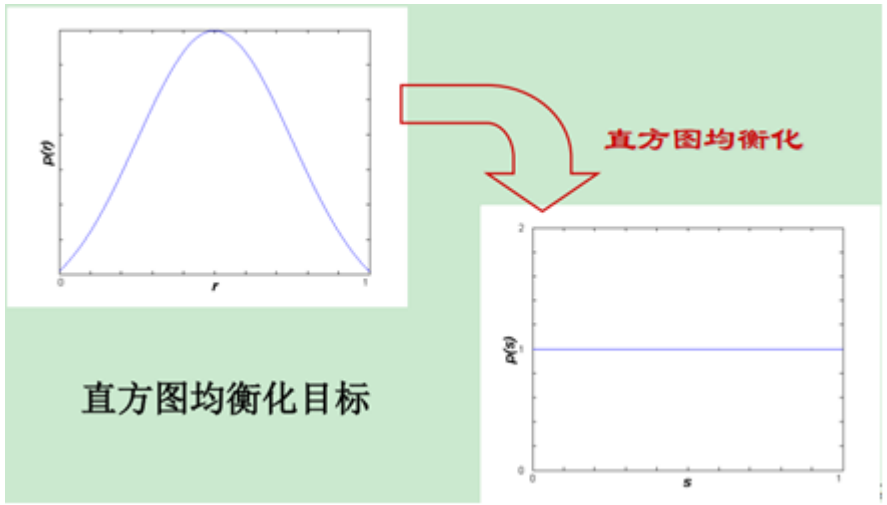




空间域增强——图像灰度直方图

- 灰度直方图均衡
  - 目的

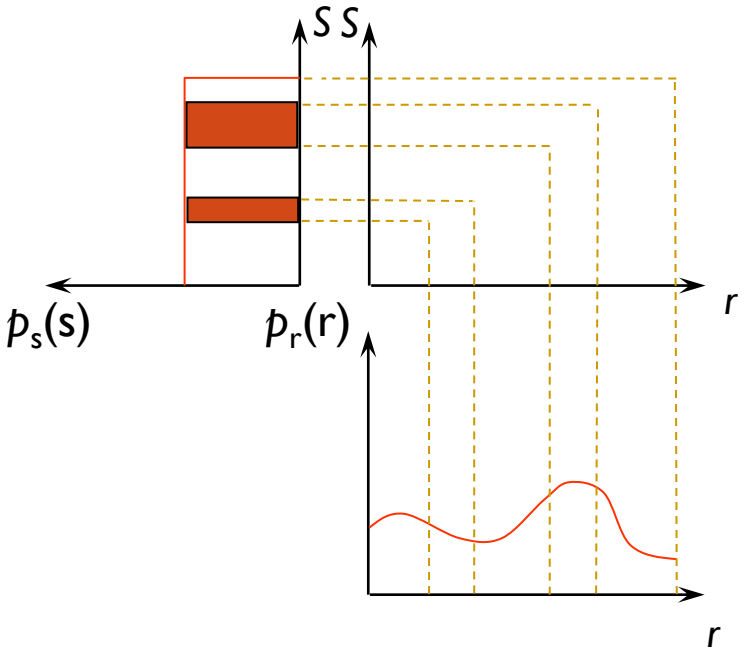
将输入图像通过一定变换，得到一幅图像，它的每个灰度级上都有相同的像素个数（即输出直方图是平的）如此一来，灰度级别被最大限度的利用，达到对比度增强的效果。



空间域增强——图像灰度直方图

- 灰度直方图均衡

➢ 连续函数



对于连续的函数，r和s分别表示归一化的原图像灰度和经直方图修正后的图像灰度，$P_r(r)$和$P_s(s)$分别是灰度r和s的概率密度函数，可知：

$$s=T(r)$$

$$p_s(s)ds = p_r(r)dr$$

直方图均衡化的目的是保证每个灰度级的概率密度相等，即是一个常数，这里归一化为1.

$$p_s(s) = k = 1$$

$$p_s(s)=1 \Rightarrow ds = P_r(r)dr$$

$$\because s=T(r) \quad \therefore ds=dT(r)=p_r(r)d_r$$

$$s=T(r)=\int_0^r p_r(r)d_r$$

空间域增强——图像灰度直方图

- 灰度直方图均衡

  - 离散情况

    设一幅图像总像素为n，分为L个灰度级，n_k代表第k个灰度级r_k出现的频率，则第k个灰度级出现的概率为：

$$p_r(r_k) = n_k / n, \qquad (0 \le r_k \le 1, \quad k = 0,1,\ldots L-1)$$

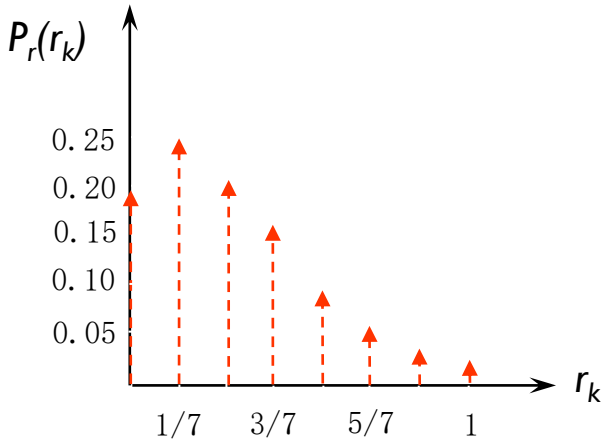
    此时，变换函数可以表示为：

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k} n_j / n$$

空间域增强——图像灰度直方图

- 灰度直方图均衡

  - 假设原图像大小为64×64,$n=64\times64=4096$，有 8 个灰度级，灰度级分布在直方图$p_r(r_k)$:



$$r_0=0 \quad n_0=790 \quad p_r(0)=0.19$$
$$r_1=1/7 \quad n_1=1023 \quad p_r(1)=0.25$$
$$r_2=2/7 \quad n_2=850 \quad p_r(2)=0.21$$
$$r_3=3/7 \quad n_3=656 \quad p_r(3)=0.16$$
$$r_4=4/7 \quad n_4=329 \quad p_r(4)=0.08$$
$$r_5=5/7 \quad n_5=245 \quad p_r(5)=0.06$$
$$r_6=6/7 \quad n_6=122 \quad p_r(6)=0.03$$
$$r_7=1 \quad n_7=81 \quad p_r(7)=0.02$$

空间域增强——图像灰度直方图

- 灰度直方图均衡

$r_0 = 0$
$r_1 = 1/7$
$r_2 = 2/7$
$r_3 = 3/7$
$r_4 = 4/7$
$r_5 = 5/7$
$r_6 = 6/7$
$r_7 = 1$

$n_0 = 790$ $p_r(0) = 0.19$
$n_1 = 1023$ $p_r(1) = 0.25$
$n_2 = 850$ $p_r(2) = 0.21$
$n_3 = 656$ $p_r(3) = 0.16$
$n_4 = 329$ $p_r(4) = 0.08$
$n_5 = 245$ $p_r(5) = 0.06$
$n_6 = 122$ $p_r(6) = 0.03$
$n_7 = 81$ $p_r(7) = 0.02$

根据变换函数公式可算出S_k如下:

$$s_{0\text{计算}} = T(r_0) = \sum_{j=0}^{0} p_r(r_j) = p_r(r_0) = 0.19$$

$$s_{1\text{计算}} = T(r_1) = \sum_{j=0}^{1} p_r(r_j) = p_r(r_0) + p_r(r_1) = 0.19 + 0.25 = 0.44$$

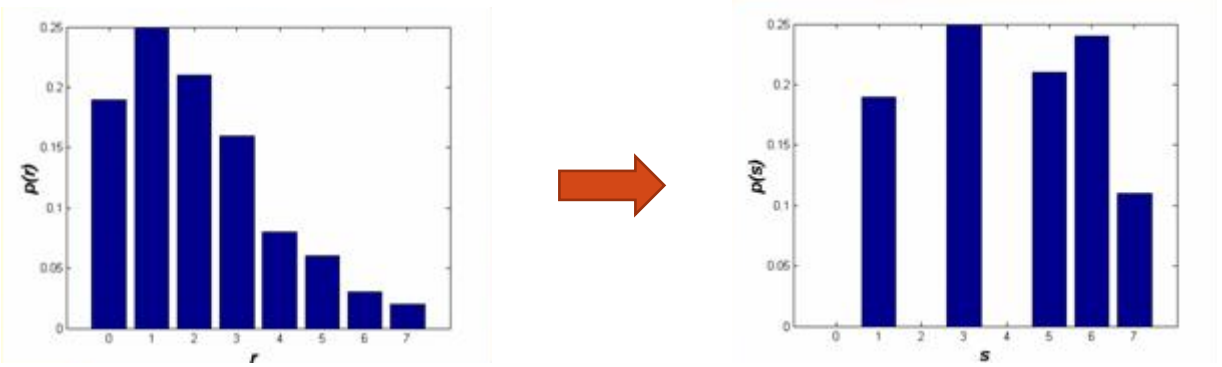
$s_{2\text{计算}} = 0.65$ $s_{3\text{计算}} = 0.81$ $s_{4\text{计算}} = 0.89$

$s_{5\text{计算}} = 0.95$ $s_{6\text{计算}} = 0.98$ $s_{7\text{计算}} = 1.00$

空间域增强——图像灰度直方图

- 灰度直方图均衡

r_k	n_k	$p(r_k)$	$S_{k计算}$	$S_{k映射}$	$S_{k舍入}$
r_0=0	790	0.19	0.19	1.33	1
r_1=1	1023	0.25	0.44	3.08	3
r_2=2	850	0.21	0.65	4.55	5
r_3=3	656	0.16	0.81	5.67	6
r_4=4	329	0.08	0.89	6.23	6
r_5=5	245	0.06	0.95	6.65	7
r_6=6	122	0.03	0.98	6.86	7
r_7=7	81	0.02	1	7	7



直方图均衡，增强图像的对比度是以灰度级减少为代价的，结果并不是理想的平直方图。

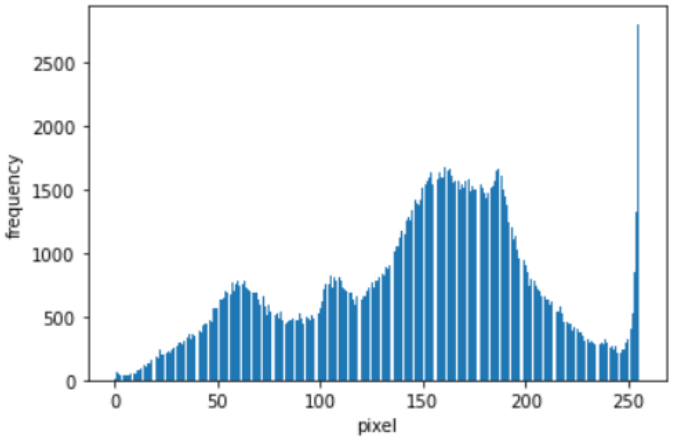
空间域增强——图像灰度直方图

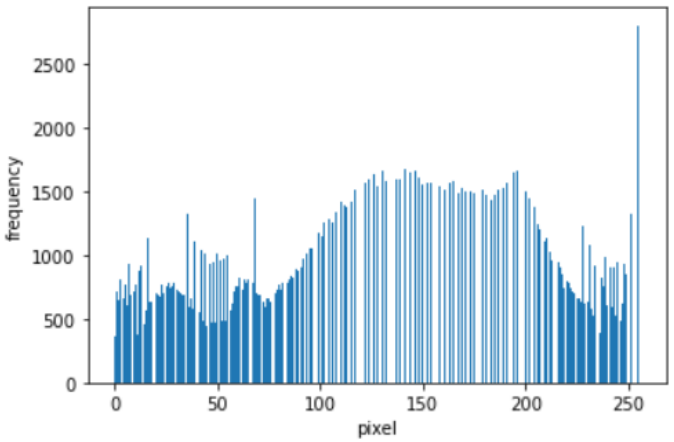
- 灰度直方图均衡实现

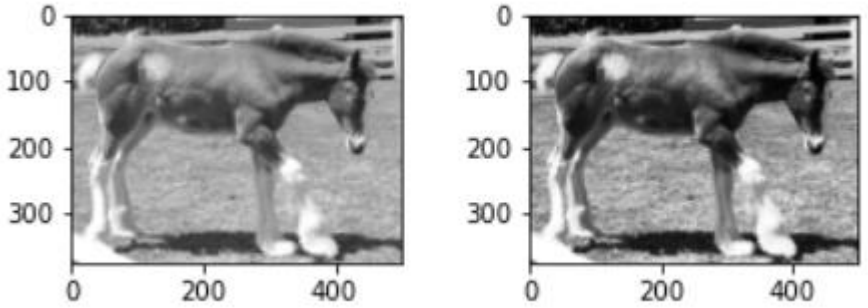
```python
def equalization(image):
    #计算概率密度
    h,w = image.shape
    num_px = h*w
    freq1 = get_histogram(image)
    #频率
    prob = freq1/num_px
    prob2 = []
    for i in range(len(freq1)-1):
        prob2.append(sum(prob[:i+1]))
    #最后一个肯定是1
    prob2.append(1)

    #像素值替换
    out = np.zeros((h,w),dtype = np.uint8)
    freq2 = np.round(np.array(prob2)*255)
    for i in range(h):
        for j in range(w):
            data = image[i][j]
            out[i][j] = freq2[data]
    return out

```

空间域增强——图像灰度直方图

- 灰度直方图均衡实现
  - 单通道均衡效果



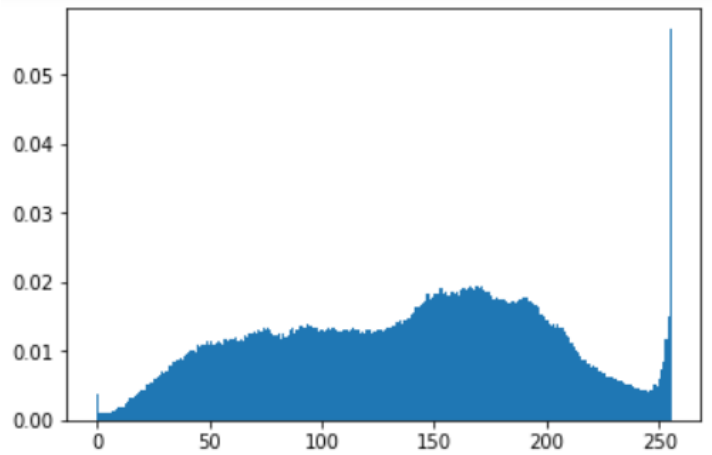


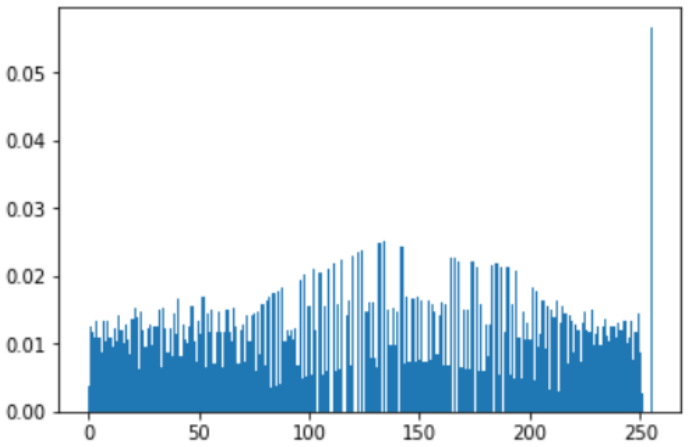


空间域增强——图像灰度直方图

- 灰度直方图均衡实现
  - 真彩色图像均衡：

三通道分别均衡然后融合





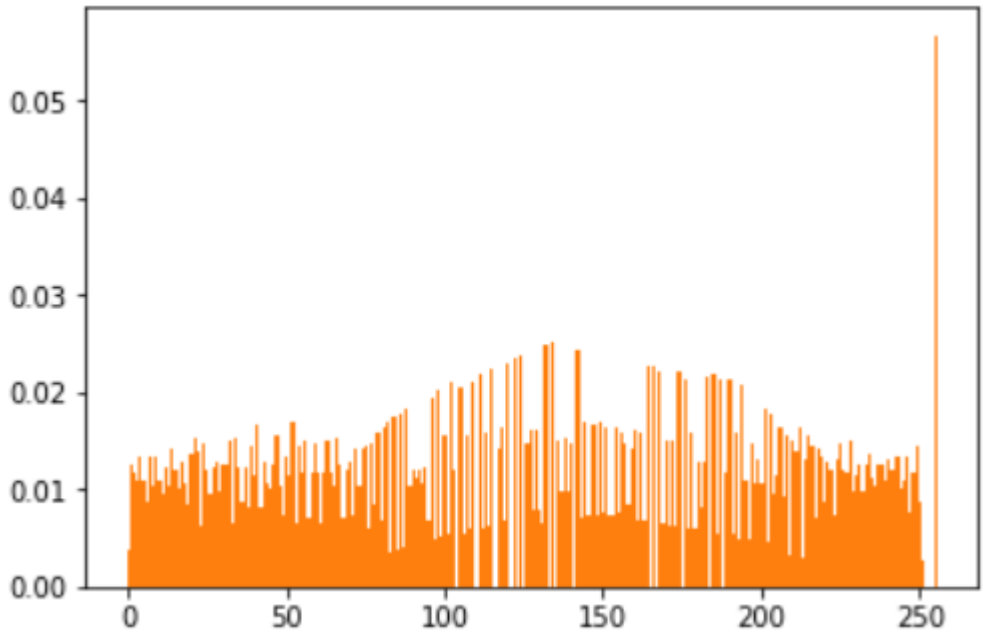
空间域增强——图像灰度直方图

- 灰度直方图均衡实现
  - 真彩色图像均衡效果

空间域增强——图像灰度直方图

- 灰度直方图均衡实现
  - 真彩色图像均衡效果

思考：对均衡后的图像再次进行均衡效果会怎样？

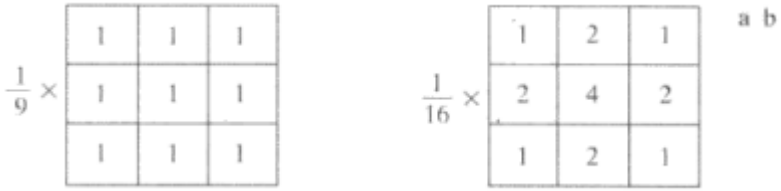


图像增强——滤波器

- 空域与频率域滤波器
  - 空间域和频域线性滤波的基础都是卷积定理：

$$f(x,y)*h(x,y) \Rightarrow H(u,v)F(u,v)$$

  - 空间域：h(x,y)为滤波器、滤波掩模、核、模板或窗口

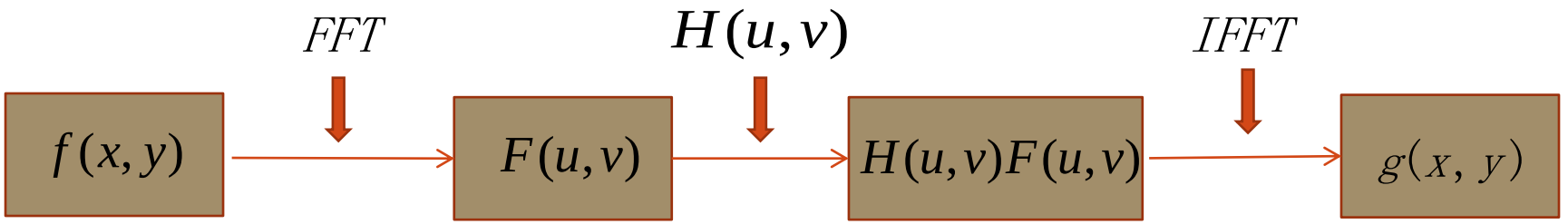


  - 频域：H(u,v)通常被称为滤波传递函数或者频响函数

$$H(u,v)=\begin{cases}1, & D(u,v)\le D_0\\ 0, & D(u,v)>D_0\end{cases} \qquad H(u,v)=\begin{cases}0 & D(u,v)\le D_0\\ 1 & D(u,v)>D_0\end{cases}$$

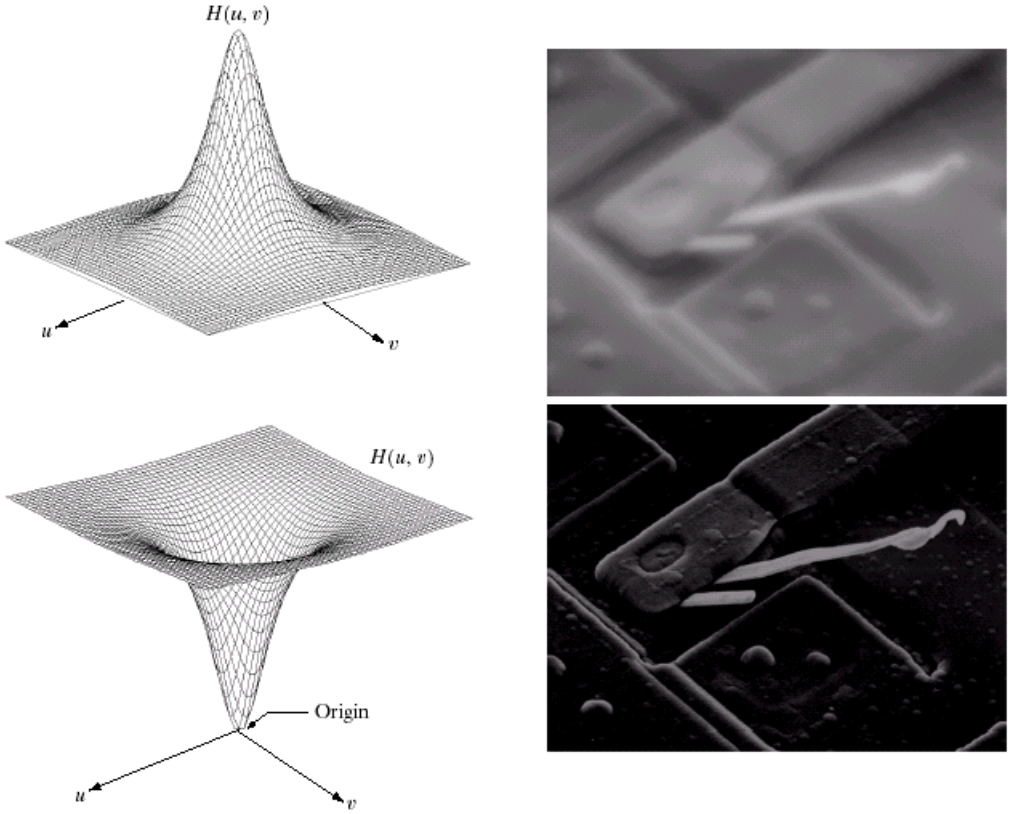
图像增强——频率域滤波器

- 频率域滤波器一般过程



图像增强——频率域滤波器

- 高通滤波器和低通滤波器



a b
c d

FIGURE 4.7 (a) A two-dimensional lowpass filter function. (b) Result of lowpass filtering the image in Fig. 4.4(a). (c) A two-dimensional highpass filter function. (d) Result of highpass filtering the image in Fig. 4.4(a).

➢ **低通滤波器**

抑制高频成分，消除噪声
边缘信息损失，图像边缘模糊

➢ **高通滤波器**

削弱低频成分，消除模糊
图像锐化，边缘信息突出

图像增强——频率域滤波器

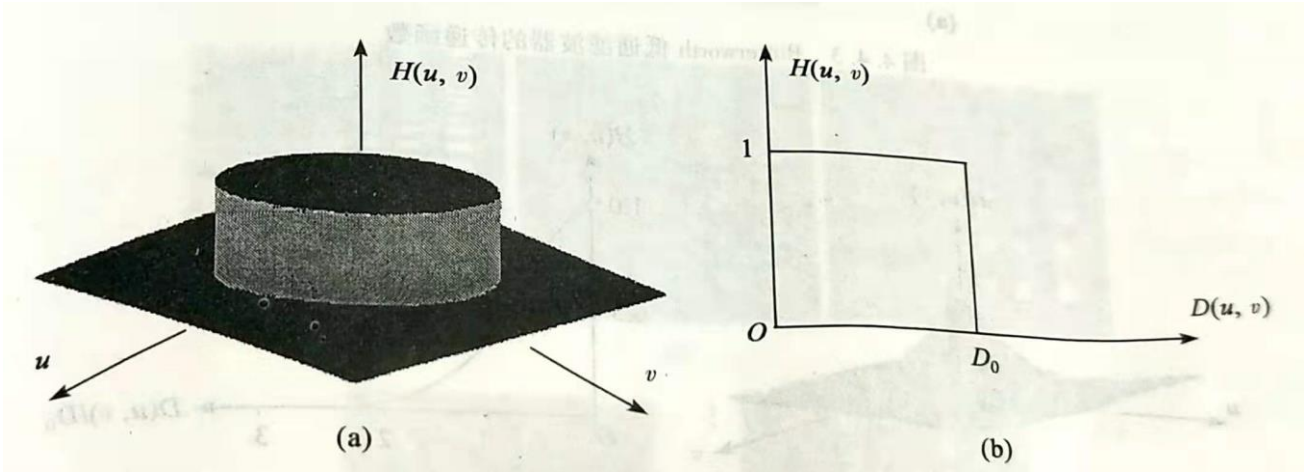
- 低通滤波
  - 去除噪声、图像平滑
  - 常用低通滤波器
    - **矩形滤波器（理想低通滤波器）**
    - **布特沃森（Butterworth）低通滤波器**
    - **高斯低通滤波器**
    - 梯形低通滤波器
    - 指数低通滤波器…

图像增强——频率域滤波器

- 理想低通滤波

理想低通滤波器的传递函数：

$$H(u,v)=\begin{cases}1, & D(u,v)\le D_0\\ 0, & D(u,v)>D_0\end{cases}$$



其中，D_0为截止频率，

$$D(u,v)=(u^2+v^2)^{\frac{1}{2}}$$

说明：理论上$H(u,v)$在D_0内的频率分量无损通过；而在 $D>D_0$的分量被除掉。然后经过傅里叶逆变换得到平滑图像，由于高频成分包含大量的边缘信息，因此采用该滤波器在去噪声的同时将会导致边缘信息损失而使图像边缘模糊，产生振铃效果、图像模糊。

图像增强——频率域滤波器

- 理想低通滤波

```python
def low_pass_mask(image,threshold):
    h,w = image.shape
    mask = np.zeros((h,w))
    for i in range(h):
        for j in range(w):
            #计算当前像素与图像中心距离
            d = np.sqrt((i-h//2)**2+(j-w//2)**2)
            if d <=threshold:
                mask[i][j] = 1
    return mask
```

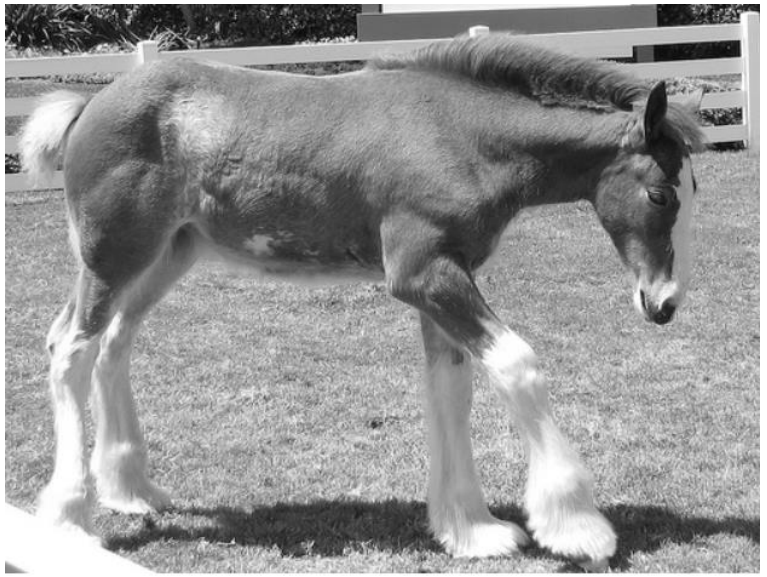
```python
def frequency_filter(image,mask):
    #图像快速傅里叶变换
    image_fft = np.fft.fft2(image)
    image_fft_shift = np.fft.fftshift(image_fft)
    #生成mask
    out = image_fft_shift.copy()
    #mask与原图位运算
    out = out * mask
    #傅里叶反变换
    out = np.fft.ifftshift(out)
    out = np.fft.ifft2(out)
    out = np.clip(out,0,255)
    out = np.abs(out).astype(np.uint8)
    return out
```

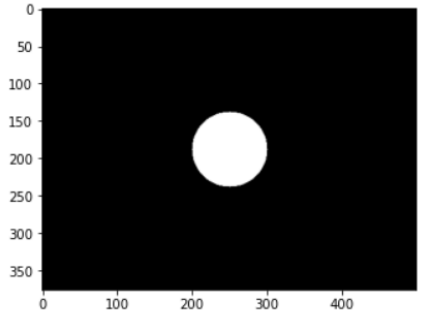
图像增强——频率域滤波器

- 理想低通滤波

```
1  mask = low_pass_mask(image_np,50)
2  out = frequency_filter(image_np,mask)
3  plt.imshow(mask,"gray")
4  Image.fromarray(out)
```

✓ 0.4s



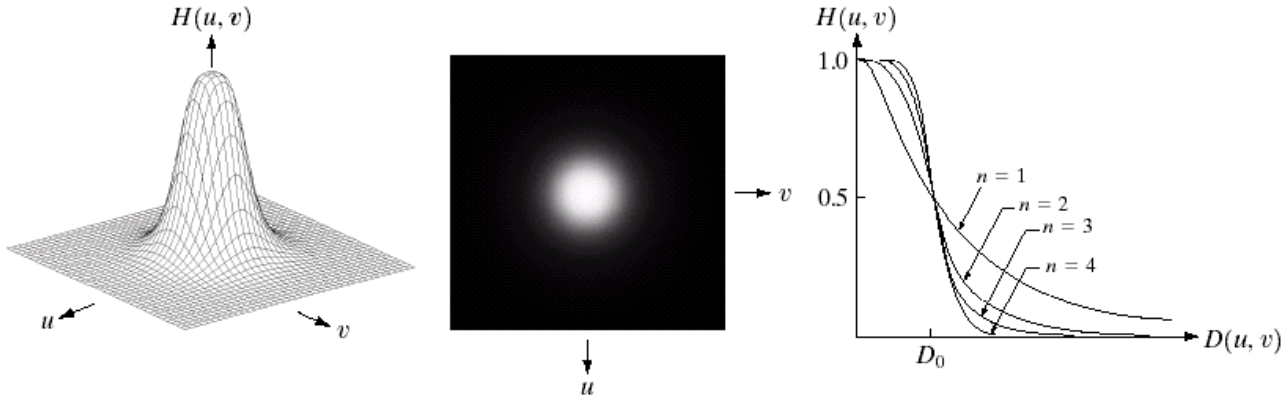


图像增强——频率域滤波器

- Butterworth低通滤波器

Butterworth低通滤波的传递函数如下：

$$H(u,v)=\frac{1}{1+\left[\frac{D(u,v)}{D_0}\right]^{2n}}$$



a b c

FIGURE 4.14 (a) Perspective plot of a Butterworth lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4.

图像增强——频率域滤波器

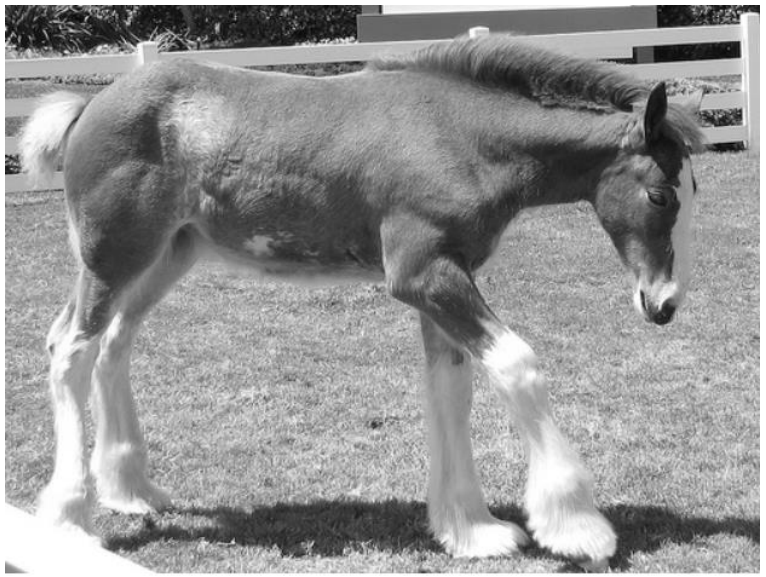
- Butterworth低通滤波器

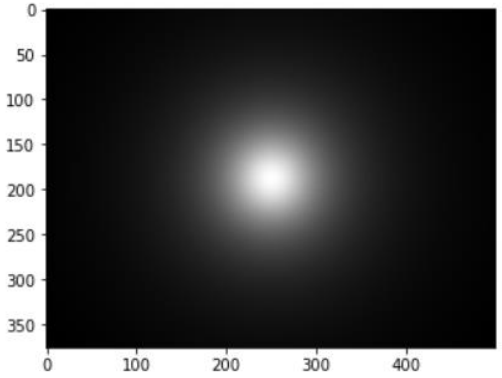
```
#巴通沃斯滤波器
def butterworth_low_pass_mask(image,cut_off,butterworth_order):
    h,w = image.shape
    mask = np.zeros((h,w))
    for i in range(h):
        for j in range(w):
            #计算当前像素与图像中心距离
            d = np.sqrt((i-h//2)**2+(j-w//2)**2)
            mask[i][j] = 1.0/(1.0+(d/cut_off)**(2*butterworth_order))
    return mask
```

```
mask = butterworth_low_pass_mask(image_np,50,1)
out = frequency_filter(image_np,mask)
plt.imshow(mask,"gray")
Image.fromarray(out)
```

图像增强——频率域滤波器

- Butterworth低通滤波器



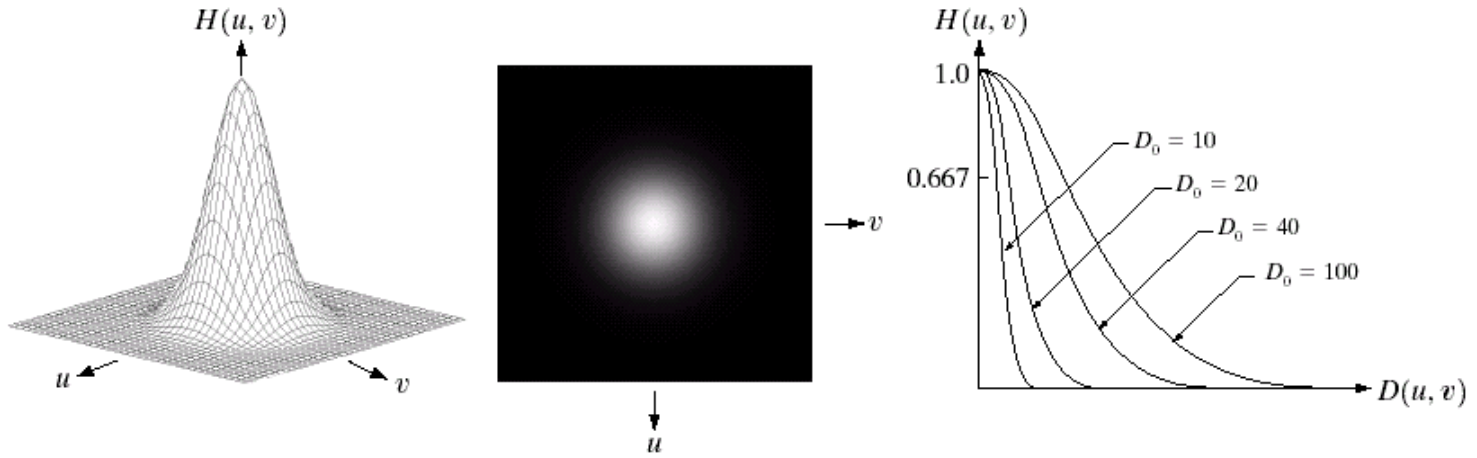


图像增强——频率域滤波器

- 高斯低通滤波器

高斯低通滤波的传递函数如下：

$$H(u,v) = e^{\frac{-D^2(u,v)}{2D_0^2}}$$

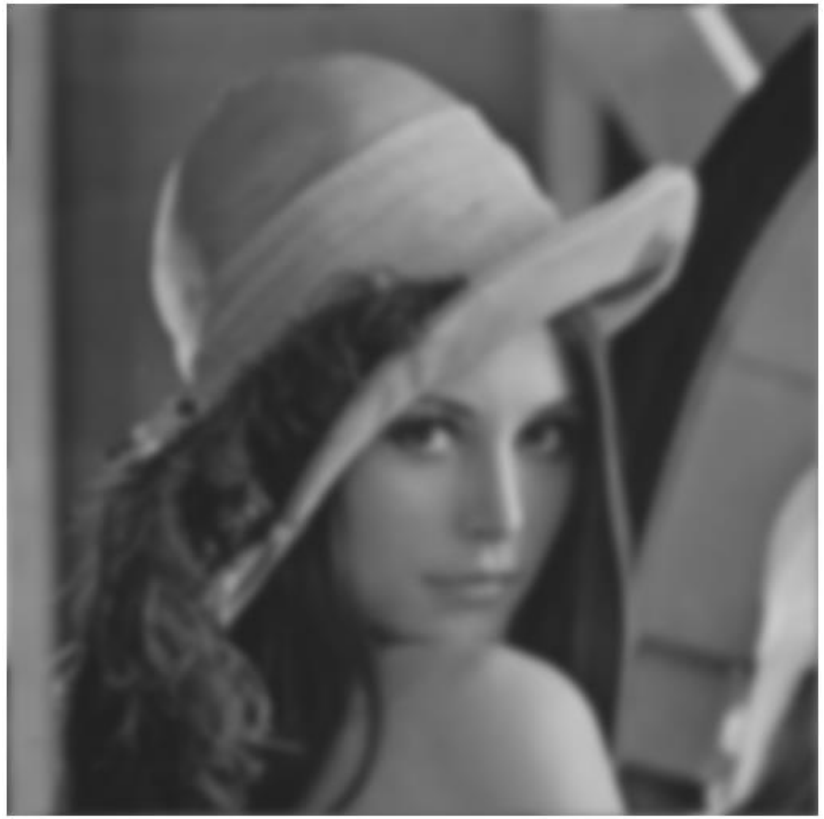


a b c

FIGURE 4.17 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

图像增强——频率域滤波器

- 高斯低通滤波器

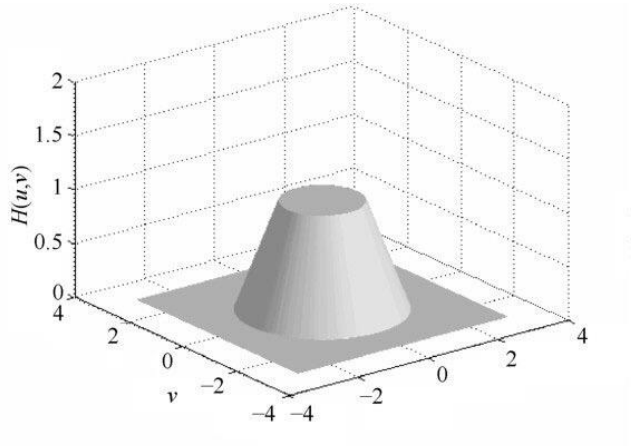


图像增强——频率域滤波器

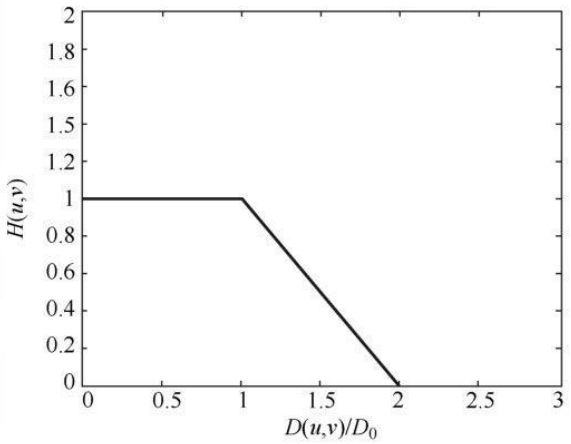
- 梯形低通滤波器

梯形低通滤波的传递函数如下：

$$H(u,v)=\begin{cases}1 & D(u,v)<D_0\\ \frac{D(u,v)-D_1}{D_0-D_1} & D_0<D(u,v)\le D_1\\ 0 & D(u,v)>D_1\end{cases}$$



（a）三维图形



（b）剖面图形

图像增强——频率域滤波器

- 梯形低通滤波器

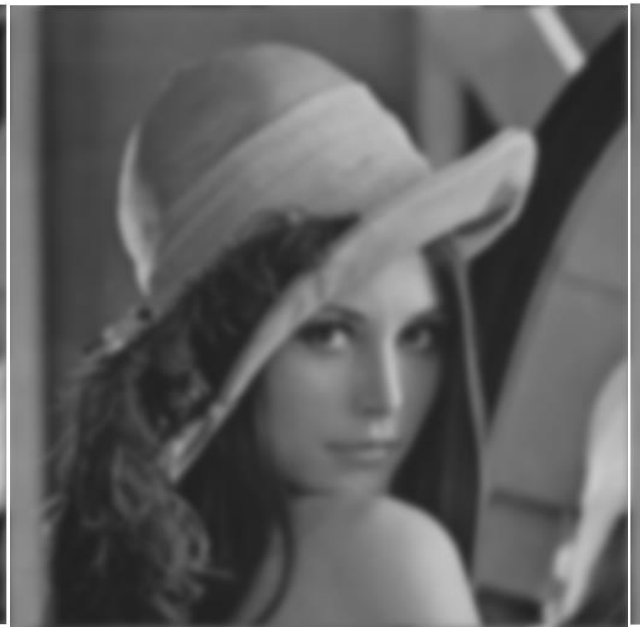
图像增强——频率域滤波器

- 低通滤波器比较

D0=25

理想低通滤波器

梯形低通滤波器



高斯低通滤波器

Butterworth低通滤波器

图像增强——频率域滤波器

- 高通滤波
  - 噪声提取、图像锐化
  - 常用高通滤波器
    - **理想高通滤波器**
    - **布特沃森（Butterworth）高通滤波器**
    - **高斯高通滤波器**
    - 梯形高通滤波器
    - 指数高通滤波器…

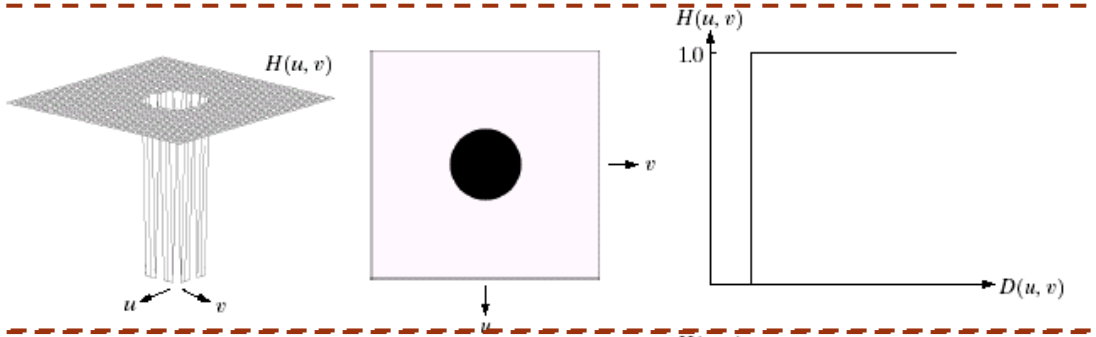
图像增强——频率域滤波器

- 高通滤波

透视图　　图像　　剖面图

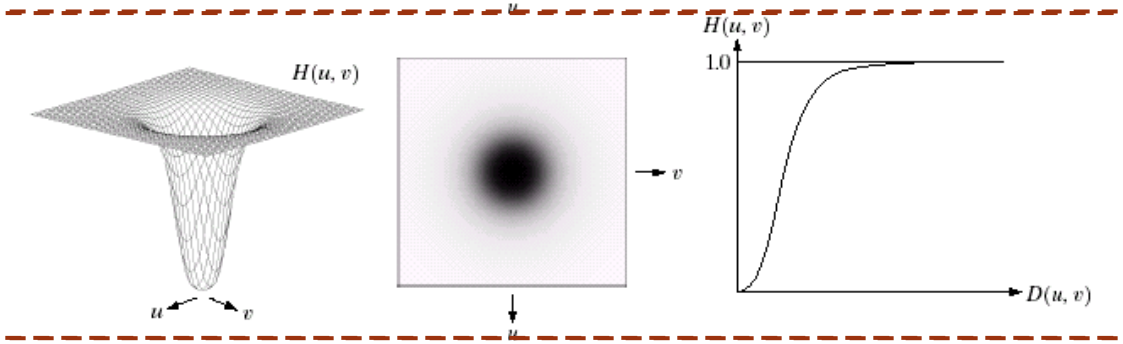
➢ 理想高通滤波器

$$H(u,v)=\begin{cases}0 \text{ if } D(u,v)\le D_0\\ 1 \text{ if } D(u,v)>D_0\end{cases}$$



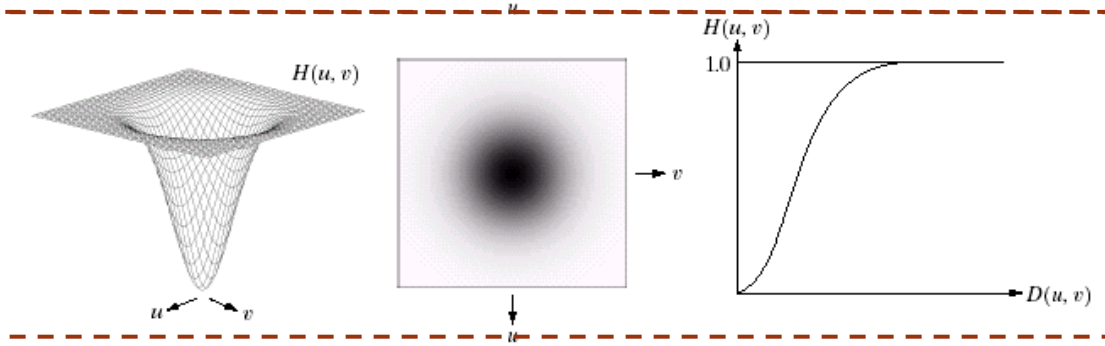
➢ 巴特沃斯高通滤波器

$$H(u,v)=\frac{1}{1+\left[D_0/D(u,v)\right]^{2n}}$$



➢ 高斯高通滤波器

$$H(u,v)=1-e^{-D^2(u,v)/2D_0^2}$$

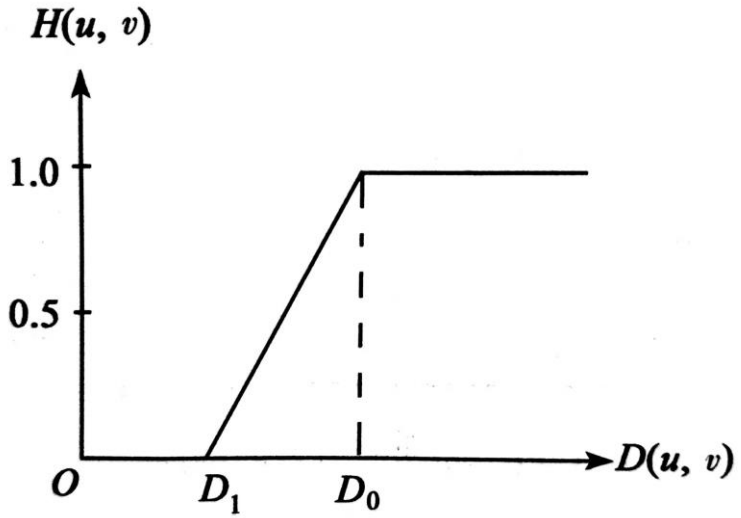


图像增强——频率域滤波器

- 高通滤波
  - 梯形高通滤波器

    梯形高通滤波的传递函数如下：

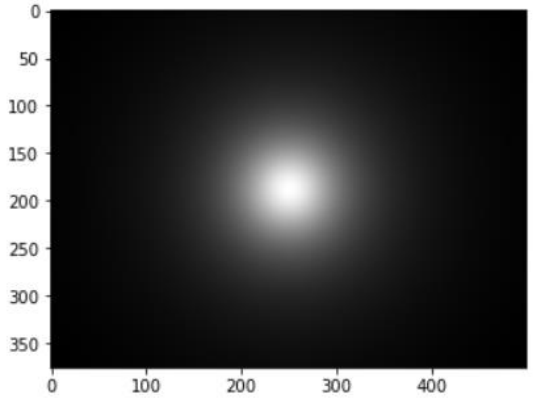
$$H(u,v)=\begin{cases}0, & D(u,v)<D_0\\ \dfrac{D(u,v)-D_0}{D_1-D_0}, & D_0\leqslant D(u,v)\leqslant D_1\\ 1, & D(u,v)>D_1\end{cases}$$

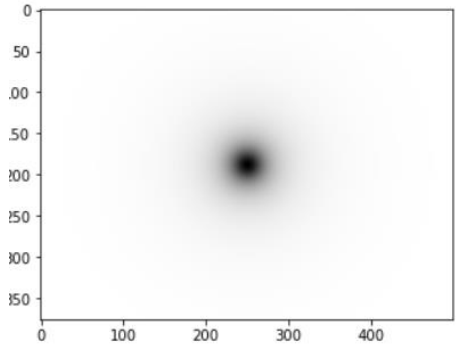


梯形高通滤波剖面图

图像增强——频率域滤波器

- Butterworth高通滤波器



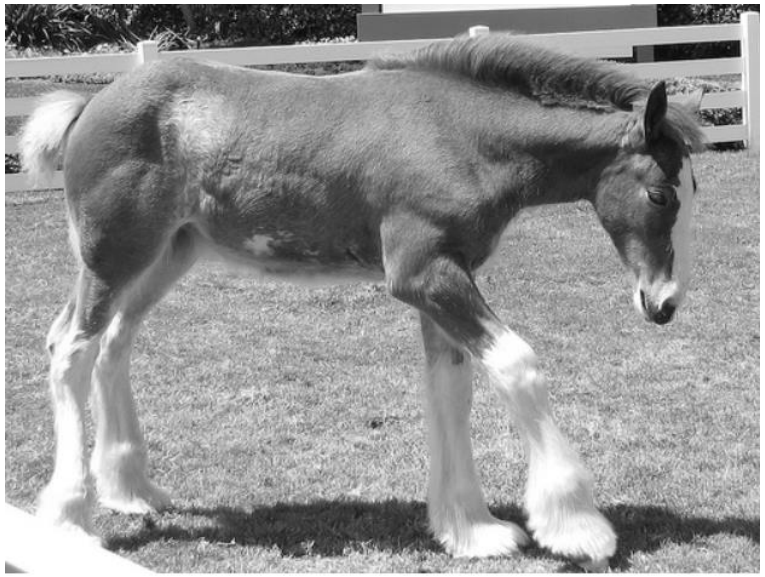



```python
#输入图像和滤波掩膜
def frequency_filter(image,mask,mode = "low"):
    #图像快速傅里叶变换
    image_fft = np.fft.fft2(image)
    image_fft_shift = np.fft.fftshift(image_fft)
    #生成mask
    if mode == "low":
        pass
    if mode == "high":
        mask = 1 - mask
    out = image_fft_shift.copy()
    #mask与原图位运算
    out = out * mask
    #傅里叶反变换
    out = np.fft.ifftshift(out)
    out = np.fft.ifft2(out)
    out = np.clip(out,0,255)
    out = np.abs(out).astype(np.uint8)
    return out
```

✓ 0.3s

图像增强——频率域滤波器

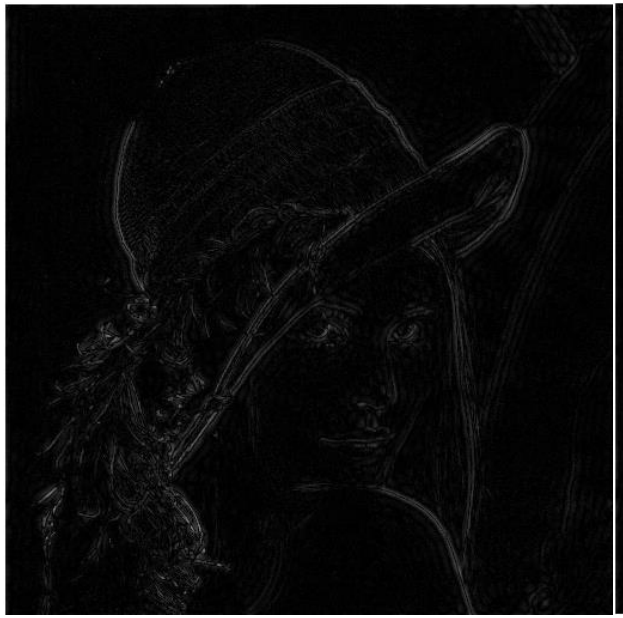
- Butterworth高通滤波器



图像增强——频率域滤波器

- 高通滤波器比较

D0=35



理想高通滤波

梯形高通滤波

高斯高通滤波

Butterworth高通滤波

图像增强——频率域滤波器

- 带通/带阻滤波器
  - 带阻滤波器

    高频和低频成分通过，中频成分不通过
  - 带通滤波器

    中频成分通过，低频与高频成分不通过

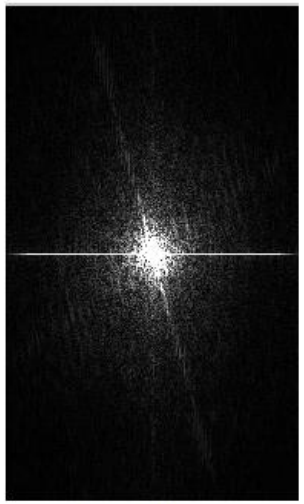


图 1 频谱图

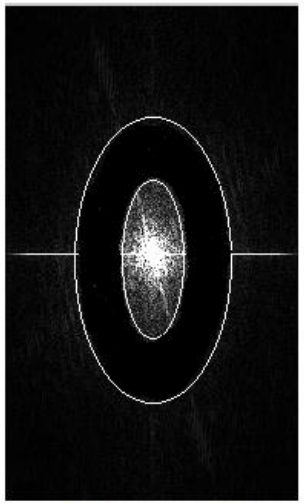


图 2 带阻滤波器后的频谱图

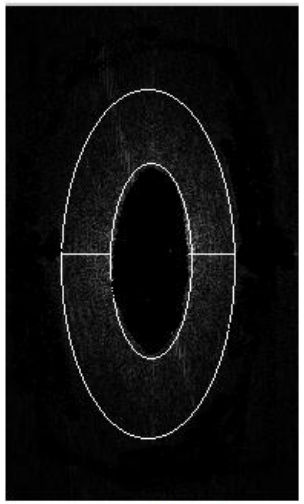


图 3 带通滤波器后的频谱图

图像增强——频率域滤波器

- 高斯带通/带阻滤波器
  - 高斯带阻滤波传递函数

$$H(u,v) = 1 - e^{-\frac{1}{2}\left[\frac{D^2(u,v)-D_0^2}{D(u,v)W}\right]^2}$$

  - 高斯带通滤波传递函数

$$H(u,v) = e^{-\frac{1}{2}\left[\frac{D^2(u,v)-D_0^2}{D(u,v)W}\right]^2}$$

图像增强——频率域滤波器

- 高斯带通/带阻滤波器

```
def gaussian_bandstop_kernel(img,D0,W):
    height,width = img.shape
    low_kernel = np.zeros(shape=img.shape)
    for u in range(height):
        for v in range(width):
            D = np.sqrt( (u-height/2)**2 + (v-width/2)**2 )
            low_kernel[u][v]=1.0 - np.exp(-0.5 * (((D ** 2 - D0**2) / (D *W + 1.0e-5))**2))
    return low_kernel
```

```
def gaussian_bandstop_filter(img,D0,W):
    assert img.ndim == 2
    kernel = gaussian_bandstop_kernel(img,D0,W)
    gray = np.float64(img)
    gray_fft = np.fft.fft2(gray)
    gray_fftshift = np.fft.fftshift(gray_fft)
    dst = np.zeros_like(gray_fftshift)
    dst_filtered = kernel * gray_fftshift
    dst_ifftshift = np.fft.ifftshift(dst_filtered)
    dst_ifft = np.fft.ifft2(dst_ifftshift)
    dst = np.abs(np.real(dst_ifft))
    dst = np.clip(dst,0,255)
    return np.uint8(dst)
```

```
def gaussian_bandpass_filter(img,D0,W):
    assert img.ndim == 2
    kernel = 1.0 - gaussian_bandstop_kernel(img,D0,W)
    gray = np.float64(img)
    gray_fft = np.fft.fft2(gray)
    gray_fftshift = np.fft.fftshift(gray_fft)
    dst = np.zeros_like(gray_fftshift)
    dst_filtered = kernel * gray_fftshift
    dst_ifftshift = np.fft.ifftshift(dst_filtered)
    dst_ifft = np.fft.ifft2(dst_ifftshift)
    dst = np.abs(np.real(dst_ifft))
    dst = np.clip(dst,0,255)
    return np.uint8(dst)
```

图像增强——频率域滤波器

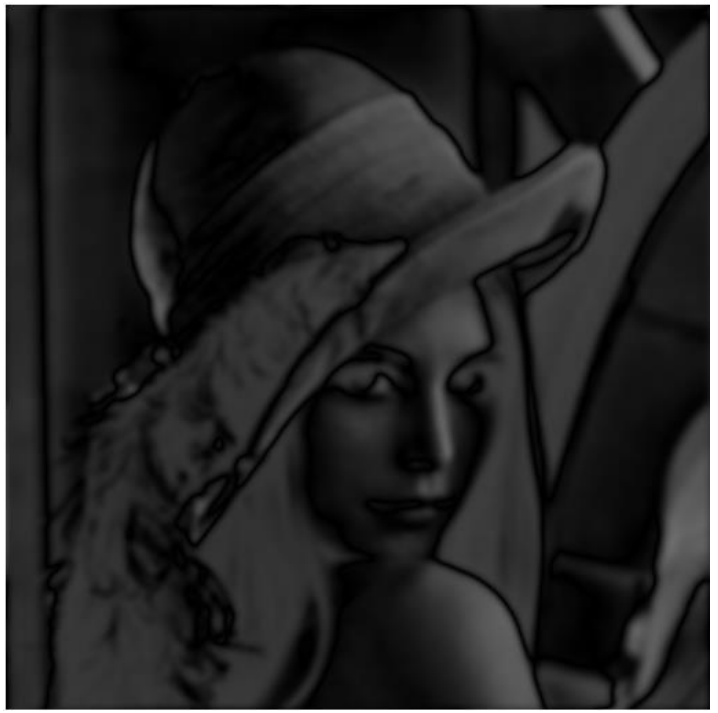
- 高斯带阻滤波器

```
Image.fromarray(gaussian_bandstop_filter(np.array(img_PIL),5,15)).show()
Image.fromarray(gaussian_bandstop_filter(np.array(img_PIL),5,25)).show()
```

图像增强——频率域滤波器

- 高斯带通滤波器

```
Image.fromarray(gaussian_bandpass_filter(np.array(img_PIL),5,15)).show()
Image.fromarray(gaussian_bandpass_filter(np.array(img_PIL),5,30)).show()
```



图像增强——动手实践

- **空间域图像增强**

  - 对图像进行灰度直方图信息查看；
  - 灰度直方图拉伸，通过拉伸后的图像的灰度PDF以及增强图像体会拉伸效果；
  - 进行图像均衡，通过对灰度图像以及原图均衡，体会均衡效果；

- **频率域图像增强**

  - 对不同滤波器采用相同参数，体会不同滤波器的图像增强效果；
  - 同一种滤波器改变其参数进行图像增强，体会不同参数对增强效果的影响；