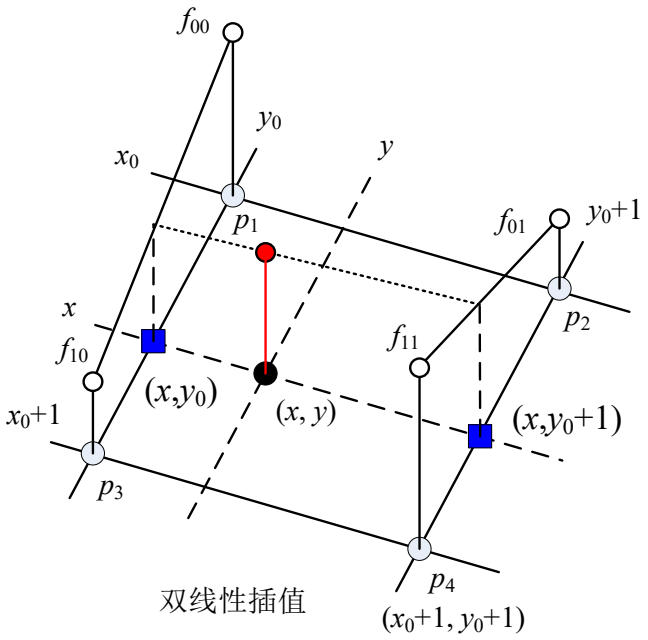
练习：灰度插值

在某256阶灰度图像的几何变换计算中，需计算 (221.3, 396.7) 这一位置的像素灰度值。

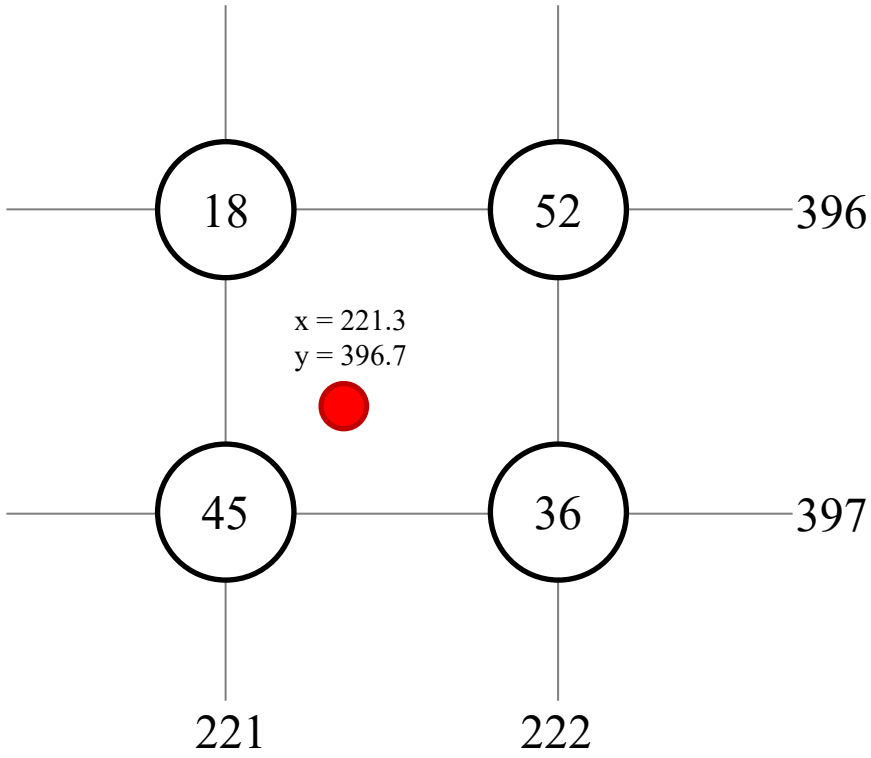
现已知：$f(221, 396) = 18, f(221, 397) = 45, f(222, 396) = 52, f(222, 397) = 36$，请用①最近邻插值、②双线性插值两种方法，计算 $f(221.3, 396.7)$ 。

$$\begin{aligned}\hat{f}(x,y) = {} & (x_0+1-x)(y_0+1-y)\cdot f_{00} \\ & +(x-x_0)(y_0+1-y)\cdot f_{10} \\ & +(x_0+1-x)(y-y_0)\cdot f_{01} \\ & +(x-x_0)(y-y_0)\cdot f_{11}\end{aligned}$$



双线性插值

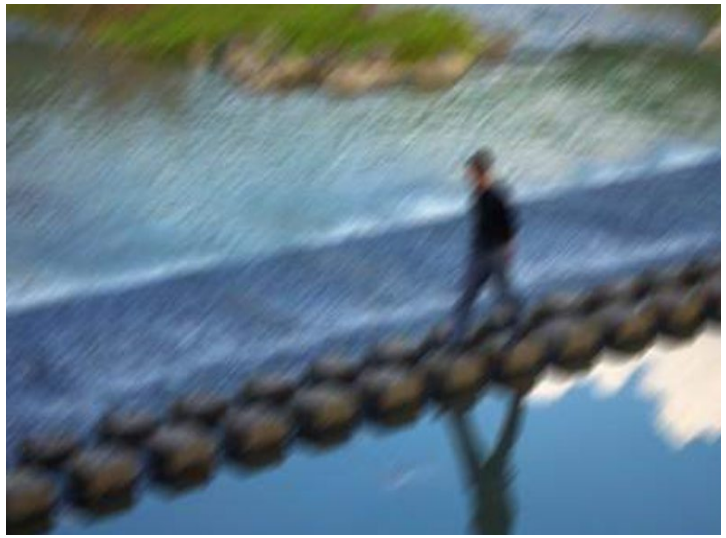
“对角瞭望法”

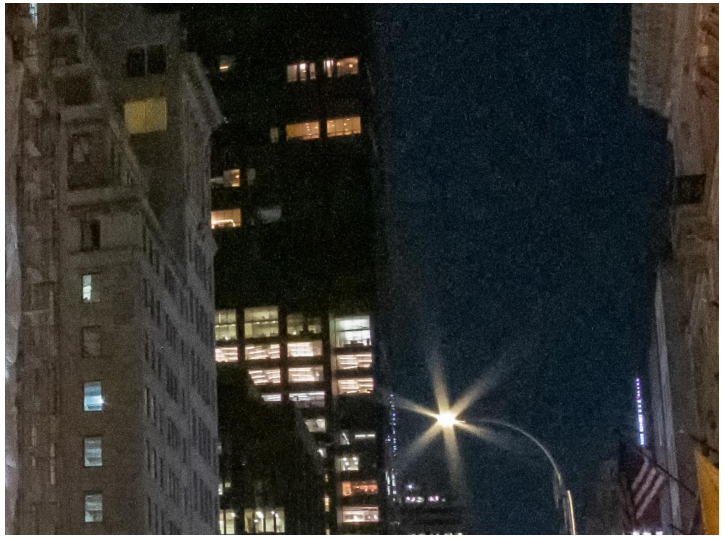


第7章
图像复原

目录

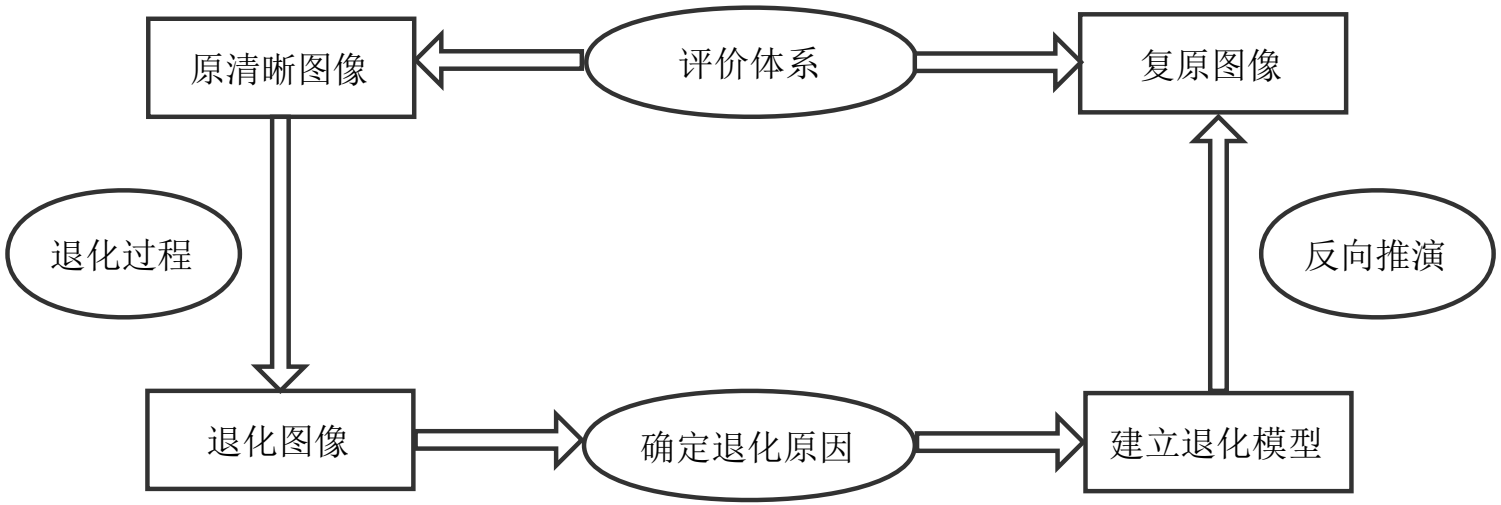
背景知识





背景知识

- **图像退化：成像过程中，可能会因传感器噪声、照相机镜头失焦、照相机与目标之间的相对运动、航空拍摄时大气湍流的随机扰动、水下拍摄时水流的干扰、雾霾等原因导致图像模糊，称为图像退化（Image degradation）。**
  - 图像退化将影响后继处理过程，增加图像计算、分析、特征提取及目标识别的难度，降低图像数据的应用价值。
- **图像复原：根据图像退化过程的先验知识，建立图像退化过程的数学模型，对退化图像进行修复或者重建，称为图像复原（Image restoration）。**

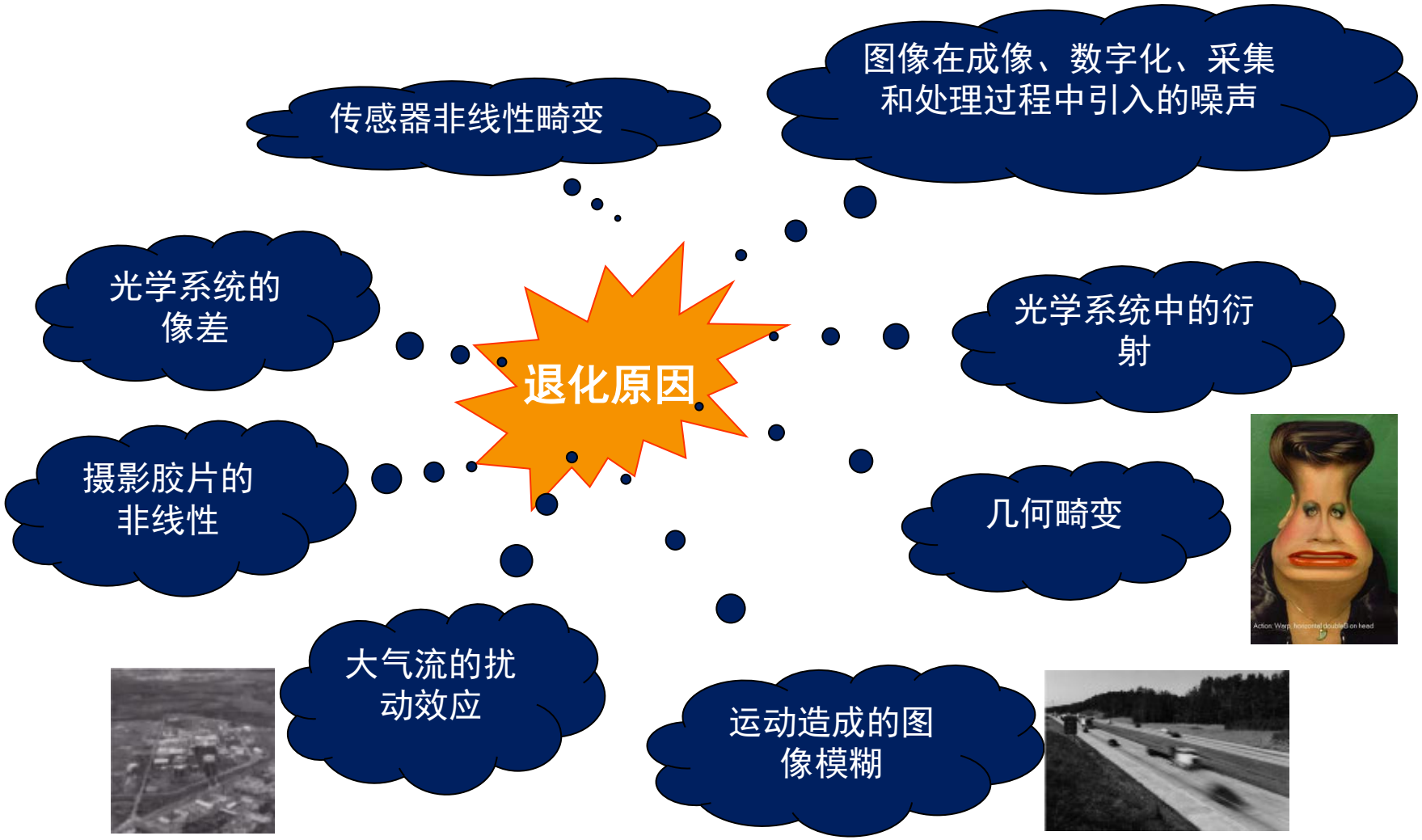


辨析：图像复原与图像增强的比较

- **Image enhancement : process image so that the result is more suitable for a specific application, is largely a subjective process.**
- **图像增强：基本上是一个探索性过程，为了人类视觉系统的生理接受特点而设计一种改善图像的方法。**
- **Image restoration : recover image from distortions to its original image, is largely an objective process.**
- **图像复原：试图利用退化现象的某种先验知识来重建或复原被退化的图像。因而，图像复原就是建立退化过程的模型，然后采用相反的过程进行处理，以恢复原图像。**

7.1 图像退化过程及其模型化方法

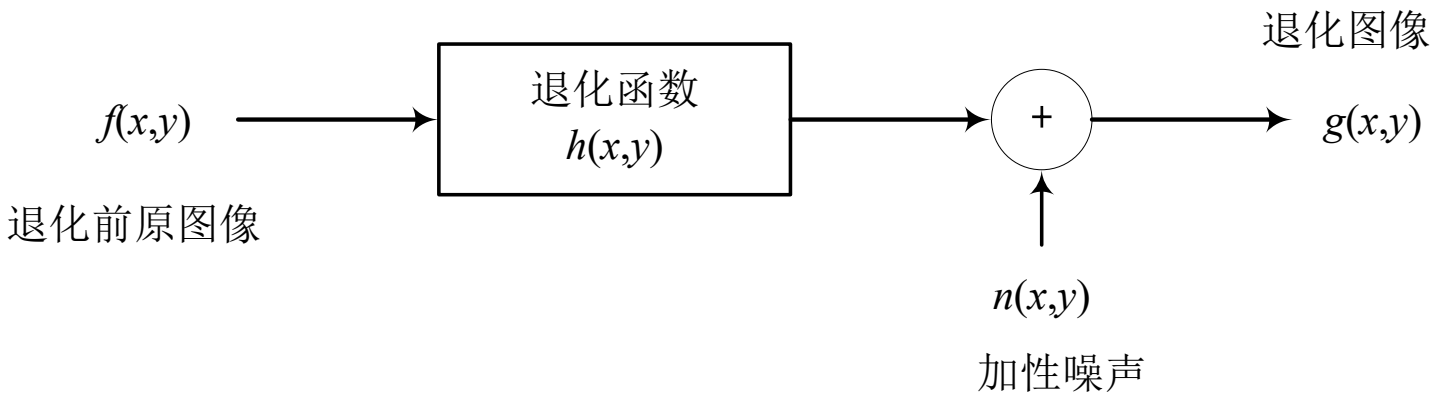
图像退化原因


退化原因
传感器非线性畸变
图像在成像、数字化、采集和处理过程中引入的噪声
光学系统的像差
光学系统中的衍射
摄影胶片的非线性
几何畸变
大气流的扰动效应
运动造成的图像模糊

图像退化模型

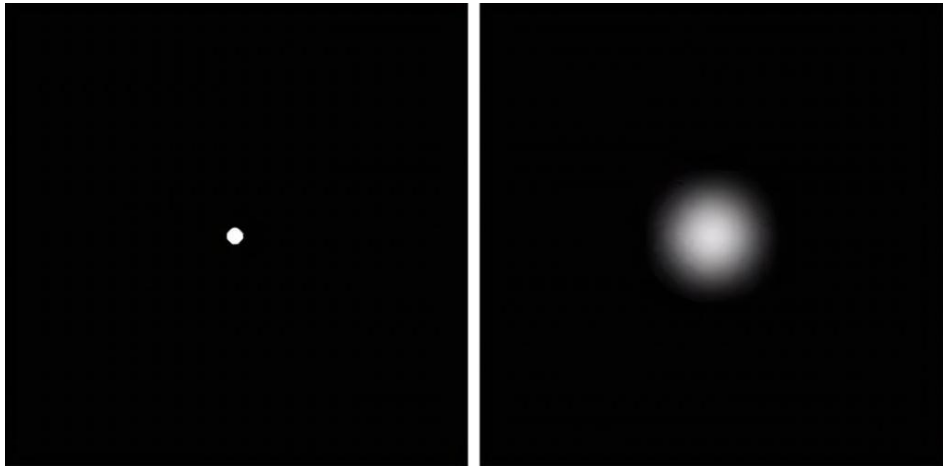
- 假设图像$f(x,y)$经退化过程$h(x,y)$及加性噪声$n(x,y)$的共同作用，产生退化图像$g(x,y)$。假定这个退化过程是一个**线性移不变系统（Linear Shift-invariant System）**，其退化函数可以用二维系统的单位冲激响应$h(x,y)$来表征，那么$f(x,y)$的退化过程可表示为以下线性卷积形式：

$$g(x,y)=h(x,y)*f(x,y)+\eta(x,y)$$
$$G(u,v)=H(u,v)F(u,v)+N(u,v)$$



图像退化模型

- **点扩散函数：光学中的单位冲激函数是一个光点，通过光学系统后会扩散为一个模糊光斑，其模糊程度由光学部件的质量决定，通常把光学系统的单位冲激响应$h(x,y)$称为点扩散函数（PSF，Point Spread Function）。**
- **图像退化过程是理想图像与退化函数的卷积（convolution），因此，图像复原又常被称为去卷积、解卷积、反卷积（deconvolution）。**



光点扩散现象

线性移不变系统

$$g(x,y)=H\left[f(x,y)\right]+\eta(x,y)$$

线性系统： $H\left[af_1(x,y)+bf_2(x,y)\right]=aH\left[f_1(x,y)\right]+bH\left[f_2(x,y)\right]$

线性系统： 可加性、齐次性

位置不变： $H\left[f(x-\alpha,y-\beta)\right]=g(x-\alpha,y-\beta)$

图像任意一点的响应只取决于该点的输入值，而与该点的位置无关。

退化函数的估计方法

- 观察法 (*observation*)
- 试验法 (*experimentation*)
- 数学建模法 (*mathematical modeling*)

观察法

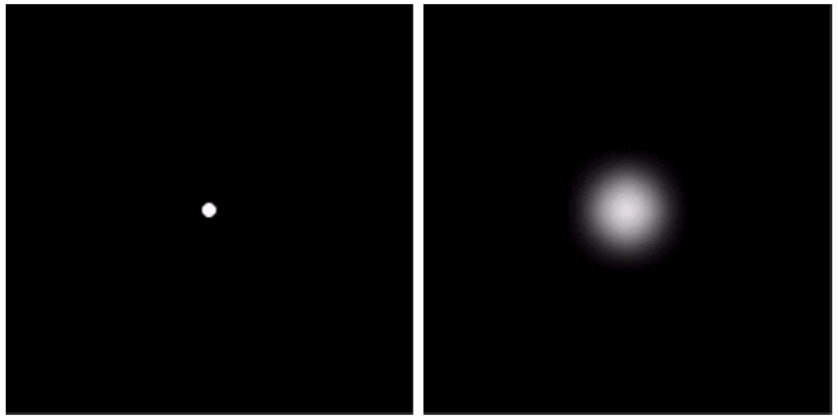
- 假设$g(x,y)$是一幅退化图像，但没有提供退化函数H的知识，那么估计该函数的一个方法就是收集图像自身的信息。
- 例如，如果图像是模糊的，可以观察包含简单结构的一小部分图像，例如某一物体和背景的一部分。为了减少观察时的噪声影响，可以寻找强信号内容区。
- 使用目标和背景的样品灰度级，可以构建一个不模糊的图像，该图像和看到的子图像有相同大小和特性。
- 令$g_s(x,y)$表示观察到的子图像，用$\hat{f}_s(x,y)$表示对其复原的子图像(原始图像在该区域的估计图像)。假定噪声效果可忽略（由于选择了一强信号区）则有：

$$H_s(u,v)=\frac{G_s(u,v)}{\hat{F}_s(u,v)}$$

试验法

- 如果可以使用与获取退化图像的设备相似的装置，理论上可以得到一个准确的退化估计。
- 改变系统设置，使系统处于产生尽可能接近希望复原的退化图像的工作状态。
- 利用相同的系统设置，由成像一个脉冲(小亮点)得到退化的冲激响应。一个冲激可由明亮的亮点来模拟，并使它尽可能亮以减少噪声的干扰。
- 冲激的傅里叶变换是一个常量，系统的点扩散函数(PSF)为：

$$H(u,v)=\frac{G(u,v)}{A}$$



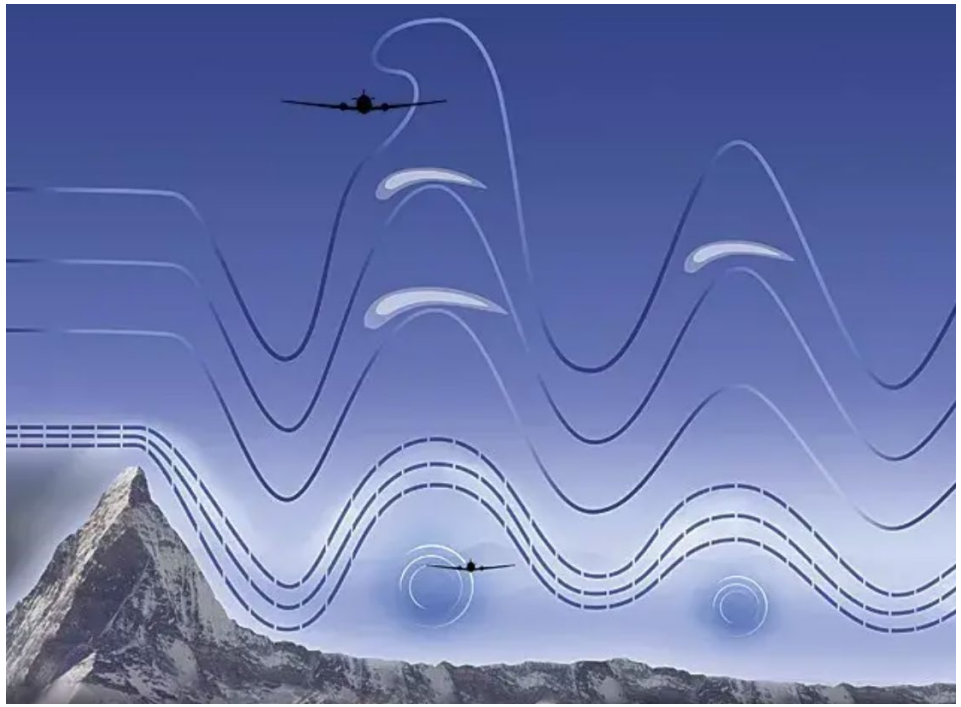
数学机理建模法

- **大气湍流图像模糊退化模型：大气作为光学成像过程中光的传输介质，受外界诸多因素影响，导致对光的折射率发生变化。当光线通过不均匀传输介质时会发生折射或衍射，由光线会聚而成的像点将发生偏移，导致图像模糊。其点扩散函数PSF的傅里叶变换为：**

$$H(u,v)=\mathrm{e}^{-k\left(u^2+v^2\right)^{5/6}}$$

  - 式中，k为大气湍流常数，用于控制模糊退化程度，k越大，大气湍流越剧烈，导致图像越模糊。

数学机理建模法



示例：模拟大气湍流的图像模糊退化过程

```
#采用自定义函数AtmoTurbulenceSim模拟大气湍流模糊退化
img = io.imread('./imagedata/aerial_image.png')   #读入一幅航拍图片
imgout1, Hatm1 = AtmoTurbulenceSim(img, 0.001)   #令k=0.001
imgout2, Hatm2 = AtmoTurbulenceSim(img, 0.0025)  #令k=0.0025
#显示退化结果（略，详见本章Jupyter Notebook可执行笔记本文件）
```

(1)原图像

(2)中等湍流, k=0.001

(3)剧烈湍流, k=0.0025

模拟大气湍流图像模糊退化函数 AtmoTurbulenceSim

```
def AtmoTurbulenceSim(image,k):
    """
    模拟大气湍流图像模糊退化AtmoTurbulenceSim
    输入参数:
          image - 原图像, 灰度图像或RGB彩色图像;
                k - 大气湍流模型的系数,k越大,大气湍流越剧烈,导致图像越模糊;
    输出:    imgout - 大气湍流影响的退化图像;
                 Hatm -大气湍流模糊退化函数;
    """
    rows, cols = image.shape[0:2]  #获取图像高/宽
    #采用'reflect'方式扩展图像(下面扩展rows行,右面扩张cols列)
    if image.ndim==3:  #彩色图像
        imgex = np.pad(image,((0,rows),(0,cols),(0,0)),mode='reflect')
    elif image.ndim==2:  #灰度图像
        imgex = np.pad(image,((0,rows),(0,cols)),mode='reflect')
    #计算扩展图像的DFT并中心化
    img_dft = fftshift(fft2(imgex,axes=(0,1)),axes=(0,1))
```

模拟大气湍流图像模糊退化函数 AtmoTurbulenceSim

```
#生成大气湍流模糊退化函数
#构建频域平面坐标网格数组，坐标轴定义v列向/u行向
v = np.arange(-cols, cols)
u = np.arange(-rows, rows)
Va, Ua = np.meshgrid(v, u)
D2 = Ua ** 2 + Va ** 2
Hatm = np.exp(-k *(D2 ** (5.0/6.0)))
if image.ndim==3:
    Hatm = np.dstack(( Hatm, Hatm, Hatm))  #彩色图像把H串接成三维数组
#计算图像DFT与大气湍流模糊退化函数的点积
Gp= img_dft * Hatm
Gp = ifftshift(Gp,axes=(0,1))              #去中心化
imgp = np.real(ifft2(Gp,axes=(0,1)))   #DFT反变换并取实部
imgp = np.uint8(np.clip(imgp,0,255))  #把输出图像的数据格式转换为uint8
#截取imgp左上角与原图像大小相等的区域作为输出
imgout = imgp[0:rows,0:cols]
return imgout, Hatm
```

运动模糊图像退化模型

- 采集图像时，因曝光时间内成像设备与被摄物体或场景间发生相对运动，使物体像点在图像传感器靶面上发生移位进而导致图像模糊，称为图像运动模糊。根据图像运动模糊的生成机理将其划分为局部运动模糊、全局运动模糊和混合运动模糊三类。
- 假设场景在传感器靶面上沿水平和垂直方向做匀速直线运动，成像设备的曝光时间用T表示，在曝光时间T内像点在水平和垂直方向上的移动量分别用a和b分别表示，那么该情况下的运动模糊退化点扩散函数PSF的傅里叶变换为：

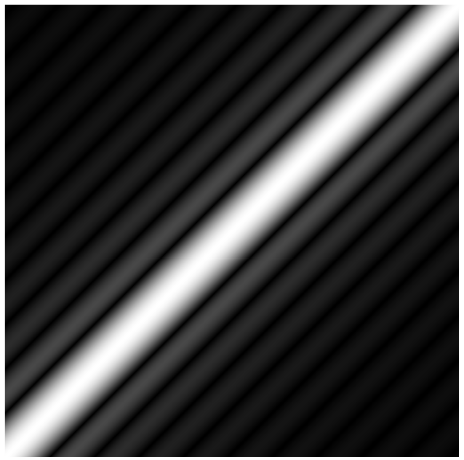
$$H(u,v)=\frac{T\sin[\pi(ua+vb)]}{\pi(ua+vb)}\mathrm{e}^{-\mathrm{j}\pi(ua+vb)}$$

示例：匀速直线运动引起的图像模糊

```
#采用自定义函数MotionBlurSim模拟匀速直线运动图像模糊退化
img = io.imread('./imagedata/cameraman.tif')  #读入一幅图片
#运动模糊退化参数
Te = 1  #曝光时间
xa = 0.02; yb = 0.02  #运动速度
imgout, Hmb = MotionBlurSim(img,Te, xa, yb)
#显示退化结果（略）
```

(1)原图像

(2)匀速直线运动模糊退化结果, T=1, a=b=0.02



(3)匀速直线运动模糊退化函数的幅度谱

噪声模型-图像噪声的成因

- **噪声主要来源于两个方面：**
  - 图像的获取过程，图像传感器CCD和CMOS采集图像过程中，由于受传感器材料属性、工作环境、电子元器件和电路结构等影响，会引入各种噪声，如电阻引起的热噪声、场效应管的沟道热噪声、光子噪声、暗电流噪声、光响应非均匀性噪声。
  - 图像传输过程，由于传输介质和记录设备等的不完善，数字图像在其传输过程中往往会受到多种噪声的污染。另外，在图像处理的某些环节也会引入噪声。

噪声模型-图像噪声的特征

- 噪声在图像中的分布和大小不规则，即具有随机性。
- 噪声与图像之间一般具有相关性。例如，图像黑暗部分噪声大，明亮部分噪声小。又如，数字图像中的量化噪声与图像相位相关，图像内容接近平坦时，量化噪声呈现伪轮廓，但图像中的随机噪声会因为颤噪效应反而使量化噪声变得不很明显。
- 噪声具有叠加性。在串联图像传输系统中，各部分窜入噪声若是同类噪声可以进行功率相加，依次信噪比要下降。

噪声模型-图像噪声的分类

● 加性噪声与乘性噪声

- 假定信号为$S(t)$，噪声为$n(t)$，如果混合叠加波形是$S(t)+n(t)$的形式，则称其为加性噪声。加性嗓声和图像信号强度是不相关的，如图像在传输过程中引进的“信道噪声”、电视摄像机扫描图像的噪声等。
- 如果叠加波形为$S(t)$［$1+n(t)$］的形式，则称其为乘性噪声。乘性噪声则与信号强度有关，往往随图像信号的变化而变化，如电视扫描光栅、胶片颗粒造成等。

● 外部噪声与内部噪声

- 按照产生原因，图像噪声可分为外部噪声和内部噪声。外部噪声，即指系统外部干扰以电磁波或经电源串进系统内部而引起的噪声。如外部电气设备产生的电磁波干扰、天体放电产生的脉冲干扰等。由系统电气设备内部引起的噪声为内部噪声，如内部电路的相互干扰。

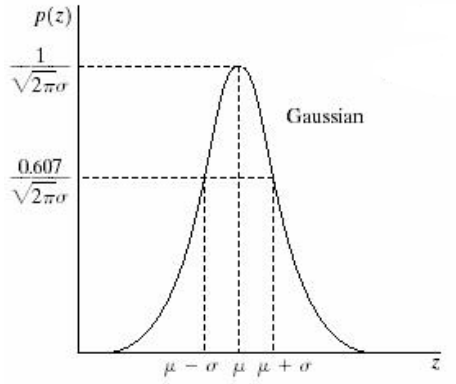
● 平稳噪声与非平稳噪声

- 按照统计特性，图像噪声可分为平稳噪声和非平稳噪声。统计特性不随时间变化的噪声称为平稳噪声。统计特性随时间变化的噪声称为非平稳噪声。

一些重要噪声的概率密度函数

◆ 高斯噪声(Gaussian noise)

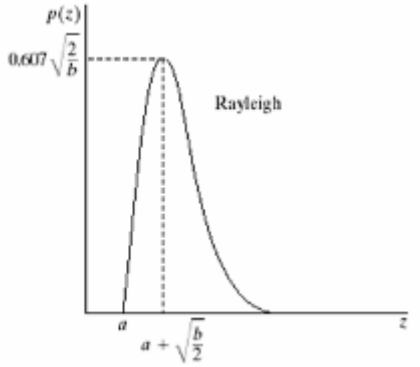
$$p(z)=\frac{1}{\sqrt{2\pi}\sigma}e^{-(z-\mu)^2/2\sigma^2}$$



◆ 瑞利噪声(Rayleigh noise)

$$p(z)=\begin{cases}\frac{2}{b}(z-a)e^{-(z-a)^2/b}, & z\geq a>0\\ 0, & z<a\end{cases}$$

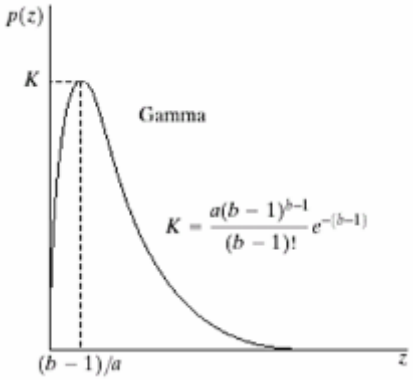
$$\mu=a+\sqrt{\pi b/4},\qquad \sigma^2=\frac{b(4-\pi)}{4}$$



◆ 爱尔兰（伽马）噪声Erlang (Gamma) noise

$$p(z)=\begin{cases}\frac{a^b z^{b-1}}{(b-1)!}e^{-az}, & z\geq 0\\ 0, & z<0\end{cases}$$

$$\mu=\frac{b}{a},\qquad \sigma^2=\frac{b}{a^2}$$

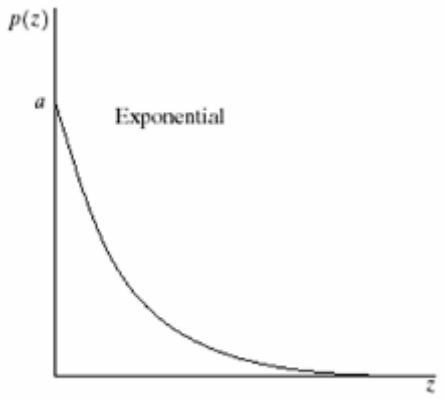


一些重要噪声的概率密度函数

- **指数噪声（ Exponential noise)**

$$p(z)=\begin{cases} ae^{-az}, z\geq 0 \\ 0 \quad\quad , z<0 \end{cases} \quad \text{其中，} a>0$$

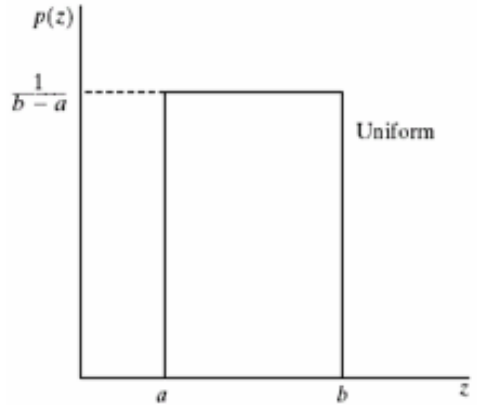
$$\mu=\frac{1}{a}, \quad \sigma^2=\frac{1}{a^2}$$



- **均匀噪声（ *Uniform noise*)**

$$p(z)=\begin{cases} \dfrac{1}{b-a}, & a\leq z\leq b \\ 0 \quad , & \text{otherwise} \end{cases}$$

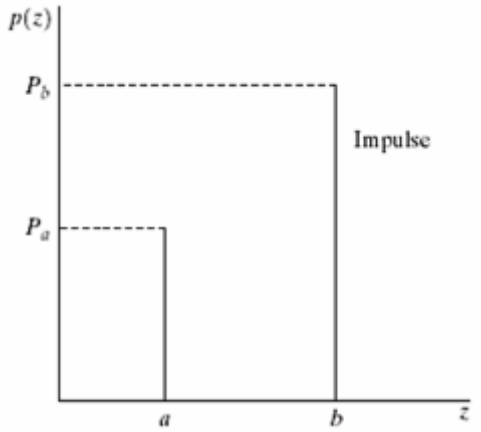
$$\mu=\frac{a+b}{2}, \quad \sigma^2=\frac{(b-a)^2}{12}$$



一些重要噪声的概率密度函数

- **脉冲（椒盐）噪声（impulse(salt-and-pepper) noise）**

$$p(z)=\begin{cases}P_a, & z=a\\ P_b, & z=b\\ 1\text{-}P_a\text{-}P_b, & \text{otherwise}\end{cases}$$



- 如果$b>a$，灰度值b在图像中将显示为一个亮点，相反，a的值将显示为一个暗点。若P_a或P_b为零，则脉冲噪声称为单极脉冲。如果P_a和P_b均不可为零，尤其是它们近似相等时，脉冲噪声值将类似于随机分布在图像上的胡椒和盐粉微粒，称为双极脉冲噪声--也称为椒盐噪声（也称为散粒和尖峰噪声），通常a、b接近饱和值。

7.2 逆滤波图像复原

直接逆滤波

- 所谓直接逆滤波，不考虑噪声因素，用退化图像的傅里叶变换$G(u,v)$除以退化函数$H(u,v)$来计算原始图像的傅里叶变换估计，再取逆傅里叶变换得到复原图像：

$$\hat{F}(u,v)=\frac{G(u,v)}{H(u,v)} \qquad \hat{F}(u,v)=\frac{G(u,v)}{H(u,v)}-\frac{N(u,v)}{H(u,v)}$$

- 但是，实际存在噪声项，通常，图像噪声的傅里叶变换$N(u,v)$很难准确估计，即使已知退化函数$H(u,v)$，也不能简单按上式准确复原图像。更糟糕的是，当退化函数$H(u,v)=0$或者值非常小时，由$N(u,v)/H(u,v)$确定的噪声项将被极度放大，导致图像复原失败。

加窗逆滤波

- **许多情况下，$H(u,v)$会从零频点$H(0,0)$开始快速递减，而噪声$N(u,v)$几乎总是常数。为避免引起噪声的扩大，一般不直接将因子$1/H(u,v)$作为滤波器，而是先将其加窗处理，在$H(u,v)$变得太小或者达到第一个零值前，就将其在某一个频率D_0处截断：**

$$\hat{F}(u,v)=\begin{cases}\dfrac{G(u,v)}{H(u,v)}, & u^2+v^2\le D_0^2\\ G(u,v), & u^2+v^2>D_0^2\end{cases}$$

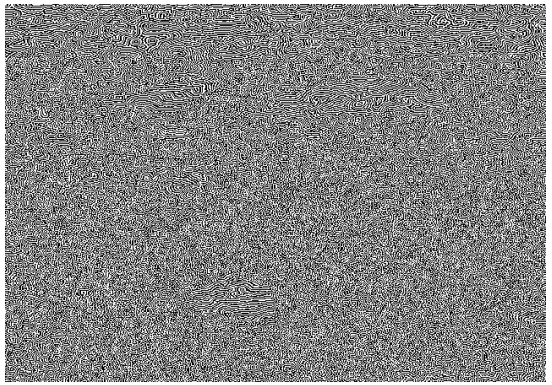
  - 式中，D_0为截止频率，选择D_0使得$H(u,v)$不包括零值点。当然也可以不采用上述矩形窗函数，而用其他窗函数，比如高阶巴特沃斯低通滤波器，使得$1/H(u,v)$在D_0处有个平滑的过渡。

示例：逆滤波图像复原

模拟大气湍流图像退化及其逆滤波复原

(1)原图像

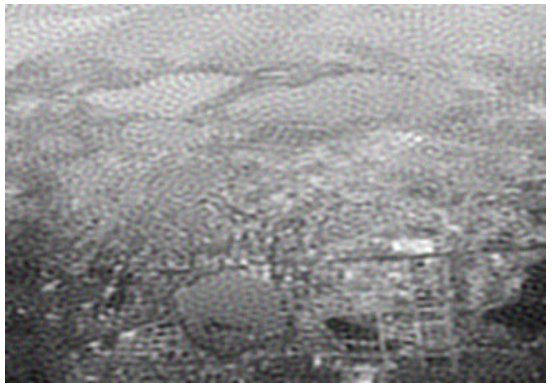
(2)退化图像,k=0.001



(3)直接逆滤波复原

(4)无噪加窗D_0=150

(5)无噪加窗D_0=200



(6)加噪加窗D_0=150

示例：逆滤波图像复原

```
#模拟大气湍流图像退化及逆滤波复原
img = io.imread('./imagedata/aerial_image.png')  #读入一幅航拍图片
#令k=0.001模拟大气湍流模糊退化
img_deg, Hatm = AtmoTurbulenceSim(img,0.001)
#向退化图像中添加向图像中添加均值为0、方差为0.001的高斯噪声
img_deg_noi = util.random_noise(img_deg,mode='gaussian',var=0.001)
img_deg_noi = util.img_as_ubyte(img_deg_noi)
#对不加噪退化图像进行直接逆滤波,令窗口半径极大，相当于不加窗直接逆滤波
img_res1 = WinInvFilter(img_deg, Hatm, np.finfo(np.float32).max)
#对不加噪退化图像进行加窗逆滤波Windowed inverse filtering
#截止频率分别radius为120,180,220
img_res2 = WinInvFilter(img_deg, Hatm, 120)
img_res3 = WinInvFilter(img_deg, Hatm, 180)
img_res4 = WinInvFilter(img_deg, Hatm, 220)
#对加噪退化图像进行加窗逆滤波Windowed inverse filtering
img_res5 = WinInvFilter(img_deg_noi,Hatm, 120)
img_res6 = WinInvFilter(img_deg_noi, Hatm, 180)
```

7.3 维纳滤波图像复原

维纳滤波--最小均方误差滤波

- 逆滤波比较简单，但并没有清楚地说明怎样处理噪声，维纳滤波综合了退化函数和噪声统计特征两个方面进行图像复原处理。
- 假定图像和噪声均为随机过程，目标是找一个未污染图像f的估计值$\hat{f}(x,y)$，使它们之间的均方误差最小：

$$e^2 = E\left\{\left[f(x,y)-\hat{f}(x,y)\right]^2\right\}$$

- 式中，$E\{\bullet\}$表示随机变量的期望值。假定图像退化过程可用一个含加性噪声的线性移不变系统来描述，即：

$$g(x,y)=h(x,y)*f(x,y)+n(x,y)$$

- 其中，噪声$n(x,y)$是一个与原图像$f(x,y)$无关的平稳噪声序列，且要求$n(x,y)$与$g(x,y)$为零均值。

维纳滤波--最小均方误差滤波

- **基于上述条件，满足均方误差最小的复原图像 $\hat{f}(x,y)$ 的频域表达为：**

$$\hat{F}(u,v)=\left[\frac{H^{*}(u,v)}{|H(u,v)|^{2}+S_{n}(u,v)/S_{f}(u,v)}\right]G(u,v)$$

$S_n(u,v)=|N(u,v)|^2$为噪声$n(x,y)$的功率谱；
$S_f(u,v)=|F(u,v)|^2$为退化前图像$f(x,y)$的功率谱。

$$\hat{F}(u,v)=\left[\frac{H^{*}(u,v)}{|H(u,v)|^{2}+NSR}\right]G(u,v)$$

在$S_n(u,v)$、$S_f(u,v)$未知或无法估计时

式中，NSR为图像信噪比（noise-to-signal power ratio），一个待定常数，可通过交互方式试探找到最好视觉效果的NSR值。在一定意义上，图像噪声越严重，NSR值就要取大些。

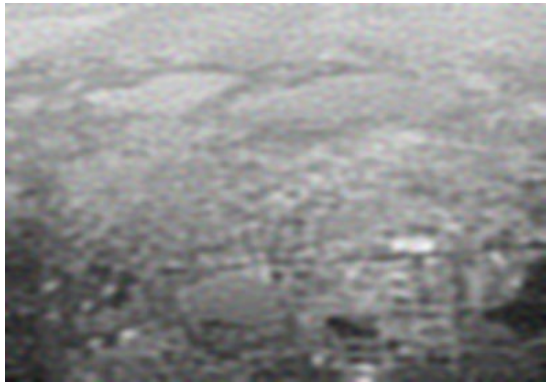
示例：维纳滤波复原图像

模拟大气湍流图像退化及其维纳滤波复原

原图像

(1)退化图像, k=0.0025

(2)无噪声, NSR=0.0001



(3)含噪, NSR=0.005

示例：维纳滤波复原图像

```
#采用维纳滤波复原图像Deblur image using Wiener filter
img = io.imread('./imagedata/aerial_image.png')  #读入一幅航拍图片
rows,cols = img.shape[0:2]  #获取退化图像的高/宽
#令k=0.0025模拟大气湍流模糊退化
img_deg, Hatm = AtmoTurbulenceSim(img,0.0025)
#向退化图像中添加向图像中添加均值为0、方差为0.001的高斯噪声
img_deg_noi = util.random_noise(img_deg,mode='gaussian',var=0.001)
img_deg_noi = util.img_as_ubyte(img_deg_noi)
#采用'reflect'方式扩展图像(下面扩展rows行,右面扩张cols列)
if img.ndim==3:  #彩色图像
    imgex = np.pad(img_deg_noi,((0,rows),(0,cols),(0,0)),mode='reflect')
elif img.ndim==2:  #灰度图像
    imgex = np.pad(img_deg_noi,((0,rows),(0,cols)),mode='reflect')
```

示例：维纳滤波复原图像

```
#计算扩展图像的DFT并中心化
img_dft = fftshift(fft2(imgex,axes=(0,1)),axes=(0,1))
#计算维纳滤波复原图像的频谱
NSR = 0.005
Gp = img_dft * np.conj(Hatm)/(np.abs(Hatm)**2 + NSR + np.finfo(np.float32).eps)
#去中心化
Gp = ifftshift(Gp,axes=(0,1))
 #DFT反变换并取实部
imgp = np.real(ifft2(Gp,axes=(0,1)))
#把输出图像的数据格式转换为uint8
imgp = np.uint8(np.clip(imgp,0,255))
#截取左上角与原图像大小相等的区域作为输出
img_res = imgp[0:rows,0:cols]
```

7.4 约束最小二乘滤波图像复原

约束最小二乘滤波图像复原

- 约束最小二乘滤波对退化前图像$f(x,y)$的复原估计，是最小化以下准则函数J的结果:

$$J \equiv \left\| p(x,y) * \hat{f}(x,y) \right\|^2$$

约束条件为：$\left\| g(x,y) - h(x,y) * \hat{f}(x,y) \right\|^2 \le \varepsilon^2$，其中$\varepsilon^2 \ge 0$

$p(x,y)$为通常选择拉普拉斯算子：$p(x,y) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

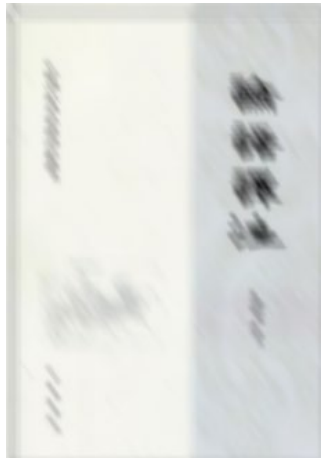
采用Lagrange乘子法，得到上述约束最小二乘优化问题的频域解：

$$\hat{F}(u,v) = \left[\frac{H^*(u,v)}{|H(u,v)|^2 + \gamma |P(u,v)|^2} \right] G(u,v)$$

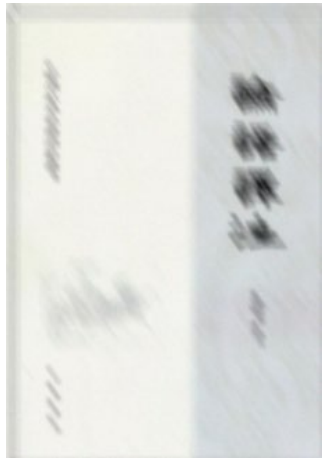
式中，γ是Lagrange乘子，一个待定参数，其选择应满足约束条件，可根据复原效果交互试探选择，也可通过迭代计算。$P(u,v)$是拉普拉斯算子$p(x,y)$的傅里叶变换。

示例：约束最小二乘滤波图像复原函数deconvreg

(1)原图像



(2)运动模糊图像

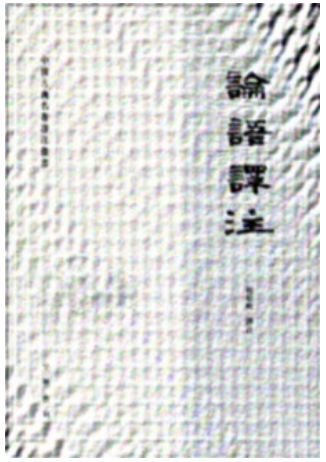


(3)加噪运动模糊图像

高斯噪声
0均值、方差为0. 001

(4)不加噪, 约束最小二乘
$\gamma = 0.005$

(5)不加噪, 维纳
$NSR = 0.005$



(6)加噪, 约束最小二乘
$\gamma = 0.01$

(7)加噪, 维纳
$NSR = 0.01$

7.5 图像修复

图像修复

- **图像修复（Image Inpainting）是对图像中各类瑕疵失真的恢复，包括块状遮挡、文本遮挡、噪声、目标遮挡、划痕等。图像修复算法大致可分成三类：**
  - 基于序列的方法
  - 基于卷积神经网络CNN（Convolutional Neural Network）的方法
  - 基于生成对抗网络GAN（Generative Adversarial Networks）的方法。

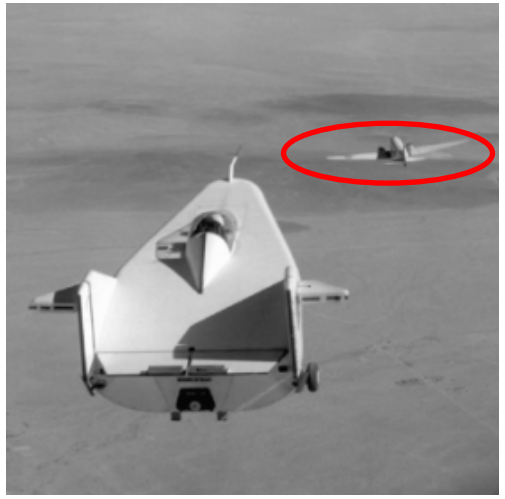
示例：调用函数inpaintExemplar修复图像划痕

```
#OpenCV: 图像修复
img = cv.imread('.\imagedata\lake_crack.png', cv.IMREAD_GRAYSCALE)   #读入瑕疵图像
mask = cv.imread('.\imagedata\lake_crack_mask.png',cv.IMREAD_GRAYSCALE)  #读入划痕区域掩膜图像
img_ns = cv.inpaint(img,mask,5,cv.INPAINT_NS)      #采用INPAINT_NS方式
img_telea = cv.inpaint(img,mask,5,cv.INPAINT_TELEA)  #采用INPAINT_TELEA
image_sk = restoration.inpaint_biharmonic(img, mask)   #采用Scikit-image函数
image_sk = util.img_as_ubyte(image_sk)   #把输出图像的数据格式转换为uint8
```

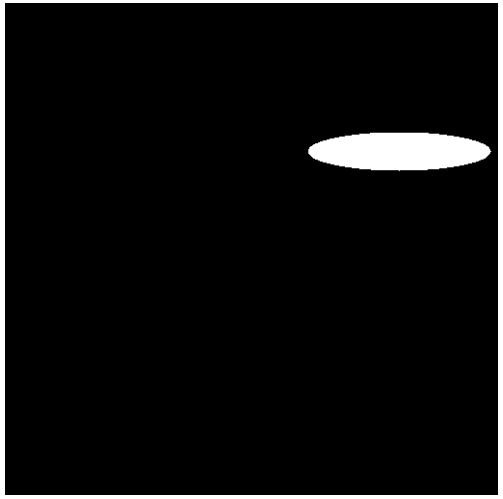
(1)原图像　(2)划痕区域掩膜　(3)采用NS方法　(4)采用TELEA　(5)Scikit-image函数

示例：采用图像修复方法移除图像中的目标区域

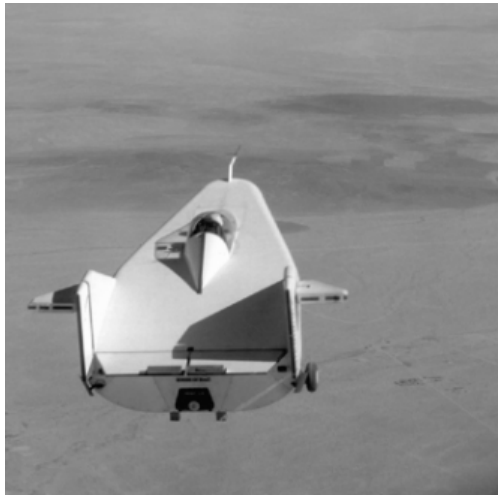
```
#OpenCV: 调用图像修复函数去除图像中指定区域物体
img = cv.imread('.\imagedata\liftingbody.png', cv.IMREAD_GRAYSCALE)  #读入图像
mask = cv.imread('.\imagedata\liftingbody_mask.png',cv.IMREAD_GRAYSCALE)  #读入要去除区域的掩膜
img_rem = cv.inpaint(img,mask,5,cv.INPAINT_NS)   #采用INPAINT_NS方式
#显示结果（略）
```



(1)原图像及要去除区域



(2)要去除区域掩膜



(3)处理结果

Q&A