《数字图像处理》习题课

- 第一次作业
- 第二次作业
- 第三次作业
- 知识点补充

《数字图像处理》第一次作业

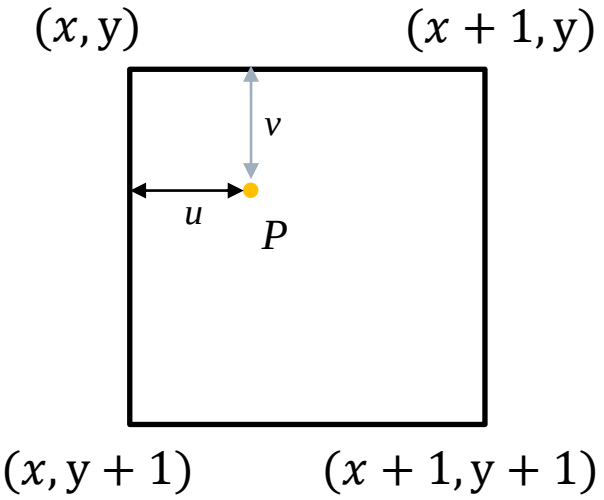
- 编程作业任务
  - 实现三种图像插值算法
    - ✓ 最近邻内插
    - ✓ 双线性内插
    - ✓ 双三次内插
  - 对给定的样张均完成0.5倍和3倍缩放

- 错误示例：
  - 双线性插值失真

3倍插值放大

3倍插值放大

```
%双线性插值算法
function img_new = myBilinear(img, ratio)
% 读取原图尺寸
[row,col,channel] = size(img);

% 根据缩放比例ratio建立空图像，待赋值
row_new = round(row*ratio);
col_new = round(col*ratio);
img_new = zeros(row_new, col_new, channel, class(img));

%padding边界
img_t = zeros(row+2, col+2, channel);
img_t(2:row+1,2:col+1,:) = img;
img_t(1,2:col+1,:) = img(1,:,:);
img_t(row+2,2:col+1,:) = img(row,:,:);
img_t(2:row+1,1,:) = img(:,1,:);
img_t(2:row+1,col+2,:) = img(:,col,:);
img_t(1,1,:) = img(1,1,:);
img_t(1,col+2,:) = img(1,col,:);
img_t(row+2,1,:) = img(row,1,:);
img_t(row+2,col+2,:) = img(row,col,:);

%遍历每个pixel，进行赋值
for i = 1:row_new
    for j = 1:col_new
            u = i/ratio - floor(i/ratio); %行权重
            v = j/ratio - floor(j/ratio); %列权重
            x = floor(i/ratio)+1;
            y = floor(j/ratio)+1;

            img_new(i,j,:) = (1-u)*(1-v)*img_t(x,y,:) + u*(1-v)*img_t(x,y+1,:)...
                +(1-u)*v*img_t(x+1,y,:) + u*v*img_t(x+1,y+1,:);
    end
end
```

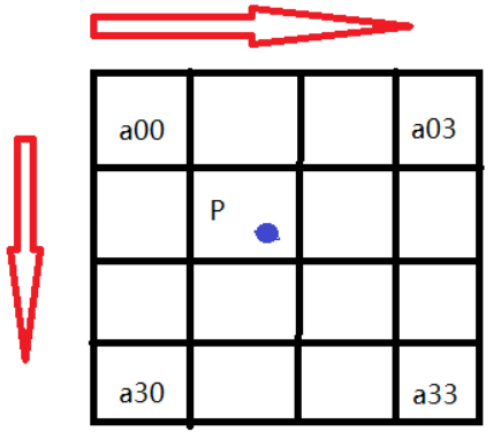


注意四个pixel的权重！

- 错误示例：
  - 双三次插值结果出现网格状条纹

  - 问题代码示例：

```
%% 双三次插值
function [imout] = myBicubic(im, ratio)
% 读取原图尺寸
[x_ori, y_ori, ~] = size(im);

%根据缩放的比例ratio建立一个空图像，待赋值
x = x_ori * ratio;
y = y_ori * ratio;
imout_raw = zeros(x, y);

%遍历每一个pixel
for i = 1:x
    for j = 1:y
        x_temp = x_ori/x*i;
        y_temp = y_ori/y*j;
        %中心放缩
        x_dn = floor(x_temp);
        x_up = ceil(x_temp);
        y_dn = floor(y_temp);
        y_up = ceil(y_temp);
        % 遍历周围16个元素，计算加权和
        for k = (x_dn - 1):(x_up + 1)
            for l = (y_dn - 1):(y_up + 1)
                if k < 1 || k > x_ori || l < 1 || l > y_ori
                    pixel = 0;
                else
                    pixel = im(k, l, 1);
                end
                imout_raw(i, j) = imout_raw(i, j) + pixel * BiCubic(k-x_temp) * BiCubic(l-y_temp);
            end
        end
    end
end

imout = uint8(imout_raw);
end
```



$$W(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & \text{for } |x| \le 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & \text{for } 1 < |x| < 2 \\ 0 & \text{otherwise} \end{cases}$$

其中，a取-0.5.

原因

Matlab

```
>> uint8(5) * 0.3

ans =

  uint8

    2
```

```
>> uint8(5) + 0.3

ans =

  uint8

    5
```

Python

```
import numpy as np
```

```
number = np.array([5]).astype(np.uint8)
number
```

```
array([5], dtype=uint8)
```

```
number * 0.3
```

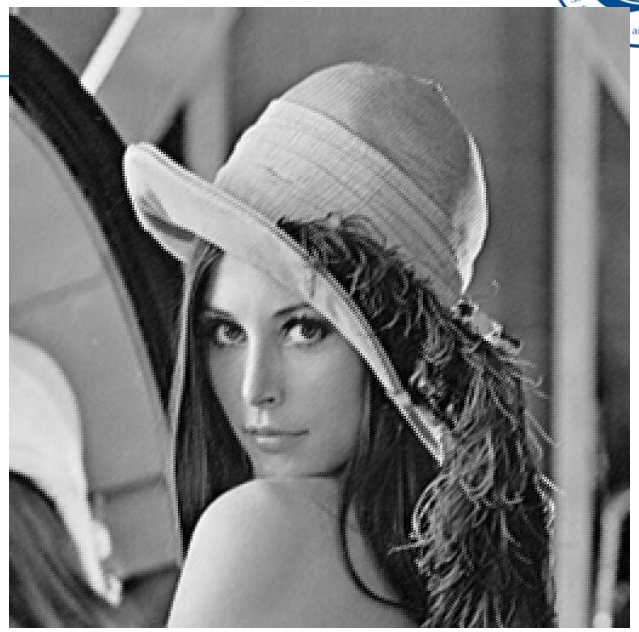
```
array([1.5])
```

```
number + 0.3
```

```
array([5.3])
```

- 错误示例：
  - 双三次插值BiCubic函数实现错误

```
%% BiCubic权值
function w=s (wx)
    wx = abs(wx);
    if wx <1
        w = 1 - 2*wx^2 + wx^3;
    elseif wx <=1 && wx < 2
        w = 4 - 8*wx +5*wx^2 - wx^3;
    else
        w = 0;
end
```



✓ && 与

《数字图像处理》第二次作业

- 编程作业任务
  - 图像直方图均衡化
  - 图像空域滤波
    - ✓ 均值滤波（滤波次数$n \to \infty$）
    - ✓ 中值滤波（滤波次数$n \to \infty$）
    - ✓ 图像锐化（拉普拉斯算子）

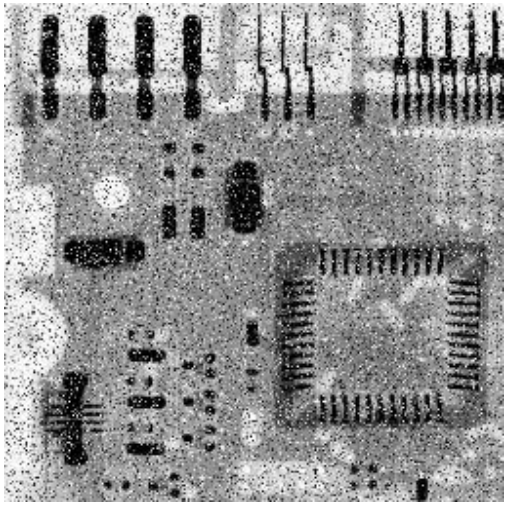
图像直方图均衡化

- 根据教材3.3.1节，实现直方图均衡化算法，对给定的样张进行均衡化，灰度级数分别为2、64、256
- 图例：bridge

```
% 直方图均衡化函数，L为灰度级数，处理单通道的灰度图像
function [img_2] = myHisteq(img_1, L)
% 读取图像尺寸
size_1 = size(img_1);
h = size_1(1);
w = size_1(2);

% 将图像转换为L级灰度图
if L>2   % 在灰度级为2时，统计特性不明显，没有采用式(1.4)的灰度级划分方法，而是采用中位数作为阈值，划分0和1
    img_1 = floor((double(img_1)*L)/256);   % 变成L个灰度级
else
    th=median(img_1(:));
    for i=1:h
        for j=1:w
            if img_1(i,j)>th
                img_1(i,j)=1;
            else
                img_1(i,j)=0;
            end
        end
    end
end

% 统计各灰度级的出现次数
n = zeros(1,L);
for i=1:h
    for j=1:w
        k = img_1(i,j);
        n(k+1)=n(k+1)+1; % 索引为灰度级加一
    end
end
% 灰度映射
img_2 = zeros(h, w);
for i=1:h
    for j=1:w
        k = img_1(i,j)+1;
        img_2(i,j) = sum(n(1:k))*(L-1)/(h*w);
    end
end

% 映射回0~255
img_2 = img_2*(256/L);
img_2 = uint8(img_2);
end
```

图像平滑

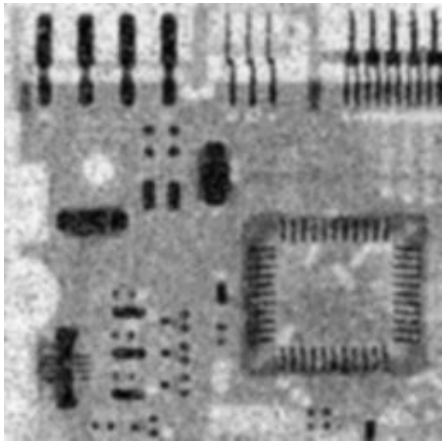
- 根据教材3.5节，实现均值滤波和中值滤波算法，对给定的样张进行滤波，滤波器模板均为3×3
  - 比较不同滤波次数对实验结果的影响（滤波次数$n \to \infty$）
- 图例：circuit



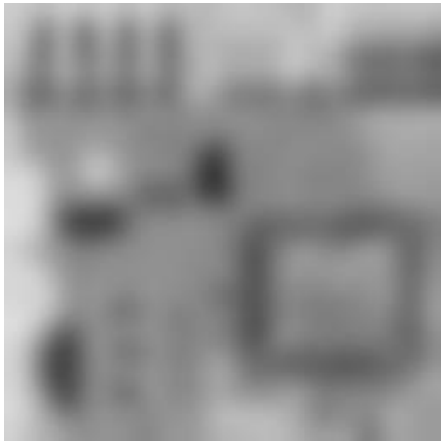
问题：

- 无限次均值滤波影响因素
  - ✓ 边界处理方式
    - Padding 0
    - Padding 图像边界
    - 不进行填充，保留图像最外层边界
  - ✓ 每次滤波完是否量化
    - 保存成uint8进行下一次滤波
    - 直接按float进行下一次滤波

- padding 边界 + 每次滤波结束不量化（浮点）

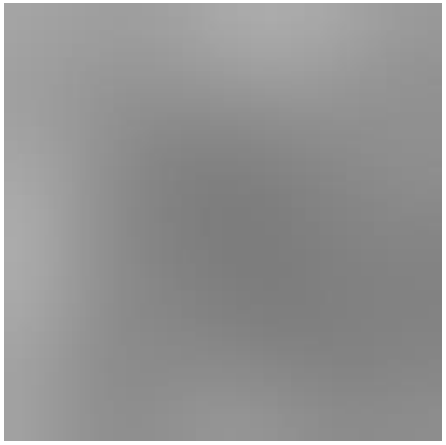


1



100

500

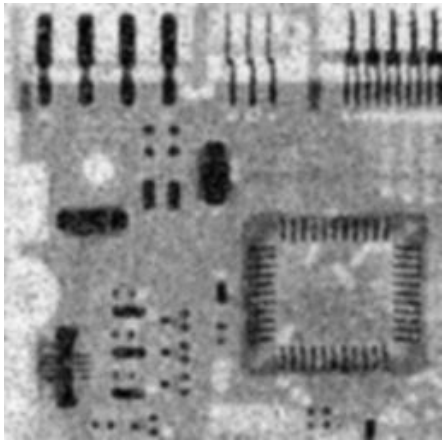


2000

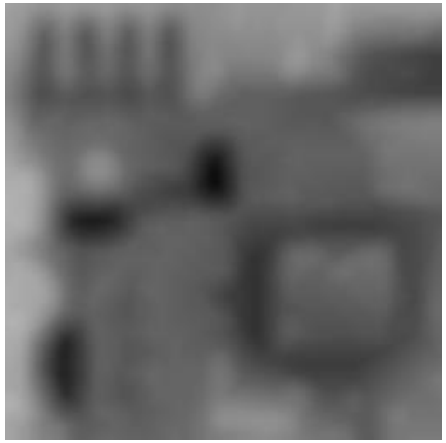
10000

20000

- padding 边界 + 每次滤波结束量化（uint8）

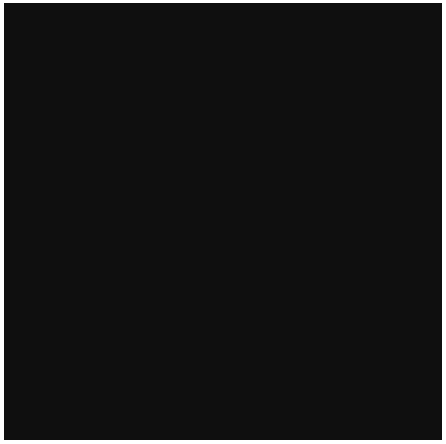


1

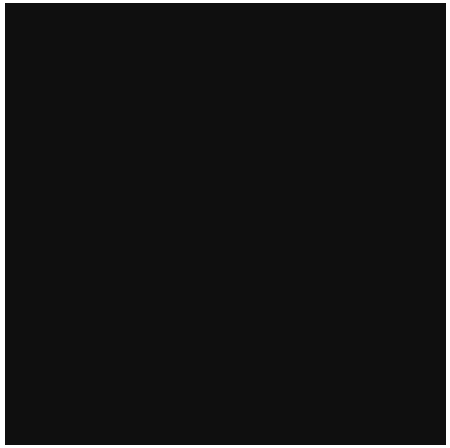


100

500



10000

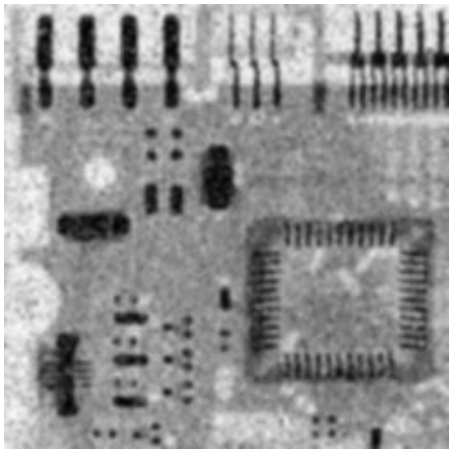


20000

```
np.array([5.1]).astype(np.uint8)
array([5], dtype=uint8)

np.array([5.9]).astype(np.uint8)
array([5], dtype=uint8)
```

Numpy中对于int8类型转换方式导致我们的灰度越来越低

- padding 边界 + 每次滤波结束量化 （uint8）



1

100

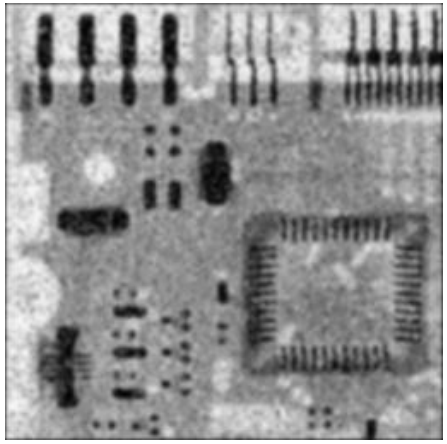
500

10000

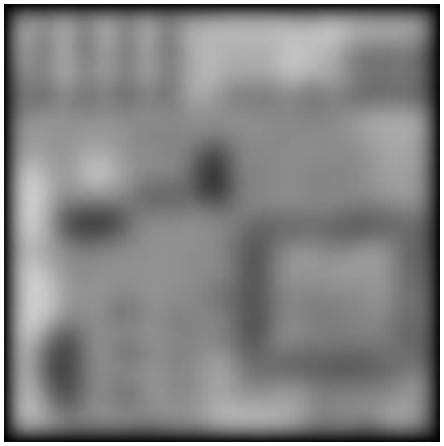
20000

```
np.array([5.1]).round().astype(np.uint8)
array([5], dtype=uint8)

np.array([5.9]).round().astype(np.uint8)
array([6], dtype=uint8)
```

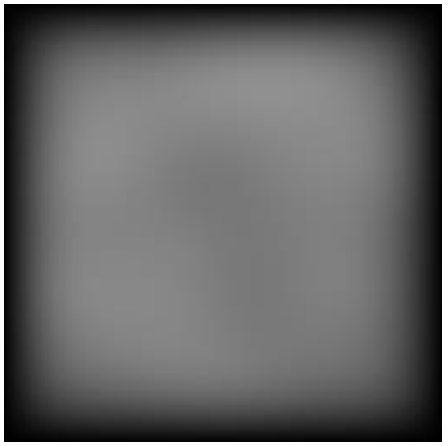
- Padding 0 + 每次滤波结束量化 (uint8)



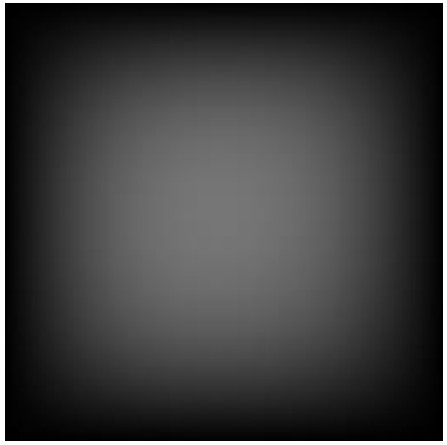
1



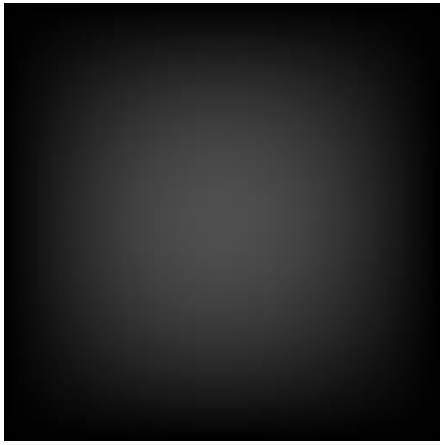
100



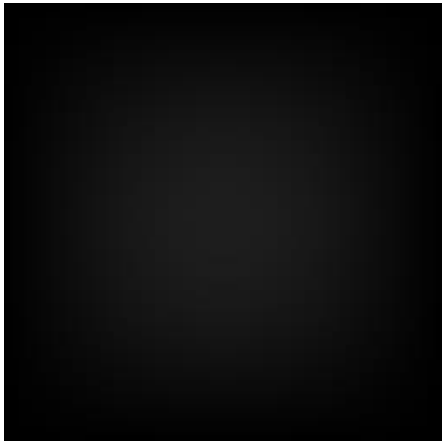
1000



5000

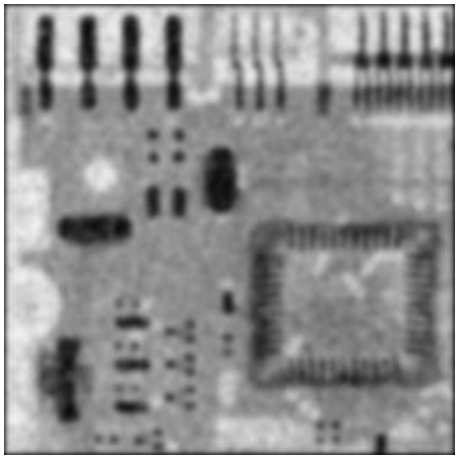


10000



20000

- 保留最外层边界 + 每次滤波结束量化 (uint8)



5

20

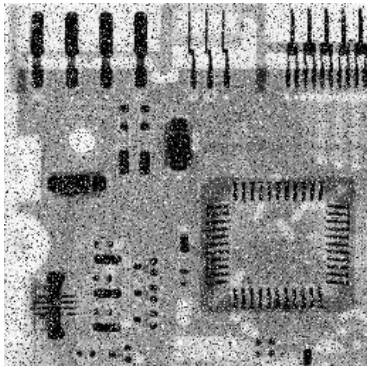
100

5000

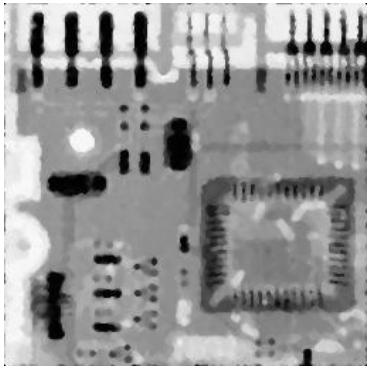
10000

20000

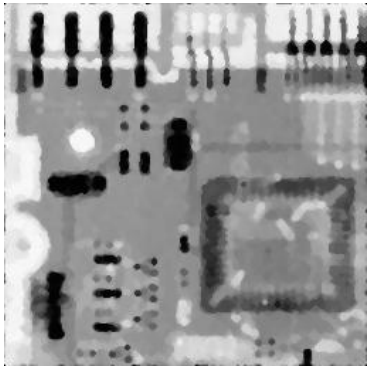
- 问题：
  - 无限次中值滤波



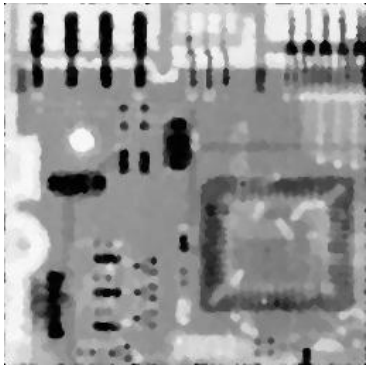
输入



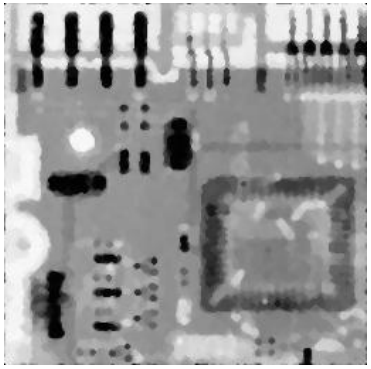
5



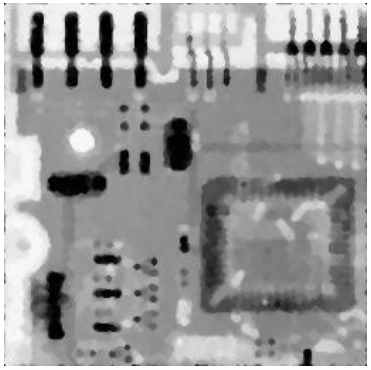
500



5000



10000



20000

图像锐化

- 根据教材3.6节，实现图像锐化算法，使用拉普拉斯算子（如下）
- 图例：moon

-1	-1	-1
-1	8	-1
-1	-1	-1

- 问题示例：
  - 灰度越界
    - ✓ Matlab

```
>> uint8([-1,256])

ans =

  1×2 uint8 行向量

    0   255
```

    - ✓ Python

```
np.array([-1, 256]).astype(np.uint8)
array([255,   0], dtype=uint8)
```

      - ➢ 使用python编程的同学请注意越界问题

《数字图像处理》第三次作业

- 根据教材5.8节例5.12，对经过大气湍流退化的图片实现全逆滤波，半径受限逆滤波以及维纳滤波，并对比。
- 编程：Matlab, Python, C++ 均可
  - Matlab代码框架已提供，完成滤波函数即可
    - ✓ atmosph.m (P221，例 5.11)
    - ✓ my_wiener.m, my_inverse.m
      - 截止的实现详见例5.11关于图5.27解释；巴特沃斯低通滤波器见第四章4.8节
- 图例：demo-1.jpg

- 运动模糊退化函数
  - 运动模糊退化模型

$$H(u,v)=\frac{T}{\pi(ua+vb)}\sin[\pi(ua+vb)]e^{-j\pi(ua+vb)} \quad x(t)=at/T;\ \ y(t)=bt/T$$

  - 错误示例及修正

```
H = zeros(M,N);
a=0.02;
b=0.02;
T=1;
for u = (-M/2):1:(M/2)-1
    for v = (-N/2):1:(N/2)-1
        D = pi*(u*a+v*b);
        H(u+(M/2)+1,v+(N/2)+1) = (T/D)*sin(D)*exp(-1i*D);
        end
    end
end
```

```
H = zeros(M,N);
a=0.02;
b=0.02;
T=1;
for u = (-M/2):1:(M/2)-1
    for v = (-N/2):1:(N/2)-1
        D = pi*(u*a+v*b);
        if D==0
            H(u+(M/2)+1,v+(N/2)+1)=T;
        else
            H(u+(M/2)+1,v+(N/2)+1) = (T/D)*sin(D)*exp(-1i*D);
        end
    end
end
```

- 噪声添加：均值为0，方差为sigma的高斯加性噪声
  - 空域添加噪声
    - ✓ Matlab: imnoise(I, ‘gaussian’, m, var)函数
      - 当I为uint8类型图像，var = (sigma)/(255×255）
      - 当I为归一化到[0,1]的double类型图像，var = sigma
  - 频域添加噪声
    - ✓ $G(u,v) = F(u,v) + N(u,v)$
      - 具体实现：

```
% % noise
noise = sqrt(sigma) .* randn(M,N);
Noise = fftshift(fft2(noise));
```

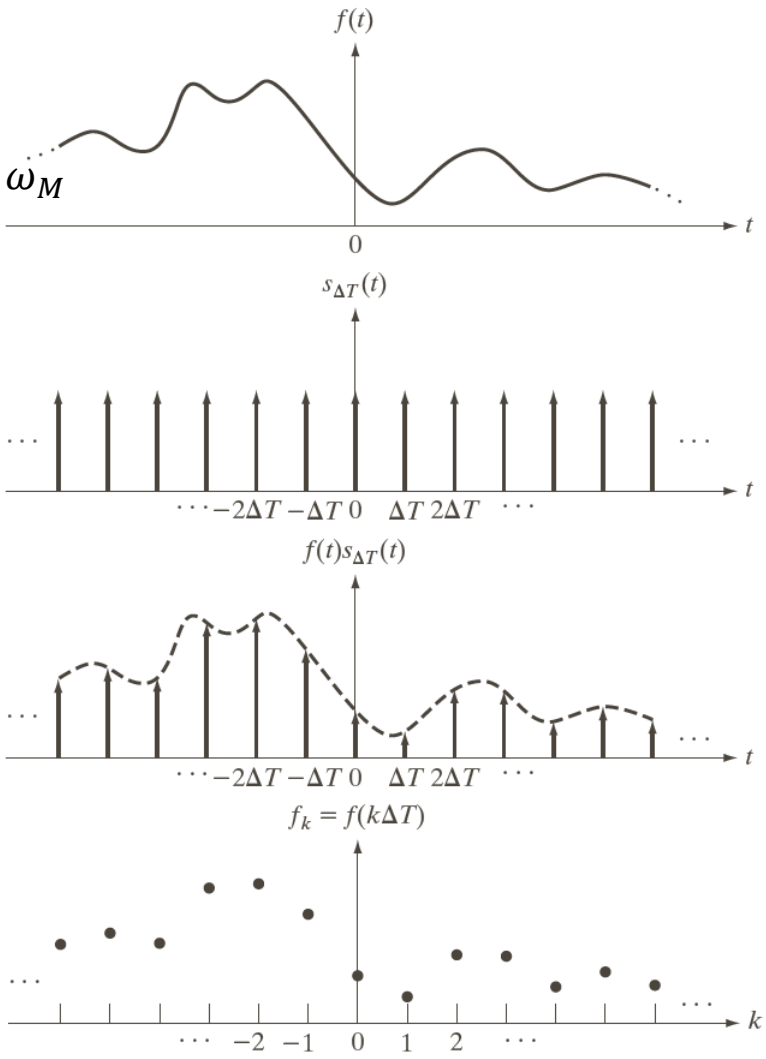
奈奎斯特抽样定理

$f(t) \leftrightarrow F(\mu)$, 且$F(\mu) = 0$, 当$|\mu| > \omega_M$

$$s_{\Delta t} = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta t)$$

$$\updownarrow$$

$$P(\mu) = \frac{2\pi}{\Delta t} \sum_{k=-\infty}^{\infty} \delta(\mu - k\frac{2\pi}{\Delta t})$$



a
b
c
d

FIGURE 4.5
(a) A continuous function. (b) Train of impulses used to model the sampling process. (c) Sampled function formed as the product of (a) and (b). (d) Sample values obtained by integration and using the sifting property of the impulse. (The dashed line in (c) is shown for reference. It is not part of the data.)

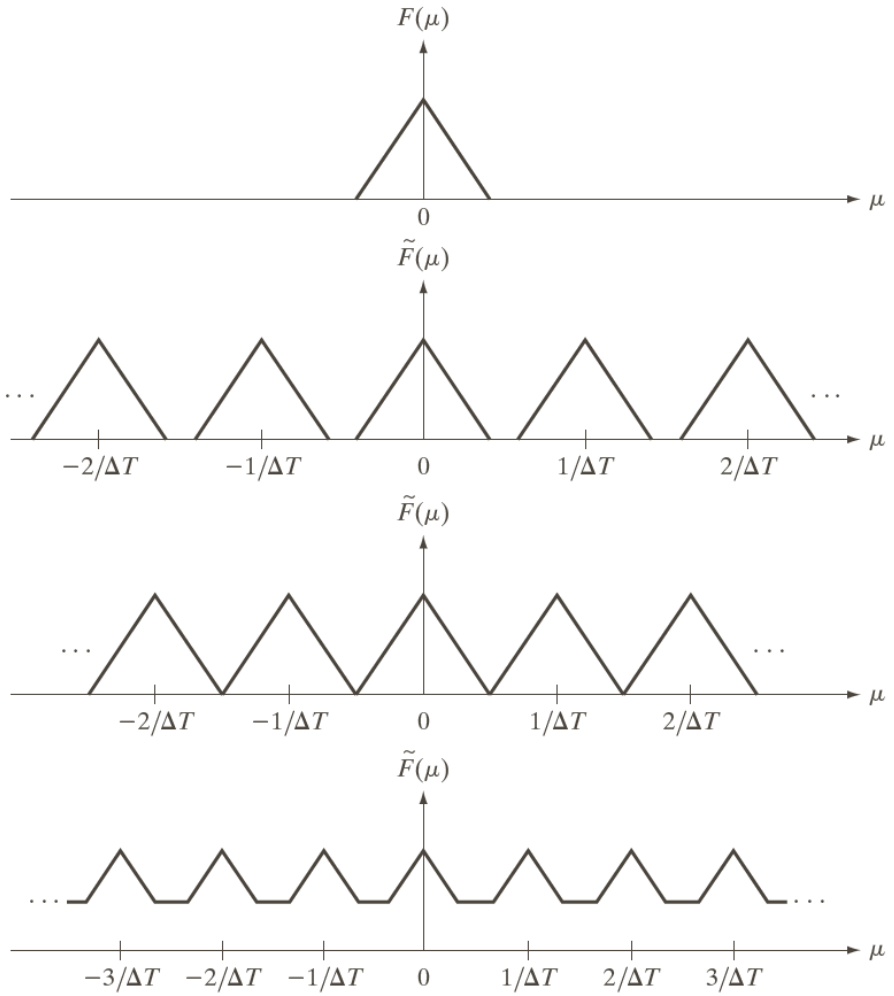
取样函数的傅里叶变换

奈奎斯特采样率：
完全等于信号最高频率的两倍的取样率。

$$f(t) \times S_{\Delta T}$$

$$\updownarrow$$

$$\frac{1}{2\pi} \times F(\mu) * P(\mu)$$



过抽样

欠抽样

频谱搬移

- 傅里叶变换的频移性质
  - CFT

$$f(t) \leftrightarrow F(\omega)$$

$$e^{j\omega_0 t} f(t) \leftrightarrow F(\omega - \omega_0)$$

  - DTFT

$$f(n) \leftrightarrow F(\Omega)$$

$$e^{j\Omega_0 t} f(n) \leftrightarrow F(\Omega - \Omega_0)$$

  - 二维DFT

$$f(x,y) \leftrightarrow F(u,v)$$

$$e^{j2\pi(\frac{u_0 x}{M}+\frac{v_0 y}{N})} f(x,y) \leftrightarrow F(u-u_0, v-v_0)$$

当$u_0 = \frac{M}{2}, v_0 = \frac{N}{2}$:

$$e^{j2\pi(\frac{u_0 x}{M}+\frac{v_0 y}{N})} = e^{j\pi(x+y)} = e^{j\pi x} \times e^{j\pi y}$$

$$= [\cos(\pi x) + j\sin(\pi x)] \times [\cos(\pi y) + j\sin(\pi y)]$$

$$= (-1)^x \times (-1)^y = (-1)^{x+y}$$

$$\therefore \quad (-1)^{x+y} f(x,y) \leftrightarrow F\left(u - \frac{M}{2}, v - \frac{N}{2}\right)$$

- 谢谢！