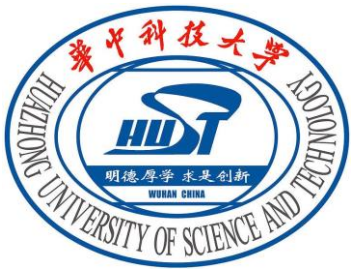


华中科技大学
生物医学数字信号处理实验报告

图像的卷积与逆卷积

学院　　工程科学学院

班级　　工程科学学院（生医）1701 班

姓名　　汪能志

学号　　U201713082

2020 年 3 月 15 日

目录

1. 实验内容

生成待卷积的图像并对其进行卷积模糊操作，并添加噪声。
使用合适的方法对其进行逆卷积，恢复出原始图像。

2. 实验数据的生成

2.1. 待卷积图像的生成

2.1.1. 椭圆

平面上的椭圆及其内部可以用以下方程表示：

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}\le 1 \tag{1}$$

2.1.2. 三角形

生成的直角三角形的直角边平行于坐标轴，因此可以认为是一个菱形的四分之一。
平面上的菱形和其内部可以用以下方程表示：

$$\frac{|x-x_0|}{a}+\frac{|y-y_0|}{b}\le 1 \tag{2}$$

通过设置 (x,y) 相对于菱形中心 (x_0,y_0) 的象限，可以控制这个菱形的方向。

2.1.3. 十字形

十字形可以认为是横竖两个方向上的矩形所组成的。

2.1.4. 图像的生成

图像可以认为是一个离散的二维矩阵，通过索引判断每个点的坐标，并代入上述公式，就可以判断这个点是否处在指定的图形内，并修改该点对应的值。

最后生成的图像为 1024×1024 的灰度图，数据使用浮点数表示，动态范围为 0-1。

2.1.5. MATLAB 实现

```
function img = Test_Image()
img = zeros(1024, 1024);

for x = 1:1024
    for y = 1:1024
        % 生成第一个椭圆
        x_1 = 160;
        y_1 = 640;
        a_1 = 144;
        b_1 = 320;
```

```
        L1 = ((x - x_1)^2) / a_1^2 + ((y - y_1)^2) / b_1^2;

        if (L1 < 1)
            img(x, y) = 0.25;
        end

        % 生成第二个椭圆
        x_2 = 320;
        y_2 = 640;
        a_2 = 144;
        b_2 = 256;
        L2 = ((x - x_2)^2) / a_2^2 + ((y - y_2)^2) / b_2^2;

        if (L2 < 1)
            img(x, y) = 0.5;
        end

        % 生成三角形
        x_3 = 480;
        y_3 = 420;
        a_3 = 360;
        b_3 = 360;
        L3 = abs(x - x_3) / a_3 + abs(y - y_3) / b_3;

        if (L3 < 1 && x > x_3 && y < y_3)
            img(x, y) = 1;
        end

    end
end

% 生成十字形
x_4 = 720;
y_4 = 720;
W = 20;
L = 420;
img(x_4 - W / 2:x_4 + W / 2, y_4 - L / 2:y_4 + L / 2) = 1;
img(y_4 - L / 2:y_4 + L / 2, x_4 - W / 2:x_4 + W / 2) = 1;
end
```

2.2. 卷积核的产生

2.2.1. 卷积模板的选择

实验中使用高斯模糊的卷积核对图像进行模糊操作。高斯模糊的卷积核可以用下式表示：

$$G(x,y)=\frac{1}{2\pi\sigma^2}\cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3}$$

再对卷积核进行归一化，就可以得到对图像进行模糊操作的卷积核。

2.2.2. MATLAB 实现

```
function h = Conv_Core(m, n, sigma)
m = (m - 1) / 2;
n = (n - 1) / 2;
[h1, h2] = meshgrid(-m:+m, -n:+n);
hg = exp(-(h1.^2 + h2.^2) / (2 * sigma^2));
h = hg ./ sum(hg, 'all');
end
```

3. 二维卷积与噪声添加

3.1. MATLAB 库函数

通过 MATLAB 中的库函数 conv2 可以实现二维卷积的操作。

3.2. 自定义实现

3.2.1. 计算方法

根据二维卷积的定义，可以将二维卷积转换为行和列方向上的两次卷积，计算公式如下：

$$y=\begin{pmatrix} Y_1 \\ \vdots \\ Y_k \\ \vdots \\ Y_{m+p-1} \end{pmatrix}=\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_m \end{pmatrix} * \begin{pmatrix} H_1 \\ H_2 \\ H_3 \\ \vdots \\ H_p \end{pmatrix} \quad Y_k=\sum_{i=-\infty}^{+\infty}(X_i * H_{k-i+1})=\sum_{i=\max(k-n+1,1)}^{\min(m,k)}(X_i * H_{k-i+1}) \tag{4}$$

3.2.2. MATLAB 实现

```
function y = conv2_double_conv(x, h)
[row_x, col_x] = size(x);
[row_h, col_h] = size(h);

row_y = row_x + row_h - 1;
col_y = col_x + col_h - 1;

y = zeros(row_y, col_y);

for y_r = 1:row_y
    for x_r = max(y_r - row_h + 1, 1):min(row_x, y_r)
        h_r = y_r - x_r + 1;
        X = x(x_r, :);
```

```
        H = h(h_r, :);
        y(y_r, :) = y(y_r, :) + conv(X, H, 'full');
    end
end

end
```

3.3. 噪声添加

MATLAB 中的 randn 函数可以生成符合标准正态分布的矩阵，通过设置一定的信噪比可以控制添加的信号的强度。添加噪声后，图像的动态范围会超过 0-1。需要对其动态范围进行调整。这里将小于 0 的部分强制置零，大于 1 的部分强制置 1。

添加噪声的部分和调用 MATLAB 库函数 awgn(X, SNR, 'measured')的效果相同。

```
function Signal_Noise = Add_Noise(Signal, SNR)
% 计算信号功率
Signal_Power = sum(abs(Signal(:)).^2) / numel(Signal);
Signal_dB = 10 * log10(Signal_Power);

% 计算噪声功率
Noise_dB = Signal_dB - SNR;
Noise_Power = 10^(Noise_dB / 10);

% 生成噪声信号
Rand_Noise = randn(size(Signal));
Noise = sqrt(Noise_Power) * Rand_Noise;

Signal_Noise = Signal + Noise;
% Signal_Noise = awgn(Signal, SNR, 'measured');

% 调整动态范围
Signal_Noise = max(Signal_Noise, 0);
Signal_Noise = min(Signal_Noise, 1);
end
```

4. 二维逆卷积

4.1. 最小二乘法和快速梯度下降

4.1.1. 计算方法

卷积过程可以线性表示为：

$$Y = X * h + N = A \cdot X + N \tag{5}$$

如果以误差最小为标准进行逆卷积，则有：

$$X^{*} = \arg\min_{X}(\| Y - A \cdot X \|^{2}) = (A^{T} \cdot A)^{-1} \cdot A^{T} \cdot Y \tag{6}$$

式 6 即为逆卷积问题的最小二乘法解（解析解）。但是这个方法在实际操作时资源开销太大，难以实现。

最小二乘法中的误差表达式可以按照式 7 的方式展开。

$$\min \| Y - A_h \cdot X \|^2 \Leftrightarrow \min[(Y - A \cdot X)^T \cdot (Y - A \cdot X)] \Leftrightarrow \min(X^T A^T A X - 2 X^T A^T Y) \tag{7}$$

计算展开后表达式的梯度如下：

$$F(X) = X^T A^T A X - 2 x^T A^T Y \quad \frac{1}{2} \cdot \nabla F(X)) = (A^T A X - A^T Y) = A^T (A \cdot X - Y) \tag{8}$$

通过式 8，可以得到求最小二乘法解的梯度下降表达式：

$$X^{[k+1]} = X^{[k]} - \delta \cdot [A^T (A \cdot X^{[k]} - Y)] \quad \text{s.t. } F(X^{[k+1]}) < F(X^{[k]}) \tag{9}$$

但是，矩阵乘法的时间复杂度极高，在运算时会消耗非常长的时间。因此需要将其优化为卷积运算以优化计算速度。

$$\begin{gathered} h = [h_1, h_2, \cdots, h_{n-1}, h_n]^T \quad \tilde{h} = [h_n, h_{n-1}, \cdots, h_2, h_1]^T \\ \tilde{h} * Y = (y_1, y_2, \cdots, y_{n-1}, \underbrace{y_n, \cdots, y_{m+n-1}}_{A^T \cdot Y}, y_{m+n}, \cdots, y_{m+2n-2}) \end{gathered} \tag{10}$$

通过式 10，可以定义提取操作：

$$M(\tilde{h} * Y) = A^T \cdot Y \tag{11}$$

则有快速梯度下降表达式如下：

$$X^{[k+1]} = X^{[k]} - \delta \cdot M[\tilde{h} * (X^{[k]} * h - Y)] \tag{12}$$

该方法可以类似地延拓到任意维度逆卷积上。

在实际计算时，可以在计算过程中添加非负约束，以提高计算准确性。

4.1.2. MATLAB 实现

```
function x = deconv2_fast_grad_descent(y, h)
[row_y, col_y] = size(y);
[row_h, col_h] = size(h);
row_x = row_y - row_h + 1;
col_x = col_y - col_h + 1;

h_tilde = rot90(h, 2);

% half_row = (row_h - 1) / 2;
% half_col = (col_h - 1) / 2;
x = zeros(row_x, col_x);

% x = y(half_row + 1:end - half_row, half_col + 1:end - half_col);

lr = 0.02;
epoch = 1000;

```

```
for i = 1:epoch
    grad = conv2(h_tilde, conv2(x, h, 'full') - y, 'full');
    grad = grad(row_h:row_y, col_h:col_y);
    descent = lr .* grad;
    x = x - descent;
    x = max(x, 0);
end

end
```

4.2. 光滑性逆卷积

4.2.1. 计算方法

如果有X光滑的先验信息，则可以得到如下的优化问题：

$$\begin{aligned} X^* &= \min \| X \|_2^2 \\ \text{s.t.} \quad & \| Y - A \cdot X \|_2^2 \le \epsilon^2 \end{aligned} \tag{13}$$

为了方便求解，可以将其转换为如下的优化问题：

$$X^* = \arg\min_X (\| Y - A \cdot X \|_2^2 + \lambda \| X \|_2^2) \tag{14}$$

λ越大，则光滑性的权重越大；λ越小，则准确性的权重越大。

式 14 的解析解为：

$$X^* = \arg\min_X (\| Y - A \cdot X \|_2^2 + \lambda \| x \|_2^2) = (A^T \cdot A + \lambda I)^{-1} \cdot A^T \cdot Y \tag{15}$$

同理，也可以使用梯度下降法进行求解：

$$\begin{aligned} F(X) &= \| Y - A \cdot X \|_2^2 + \lambda \| x \|_2^2 \quad \frac{1}{2} \nabla F(X) = A^T (A \cdot X - Y) + \lambda \cdot X \\ X^{[k+1]} &= X^{[k]} - \delta \cdot (M[\tilde{h} * (X^{[k]} * h - Y)] + \lambda \cdot X^{[k]}) \end{aligned} \tag{16}$$

4.2.2. MATLAB 实现

```
function x = deconv2_smooth(y, h, lambda)
[row_y, col_y] = size(y);
[row_h, col_h] = size(h);
row_x = row_y - row_h + 1;
col_x = col_y - col_h + 1;

h_tilde = rot90(h, 2);
% half_row = (row_h - 1) / 2;
% half_col = (col_h - 1) / 2;

% 设置初值
x = zeros(row_x, col_x);
% x = y(half_row + 1:end - half_row, half_col + 1:end - half_col);

```

```
% 设置迭代参数
lr = 0.02;
epoch = 1000;
for i = 1:epoch
    grad = conv2(h_tilde, (conv2(x, h, 'full') - y), 'full');
    grad = grad(row_h:row_y, col_h:col_y);
     Regular = lambda * x;
%     grad      梯度项
%     Regular   光滑项
     descent = lr .* (grad + Regular);
     x = x - descent;
     x = max(x, 0);

end

end
```

4.3. 稀疏性逆卷积

4.3.1. 计算方法

如果有X稀疏的先验信息，则可以得到如下的优化问题：

$$\begin{aligned} X^* &= \min \| X \|_1 \\ \text{s.t.} \quad & \| Y - A \cdot X \|_2^2 \le \epsilon^2 \end{aligned} \tag{17}$$

为了方便求解，可以将其转换为如下的优化问题：

$$X^* = \arg\min_X (\| Y - A \cdot X \|_2^2 + \lambda \| X \|_1) \tag{18}$$

λ越大，则稀疏性的权重越大；λ越小，则准确性的权重越大。

记式 18 的最优解为：

$$X^* = [x_1, x_2, \cdots]^T \tag{19}$$

对式 18 可以做如下修改：

$$\begin{cases} \Lambda = \lambda \cdot [\dfrac{\alpha}{|x_1| + \iota}, \dfrac{\alpha}{|x_2| + \iota}, \cdots]^T = [\lambda_1, \lambda_2, \cdots]^T \\ X^* = \arg\min_X (\| Y - A \cdot X \|_2^2 + \| \Lambda^T \cdot X \|_1) \end{cases} \tag{20}$$

其中，α和ι为常数。ι的值非常小，其目的是避免出现分母为 0 的情况。

由此可知，X^*中的元素越小，则其在计算稀疏性时的权重越大。因此式 20 的解比式 18 的解更具有稀疏性。

下面讨论如何求解式 20。

$$H(X) = \| Y - AX \|_2^2 \tag{21}$$

在$X^{[k]}$附近利用泰勒展开将$H(X)$展开为二阶近似多项式：

$$H(X) \approx H(X^{[k]}) + < \nabla H(X^{[k]}), X - X^{[k]} > + \frac{1}{2\sigma^{[k]}} \| X - X^{[k]} \|_2^2 \tag{22}$$

因此可得：

$$\begin{aligned}&\| \Lambda^T \cdot X \|_1 + H(X) \\ \approx &\| \Lambda^T \cdot X \|_1 + < \nabla H(X^{[k]}, X - X^{[k]} > + \frac{1}{2\delta^{[k]}} \| X - X^{[k]} \|_2^2 \\ = &\| \Lambda^T \cdot X \|_1 + [\nabla H(X^{[k]})^T \cdot (X - X^{[k]})] + \frac{1}{2\delta^{[k]}} \cdot (X - X^{[k]})^T \cdot (X - X^{[k]})\end{aligned} \tag{23}$$

在求最小值时可以忽略常数项，因此有：

$$\begin{aligned}&\min(\| Y - A \cdot X \|_2^2 + \| \Lambda^T \cdot X \|_1) \\ \Leftrightarrow &\min \| \Lambda^T \cdot X \|_1 + \nabla H(X^{[k]})^T (X - X^{[k]}) + \frac{1}{2\sigma^{[k]}} \| (X - X^{[k]}) \|_2^2 + \frac{1}{2} \delta_k \| \nabla H(X^{[k]}) \|_2^2 \\ \Leftrightarrow &\min \| \Lambda^T \cdot X \|_1 + \frac{1}{2\sigma^{[k]}} \| (X - X^{[k]}) + \delta^{[k]} \nabla H(X^{[k]}) \|_2^2 \\ \Leftrightarrow &\min \| \Lambda^T \cdot X \|_1 + \frac{1}{2\sigma^{[k]}} \| X - [X^{[k]} - \delta^{[k]} \nabla H(X^{[k]})] \|_2^2 \\ \Leftrightarrow &\min \| \Lambda^T \cdot X \|_1 + \frac{1}{2\sigma^{[k]}} \| X - X^{[k+1]} \|_2^2, \quad X^{[k+1]} = X^{[k]} - \delta^{[k]} \cdot \nabla H(X^{[k]}) \\ \Leftrightarrow &\min \sigma^{[k]} \| \Lambda^T \cdot X \|_1 + \frac{1}{2} \| X - X^{[k+1]} \|_2^2 \Leftrightarrow \min \| \Lambda'^T \cdot X \|_1 + \frac{1}{2} \| X - X^{[k+1]} \|_2^2\end{aligned} \tag{24}$$

以求解x_1为例：

$$\begin{aligned}&f(x) = \lambda_1^{'} \cdot |x| + \frac{1}{2}(x - x^{[k+1]})^2 \quad x_1 = \arg\min_x [f(x)] \\ &x \geq 0: \\ &\quad f(x) = \lambda_1^{'} \cdot x + \frac{1}{2}(x - x^{[k+1]})^2 \quad \frac{df}{dx} = \lambda_1^{'} + x - x^{[k+1]} \\ &\quad \Rightarrow \begin{cases} x_1 = x^{[k+1]} - \lambda_1^{'} & (x^{[k+1]} \geq \lambda_1^{'}) \\ x_1 = 0 & (x^{[k+1]} < \lambda_1^{'}) \end{cases} \\ &x < 0: \\ &\quad f(x) = -\lambda_1^{'} \cdot x + \frac{1}{2}(x - x^{[k+1]})^2 \quad \frac{df}{dx} = -\lambda_1^{'} + x - x^{[k+1]} \\ &\quad \Rightarrow \begin{cases} x_1 = x^{[k+1]} + \lambda_1^{'} & (x^{[k+1]} \leq -\lambda_1^{'}) \\ x_1 = 0 & (x^{[k+1]} > -\lambda_1^{'}) \end{cases}\end{aligned} \tag{25}$$

综上所述，以下为式 20 的解：

$$x_i^* = \begin{cases} \text{sign}(x_i^{[k+1]}) \cdot (|x_i^{[k+1]}| - \lambda_i^{'}) & |x_i^{[k+1]}| \geq \lambda_i^{'} \\ 0 & |x_i^{[k+1]}| < \lambda_i^{'} \end{cases} \tag{26}$$

如果将式 20 的解再次代入式 20，则有以下的迭代公式：

$$X^{[k+1]} = G(X^{[k]}) \Leftrightarrow \begin{cases} \Lambda^{[k]} = \lambda \cdot [\frac{\alpha}{|x_1^{[k]}| + \iota}, \frac{\alpha}{|x_2^{[k]}| + \iota}, \cdots]^T = [\lambda_1^{[k]}, \lambda_2^{[k]}, \cdots]^T \\ X^{[k+1]} = \arg\min_X (\| Y - A \cdot X \|_2^2 + \| (\Lambda^{[k]})^T \cdot X \|_1) \end{cases} \tag{27}$$

可知，从初值$X^{[0]}$开始，通过式 27 进行迭代，，$X^{[k]}$的稀疏性会逐渐增加，最终得到一个足够稀疏的解。其中，初值$X^{[0]}$可以使用式 18 的解。

4.3.2. MATLAB 实现

```
function x = deconv2_sparse(y, h, lambda)
[row_y, col_y] = size(y);
[row_h, col_h] = size(h);
row_x = row_y - row_h + 1;
col_x = col_y - col_h + 1;

h_tilde = rot90(h, 2);

weight = lambda * ones(row_x, col_x);
% half_row = (row_h - 1) / 2;
% half_col = (col_h - 1) / 2;

% 设置初值
x = zeros(row_x, col_x);
% x = y(half_row + 1:end - half_row, half_col + 1:end - half_col);

% 设置迭代参数
lr = 0.02;
epoch_1 = 500;
epoch_2 = 5;
epoch_3 = 500;

% 第一次迭代，等权重
for i_1 = 1:epoch_1
    grad = conv2(h_tilde, (conv2(x, h, 'full') - y), 'full');
    grad = grad(row_h:row_y, col_h:col_y);
    descent = lr .* grad;
    x = x - descent;
    x = (abs(x) > weight) .* sign(x) .* (abs(x) - weight);
end

for i_2 = 1:epoch_2
%     根据上一轮迭代的解调整稀疏性权重
    weight = lambda * 1 ./ (x + 1e-5);
    for i_3 = 1:epoch_3
        grad = conv2(h_tilde, (conv2(x, h, 'full') - y), 'full');
        grad = grad(row_h:row_y, col_h:col_y);
        descent = lr .* grad;
        x = x - descent;
        x = (abs(x) > weight) .* sign(x) .* (abs(x) - weight);
```

```
41        end
42    end
43
44    end
```

5. 实验内容和结果

5.1. 主要实验内容

生成图像，对图像进行高斯模糊并添加高斯白噪声。

对添加了噪声的图像分别使用快速梯度下降，光滑性逆卷积和稀疏性逆卷积进行去模糊。其中光滑性逆卷积和稀疏性逆卷积中使用了不同的权重，以比较其结果。

为了提高计算速度，在逆卷积计算中使用了 MATLAB 提供的 GPU 数组功能，并且将数据转换为使用单精度浮点表示。

对逆卷积得到的结果进行了动态范围的调整，将其动态范围调整至 0-1。

5.2. MATLAB 实现

5.2.1. 图像生成、卷积模糊和逆卷积

```
1    clc
2    clear all
3    close all
4    addpath(genpath('./Function_Assignment_1'))
5    %%
6    % 生成原始数据
7    SNR = 0;
8    img = Test_Image();
9    h = Conv_Core(45, 45, 15);
10   y = conv2(img, h, 'full');
11   y_N = Add_Noise(y, SNR);
12   figure, imshow(img)
13   title('origin image')
14
15   figure, imshow(y)
16   title('image without noise')
17
18   figure, imshow(y_N)
19   title('image with noise')
20
21   figure, imagesc(h)
22   title('conv core')
23   axis equal
24   axis off
25   saveas(gca, './Image/conv core.svg')
26
```

```
pause(1)

imwrite(img, './Image/test image.tif');
imwrite(y, './Image/conv image without noise.tif');
imwrite(y_N, './Image/conv image with noise.tif')

%%
% 使用 gpu 数组功能，提高逆卷积计算效率
y_N = single(y_N);
y_N = gpuArray(y_N);
h = single(h);
h = gpuArray(h);

%%
% 快速梯度下降
x_1 = deconv2_fast_grad_descent(y_N, h);
x_1 = gather(x_1);
x_1 = Rescale(x_1);
figure, imshow(x_1)
title('deconv grad descent')
imwrite(x_1, './Image/deconv grad descent.tif')
pause(1)

%%
% 光滑性逆卷积
x_L2_1 = deconv2_smooth(y_N, h, 1);
x_L2_1 = gather(x_L2_1);
x_L2_1 = Rescale(x_L2_1);
figure, imshow(x_L2_1)
title('deconv smooth 1')
imwrite(x_L2_1, './Image/deconv smooth 1.tif')
pause(1)

x_L2_2 = deconv2_smooth(y_N, h, 0.01);
x_L2_2 = gather(x_L2_2);
x_L2_2 = Rescale(x_L2_2);
figure, imshow(x_L2_2)
title('deconv smooth 0.05')
imwrite(x_L2_2, './Image/deconv smooth 0.01.tif')
pause(1)

x_L2_3 = deconv2_smooth(y_N, h, 10);
x_L2_3 = gather(x_L2_3);
x_L2_3 = Rescale(x_L2_3);
```

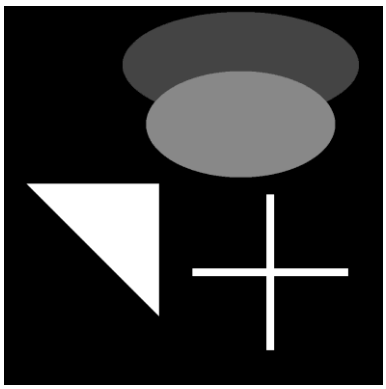
```
figure, imshow(x_L2_3)
title('deconv smooth 10')
imwrite(x_L2_3, './Image/deconv smooth 10.tif')
pause(1)

%%
% 稀疏性逆卷积
x_L1_1 = deconv2_sparse(y_N, h, 1e-3);
x_L1_1 = gather(x_L1_1);
x_L1_1 = Rescale(x_L1_1);
figure, imshow(x_L1_1)
title('deconv sparse 1e-3')
imwrite(x_L1_1, './Image/deconv sparse 1e-3.tif')
pause(1)

x_L1_2 = deconv2_sparse(y_N, h, 5e-4);
x_L1_2 = gather(x_L1_2);
x_L1_2 = Rescale(x_L1_2);
figure, imshow(x_L1_2)
title('deconv sparse 5e-4')
imwrite(x_L1_2, './Image/deconv sparse 5e-4.tif')
pause(1)

x_L1_3 = deconv2_sparse(y_N, h, 2.5e-4);
x_L1_3 = gather(x_L1_3);
x_L1_3 = Rescale(x_L1_3);
figure, imshow(x_L1_3)
title('deconv sparse 2.5e-4')
imwrite(x_L1_3, './Image/deconv sparse 2.5e-4.tif')
pause(1)
```

5.2.2. 动态范围调整

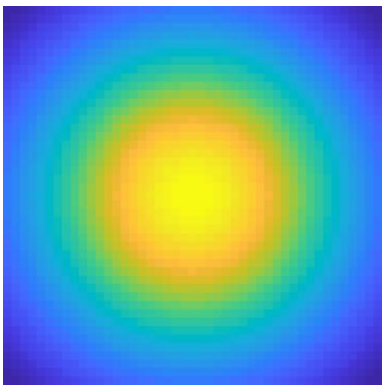
```
function y = Rescale(x)
x = double(x);
MAX = max(x(:));
MIN = min(x(:));
x = x - MIN;
y = x ./ (MAX - MIN);
end
```

5.3. 结果

5.3.1. 图像与卷积

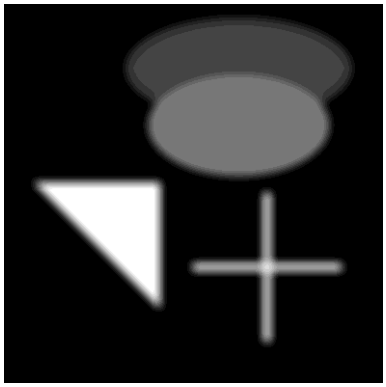


(a)

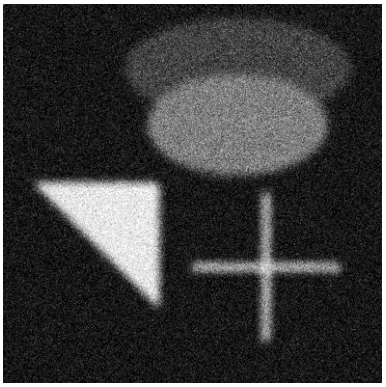


(b)

图 1：(a)，待卷积图像；(b)，卷积核



(a)



(b)

图 2：(a)，卷积后图像，无噪声；(b)卷积后图像，有噪声

5.3.2. 逆卷积结果

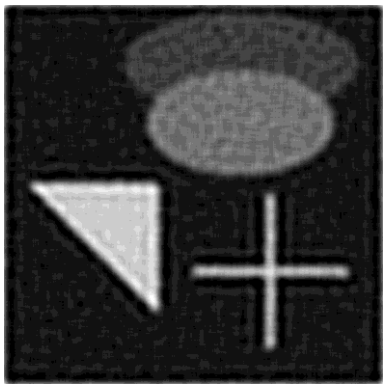
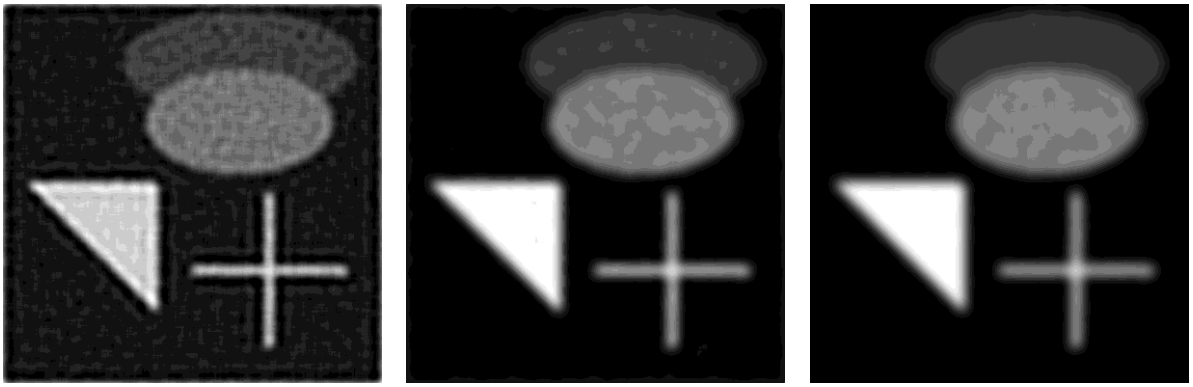
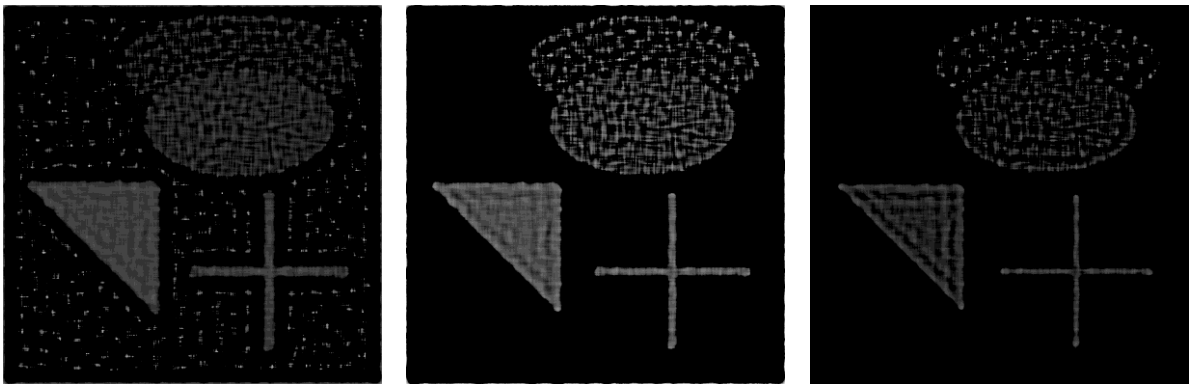


图 3：快速梯度下降逆卷积



$\lambda = 0.01$　　$\lambda = 1$　　$\lambda = 10$

图 4：光滑性逆卷积



$\lambda = 2.5 \times 10^{-4}$　　$\lambda = 5 \times 10^{-4}$　　$\lambda = 1 \times 10^{-3}$

图 5：稀疏性逆卷积

5.4. 实验结果分析

实验中设置的信噪比为 0dB，即信号和噪声的功率相同。在此情况下，通过快速梯度下降法得到的结果中有明显的噪声。

通过在逆卷积时加入光滑性的要求，得到的结果中噪声要明显比使用快速梯度下降法时的噪声小。其中光滑性的权重越大，则噪声越小。但是得到的图像的边缘也约模糊。

由于原始图像不具有稀疏性，因此使用稀疏性逆卷积得到的图像和原图相比差异最大。稀疏性权重对重建结果也有很大影响。太小的稀疏性权重，会导致重建结果中出现非常多的噪声，甚至全部为 1。而太大的权重会导致重建结果中 0 的比重太大，一些信号强度较小的地方无法重建出来。

添加光滑性和稀疏性后，有可能导致重建得到的图像亮度整体偏小。因此，需要重新调整图像的动态范围，以达到更精确的结果。

6. 实验心得

逆卷积的求解问题中有很多细节问题需要考虑。

一方面需要结合课程内容，得到求解逆卷积问题的表达式，并通过程序将其实现。在程序实现时，还需要考虑计算速度等因素。另一方面，需要注意计算方法中对于梯度下降率以及光滑性逆卷积中光滑性相对的权重等参数的值。通过调整这些参数，得到的结果也有可能有很大的差异。由于实验时间有限，对于这些参数的选择还有很多地方考虑不足。例如能否根据卷积核直接计算得到一个可用的梯度下降率等。这些问题需要进一步的思考与优化。