# DIP 第一次实验报告

一、 实验目的

a) 图象直方图均衡化: 尽可能保证图像的直方图尽可能覆盖全部可能的灰度级并且分布均匀，那么该图像就有了高对比度和丰富的灰度色调。

b) 滤波:

i. 均值滤波：它是指在图像上对目标像素给一个模板，该模板包括了其周围的临近像素（以目标像素为中心的周围 8 个像素，构成一个滤波模板，即去掉目标像素本身），再用模板中的全体像素的平均值来代替原来像素值。

ii. 中值滤波： 是一种非线性平滑技术，它将每一像素点的灰度值设置为该点某邻域窗口内的所有像素点灰度值的中值.

c) 锐化:

二、 实验原理

a) 直方图均衡化:

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j, \quad k = 0, 1, 2, \cdots, L-1 \tag{3.3-13}$$

与前面一样，其中 $MN$ 是图像的像素总数，$n_j$ 是具有灰度值 $r_j$ 的像素数，$L$ 是图像中可能的灰度级数。类似地，给定一个规定的 $s_k$ 值，式(3.3-11)的离散形式涉及计算变换函数

$$G(z_q) = (L-1)\sum_{i=0}^{q} p_z(z_i) \tag{3.3-14}$$

对一个 $q$ 值，有

$$G(z_q) = s_k \tag{3.3-15}$$

b) 滤波:

i. 均值滤波：

$$R=\frac{1}{9}\sum_{i=1}^{9}z_i$$

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

ii. 中值滤波：

$$\text{mid}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

c) 锐化：

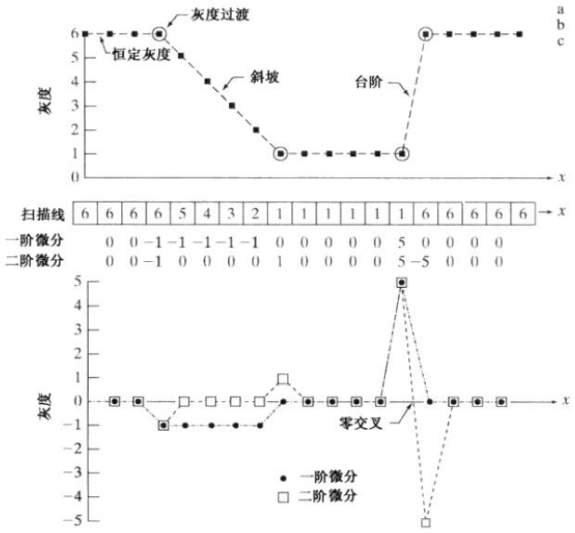



图 3.36 表示一幅图像中一段水平灰度剖面的一维数字函数的一阶微分和二阶微分的说明。在图(a)和图(c)中，为便于观看，已用虚线将数据点连接起来

可以证明(Rosenfeld and Kak[1982])，最简单的各向同性微分算子是拉普拉斯算子。一个二维图像函数 $f(x,y)$ 的拉普拉斯算子定义为

$$\nabla^2 f=\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2} \tag{3.6-3}$$

因为任意阶微分都是线性操作，所以拉普拉斯变换也是一个线性算子。为了以离散形式描述这一公式，我们使用式(3.6-2)的定义，记住，我们必须支持第二个变量。在 $x$ 方向上，我们有

$$\frac{\partial^2 f}{\partial x^2}=f(x+1,y)+f(x-1,y)-2f(x,y) \tag{3.6-4}$$

类似地，在 $y$ 方向上我们有

$$\frac{\partial^2 f}{\partial y^2}=f(x,y+1)+f(x,y-1)-2f(x,y) \tag{3.6-5}$$

所以，遵循这三个公式，两个变量的离散拉普拉斯算子是

$$\nabla^2 f(x,y)=f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)-4f(x,y) \tag{3.6-6}$$

## 三、 代码分析

a) myHisteq.m

b) myAverage.m

c) myMedian.m

d) mySharpen.m

## 四、 实验结果与分析

a) 直方图均衡化

i. 记录原图像的灰度频率

```
pre = zeros(1,256);
for i = 1:h
    for j = 1:w
        k = img_1(i,j)+1;
        pre(1,k) = pre(1,k) + 1;
    end
end
```

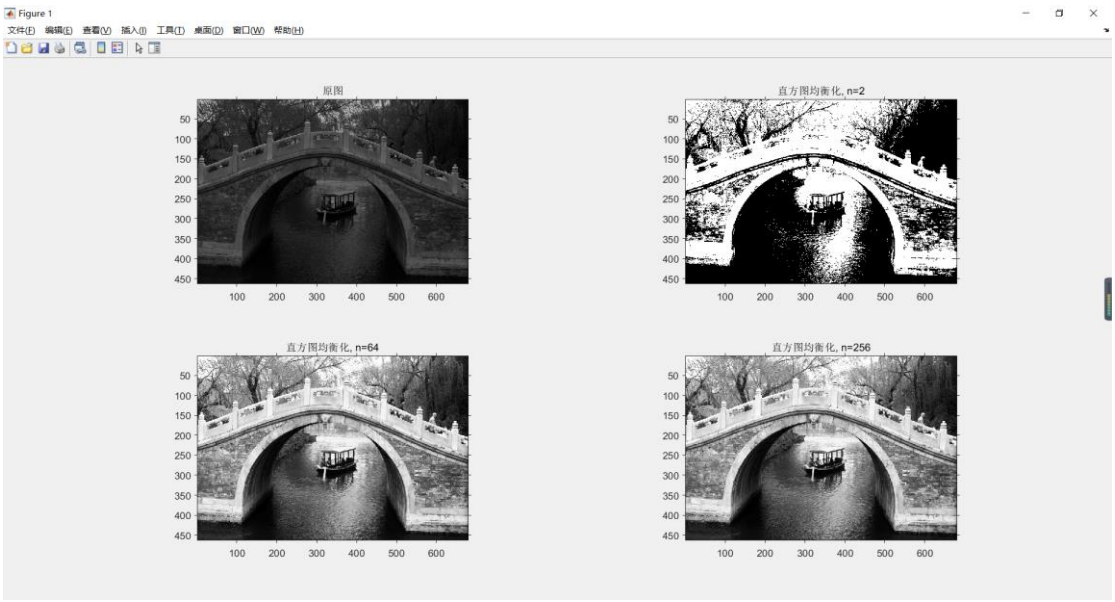
ii. 计算初始直方图的累计分布函数值，取整取 n 级灰度

```
count = zeros(1,256);
sum = 0;
for i = 1:256
   sum = sum + pre(1,i);
   count(1,i) =round(sum*(n-1)/(h*w));
end
```

iii. 映射

```
for i = 1:h
    for j = 1:w
        k = img_1(i,j);
        img_2(i,j) = count(1,k+1);
    end
end
```

iv. 实验结果




可以看出 n 越大，直方图均衡化的效果越好：不仅提高了图像的对比度，同时凸显了图像的细节，是很好的作为图像预处理的工具。

b) 均值滤波与中值滤波

i. 0 增广矩阵（考虑边界条件）

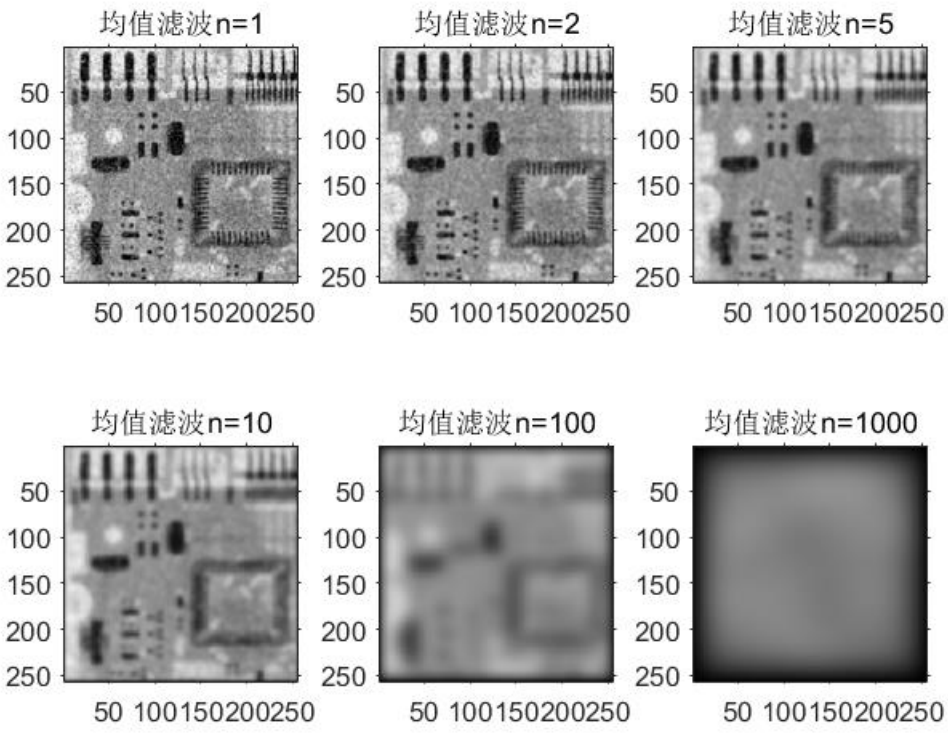
```
exp_img_2 = zeros(h+2, w+2);
exp_img_2(2:h+1, 2:w+1) = img_1;
```

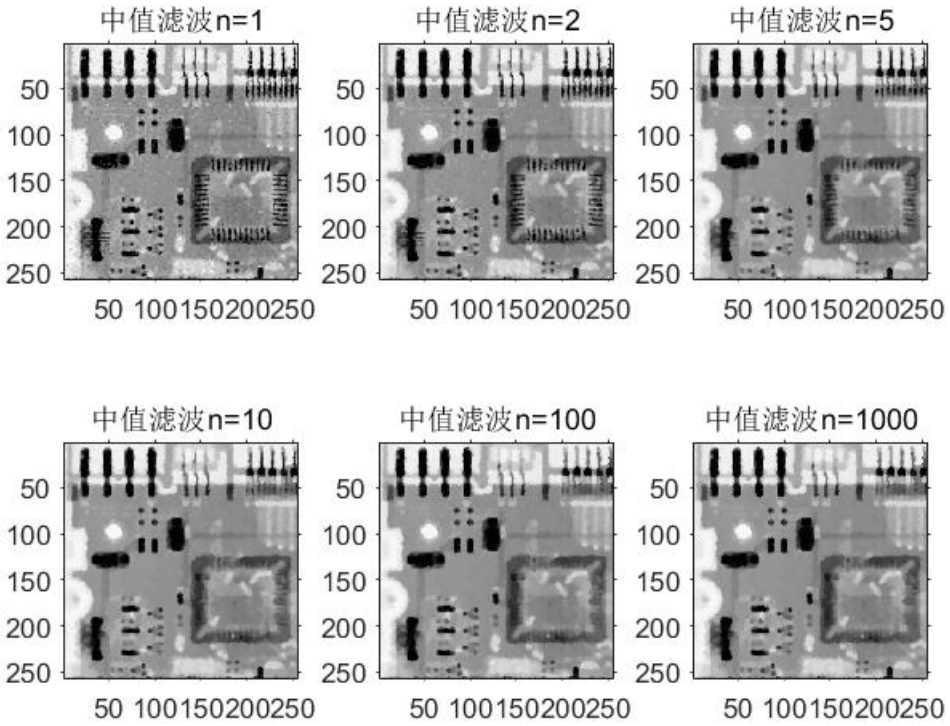
ii. 对于每个像素点求取均值（中值）

```
function [value] = mid(im, i , j)
% 求中值
array=zeros(1,9);
k=1;
for m = -1:1
    for n = -1:1
    array(1,k) = im(i+m,j+n);
    k=k+1;
    end
end
for m = 1 : 5
    for n = m + 1 : 9
        if array(1,m) > array(1,n);
            temp = array(1,n);
            array(1,n) = array(1,m);
            array(1,m) = temp;
        end
    end
end
value =array(1,5);
end
```

```
function [value] = ave(exp_img_2,i,j)
    value = exp_img_2(i-1,j-1) + exp_img_2(i-1,j) + exp_img_2(i-1,j+1) +
    value = value / 9;
end
```

## iii. 实验结果






可以看出均值滤波在迭代无穷多次时（1000 次），图像变得近乎模糊。而中值滤波在迭代无穷多次时（1000 次），图象几乎没有发生变化（肉眼不可见）。

上述两种算法的在无穷多次迭代后的结果比较中发现：均值滤波不能实现多次迭代，否则会损失细节信息。而中值滤波既可以保证噪音的去除，也可以保证细节的一定程度保存。这是由于算法不同导致的：均值算法相当于对图象灰度值进行平均操作，而中值滤波相对并未改变图像的灰度值（仅仅是取中值）。

c) 图像锐化

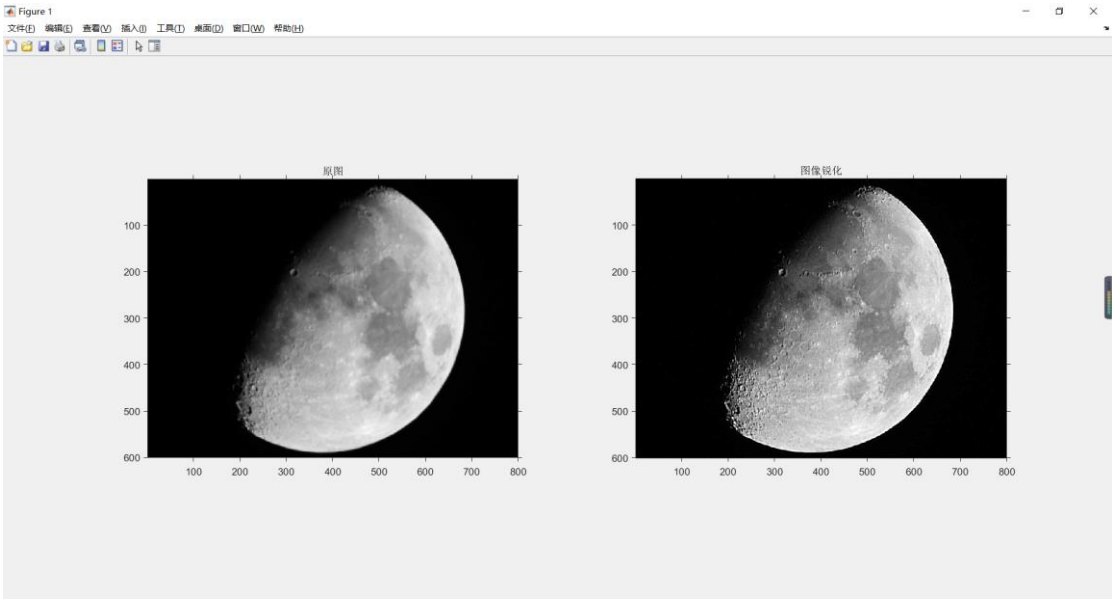
i. 0 增广矩阵（考虑边界条件）

```
exp_img_2 = zeros(h+2,w+2);
exp_img_2(2:h+1,2:w+1) = img_1;
```

ii. 对于每个像素块，采取拉普拉斯算子卷积

```
lp = [-1 -1 -1; -1 8 -1; -1 -1 -1];
for i = 2 : h+1
    for j = 2: w+1
        sum = 0;
        for m = -1 : +1
            for n = -1 : +1
                sum = sum + lp(2+m,2+n)*exp_img_2(i+m,j+n);
            end
        end
        img_2(i-1,j-1) = exp_img_2(i,j) + sum;
    end
end
```

iii. 实验结果




可以看出，图像对拉普拉斯算子做卷积能够凸显图像的边缘与细节。