1.习题

1.1 直方图均衡化

答：

结果会与第一次均衡化的结果一样。根据直方图均衡化公式

$$S_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j)$$

进行一次直方图均衡化后得到 S_k，再进行第二次均衡化时，由于（L-1）不变，$\sum_{j=0}^{k} p_r(S_k)$ 不变，所以得出的结果也跟第一次一样。

1.2 空间滤波

1）$\begin{bmatrix} 177 & 420 & 271 & 263 \\ 75 & 218 & 249 & 107 \\ -131 & -324 & -107 & -133 \\ -173 & -336 & -362 & -207 \end{bmatrix}$

2）正数表示该点上方像素值大于下方像素值

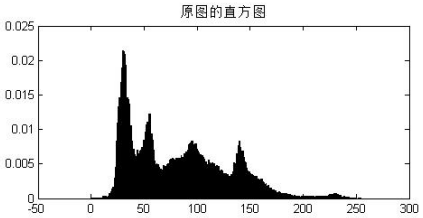
负数表示该点下方像素值大于上方像素值

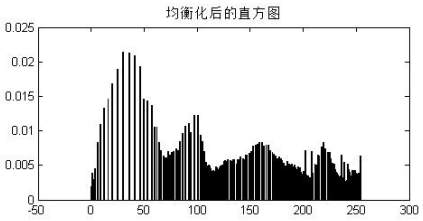
3）可以用于提取水平边缘。

2 编程题

2.2 直方图均衡化

1） & 2）





3）答

由图中的直方图可看出，经过直方图均衡化后，直方图变得更为平均，像素值不再集中在某几个范围；但是并没有做到绝对的平均，由图中可以看出，在原来相对较多的像素范围，在平衡后频率依然是相比较高；而且，均衡化的直方图出现了一些“缝隙”，这是因为在经过直方图均衡化后，一些原有的像素值缺失所导致。

4）答

①显示直方图。这里我先用了一个 data 数组，用于存储每个像素值的像素点个数。例如 data[1]的值为像素值为 1 的点的个数。接着对 data[i] = data[i]/m*n，(其中 m、n 为原图的长和宽)从而得出每个像素值的频率。

```
for x=1:src_col
    for y=1:src_row
        data(img(y,x)+1) = data(img(y,x)+1)+1;
    end
end
```

②均衡化。对于 output_img 的每个像素点，先计算其像素值 X 的 pdf(即$\sum_{i=0}^{X} data(i)$)，然后根据公式

$$S_k = T(r_k) = (L-1)\sum_{i=0}^{X} data(i)$$

这里 L = 256 得出均衡化后该像素点的像素值

```
for x=1:src_col
    for y=1:src_row
        sum = 0;
        for i = 1:img(y,x)+1
            sum = sum + data(i);
        end
        output_img(y,x) = round(double(sum*255));
        result(output_img(y,x)+1) = result(output_img(y,x)+1)+1;
    end
end
```

2.3 空间滤波

1）

原图

处理后的图

9*9 均值滤波器处理

原图

处理后的图

7*7 均值滤波器处理

原图

处理后的图

11*11 均值滤波器处理

2）

使用的滤波器：

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

原图

处理后的图

拉普拉斯滤波器处理

拉普拉斯是一种微分算子，因此其强调的是图像中灰度的突变而不是灰度级别缓慢变化的区域。这样便可以产生浅灰色的变现和突变点叠加到暗色背景中的图像。然后再加上原图像，便可以加强变化突变的边缘，从而产生锐化处理的效果。

3）

使用的平滑部分滤波器：

1/9*

1	1	1
1	1	1
1	1	1

使用的 k = 3

根据公式$g(x,y) = f(x,y) + k * g_{max}(x,y)$ ， $g_{max}(x,y) = f(x,y) - \bar{f}(x,y)$ 可以推导出输入的滤波器为：

-k/9	-k/9	-k/9
-k/9	8k/9 + 1	-k/9
-k/9	-k/9	-k/9

高提升滤波处理

4）答

已知一个滤波器，长跟宽均为 filter_size,令 n = (filter_size-1)/2

对于每个像素点(x,y)，在进行滤波时对于该点一定范围内进行相关运算。根据计算公式推导出滤波器的元素(i,j)对应的是图像中像素点(x-n+i-1,y-n+j-1)。例如一个 3*3 的滤波器，n=1，对于像素点(3,4)，与滤波器(1,2)元素相乘的是(3-1+1-1,4-1+2-1) = (2,4)像素点。

```
sum = 0;
n = (filter_col-1)/2;
for i = 1:filter_col
    for j = 1:filter_row
        a = x-n+i-1;
        b = y-n+j-1;
        %先判断像素点是否处于边缘，如果处于边缘，则补0。
        if ( a >= 1 && b >= 1 && a <= src_col && b <= src_row)
            sum = sum + filter(j,i)*img(b,a);
        else sum= sum+0;
        end
    end
end
```

注意到当像素点处于或者接近边缘时，进行相关操作时可能会出现越界情况，因此在做相关运算求和时，先判断点(x-n+i-1,y-n+j-1)是否越界，如果出现越界，则将该点的像素值当

做为 0。

```
if ( a >= 1 && b >= 1 && a <= src_col && b <= src_row)
    sum = sum + filter(j,i)*img(b,a);
else sum= sum+0;
end
```

最后由于该代码要同时满足 3 题，且拉普拉斯滤波时，根据滤波器中心值的符号进行加减滤波器相乘的结果，所以在最后对中心值进行判断，并相应改变输出值的正负号。

```
if(filter(n+1,n+1) < 0)
    output_img(y,x) = uint8(-sum);
else output_img(y,x) = uint8(sum);
end
```

(Ps:在输入拉普拉斯滤波器时要将输入矩阵的中心点的绝对值加 1)