数字图像处理实验报告

张子宁　U202215126　人工智能 2204 班

2024 年 10 月 20 日

1　理论作业

已知一幅数字图像为：

$$\text{原始矩阵} = \begin{bmatrix} 1 & 2 & 1 & 4 & 3 \\ 1 & 10 & 2 & 3 & 4 \\ 5 & 2 & 6 & 8 & 8 \\ 5 & 5 & 7 & 0 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix} \tag{1}$$

分别用 3×3 大小的均值滤波器和中值滤波器对图像进行平滑处理，给出处理步骤及结果图像。（注：图像边框像素保留不变）

1.1　均值滤波计算步骤

均值滤波的计算方法是，对于每个像素点（除了边框），用其周围 3x3 区域内的所有像素的均值来替代该像素值。具体步骤如下：

1. ** 遍历内部像素 **：从第二行到倒数第二行，从第二列到倒数第二列，计算每个像素点的 3x3 均值。2. ** 保持边框不变 **：边框的像素点不参与计算，保持原值。

1.2　计算每个内部像素的均值

- **(1,1)**: $(1+2+1+1+10+2+5+2+6)/9 = 3.33 \approx 3$
- **(1,2)**: $(2+1+4+10+2+3+2+6+8)/9 = 4.22 \approx 4$

- **(1,3)**: $(1+4+3+2+3+4+6+8+8)/9 = 4.33 \approx 4$
- **(2,1)**: $(1+10+2+5+2+6+5+5+7)/9 = 4.77 \approx 5$
- **(2,2)**: $(10+2+3+2+6+8+5+7+0)/9 = 4.77 \approx 5$
- **(2,3)**: $(2+3+4+6+8+8+7+0+8)/9 = 5.11 \approx 5$
- **(3,1)**: $(5+2+6+5+5+7+5+6+7)/9 = 5.33 \approx 5$
- **(3,2)**: $(2+6+8+5+7+0+6+7+8)/9 = 5.44 \approx 5$
- **(3,3)**: $(6+8+8+7+0+8+7+8+9)/9 = 6.77 \approx 7$

1.3 结果矩阵

经过均值滤波后的矩阵（边框不变）为：

$$\text{结果矩阵} = \begin{bmatrix} 1 & 2 & 1 & 4 & 3 \\ 1 & 3 & 4 & 4 & 4 \\ 5 & 4 & 5 & 5 & 8 \\ 5 & 5 & 5 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix} \tag{2}$$

1.4 中值滤波计算步骤

中值滤波的计算方法是，对于每个像素点（除了边框），用其周围 3x3 区域内的所有像素的中值来替代该像素值。具体步骤如下：

1. ** 遍历内部像素 **：从第二行到倒数第二行，从第二列到倒数第二列，计算每个像素点的 3x3 中值。2. ** 保持边框不变 **：边框的像素点不参与计算，保持原值。

1.5 计算每个内部像素的中值

以下是对每个内部像素点的 3x3 区域的中值计算：

- **(1,1)**: $\{1,2,1,1,10,2,5,2,6\}$ 中值为 2
- **(1,2)**: $\{2,1,4,10,2,3,2,6,8\}$ 中值为 3

- **(1,3)**: $\{1,4,3,2,3,4,6,8,8\}$ 中值为 4
- **(2,1)**: $\{1,10,2,5,2,6,5,5,7\}$ 中值为 5
- **(2,2)**: $\{10,2,3,2,6,8,5,7,0\}$ 中值为 5
- **(2,3)**: $\{2,3,4,6,8,8,7,0,8\}$ 中值为 6
- **(3,1)**: $\{5,2,6,5,5,7,5,6,7\}$ 中值为 5
- **(3,2)**: $\{2,6,8,5,7,0,6,8,7\}$ 中值为 6
- **(3,3)**: $\{6,8,8,7,0,8,7,8,9\}$ 中值为 8

1.6 结果矩阵

经过中值滤波后的矩阵（边框不变）为：

$$\text{结果矩阵} = \begin{bmatrix} 1 & 2 & 1 & 4 & 3 \\ 1 & 2 & 3 & 4 & 4 \\ 5 & 5 & 5 & 6 & 8 \\ 5 & 5 & 6 & 8 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix} \tag{3}$$

2 编程作业

2.1 将彩色图灰度化

```
% 1.读取低照度图像
filename = 'picture.jpg';  % 低照度图像文件名
img = imread(filename);  % 读取图像
[rows, cols, channels] = size(img);  % 获取图像的大小

% 1. 将图像转换为灰度图
gray_img = zeros(rows, cols);  % 初始化灰度图
for r = 1:rows
        for c = 1:cols
                % 采用加权平均法转换为灰度
```

```
11                          gray_img(r, c) = 0.2989 * img(r, c, 1)
                                + 0.5870 * img(r, c, 2) + 0.1140 *
                                img(r, c, 3);
12                  end
13          end
14
15          % 显示灰度图像
16          figure;
17          imshow(uint8(gray_img));
18          title('灰度图像');
19          saveas(gcf, 'gray_picture.png');  % 保存灰度图像
```

图 1: 灰度图像

2.2 获得灰度直方图

```
1           % 2. 计算并显示灰度直方图
2           histogram = zeros(1, 256);  % 初始化直方图
3           for r = 1:rows
4                   for c = 1:cols
5                           intensity = uint8(gray_img(r, c));  %
                                获取灰度值
```

```
                histogram(intensity + 1) = histogram(
                    intensity + 1) + 1;  % 统计灰度值出
                    现的频率
            end
        end

        % 显示灰度直方图
        figure;
        bar(0:255, histogram);
        title('灰度直方图');
        xlabel('灰度值');
        ylabel('像素数量');
        saveas(gcf, 'gray_histogram.png');  % 保存灰度直方图
```

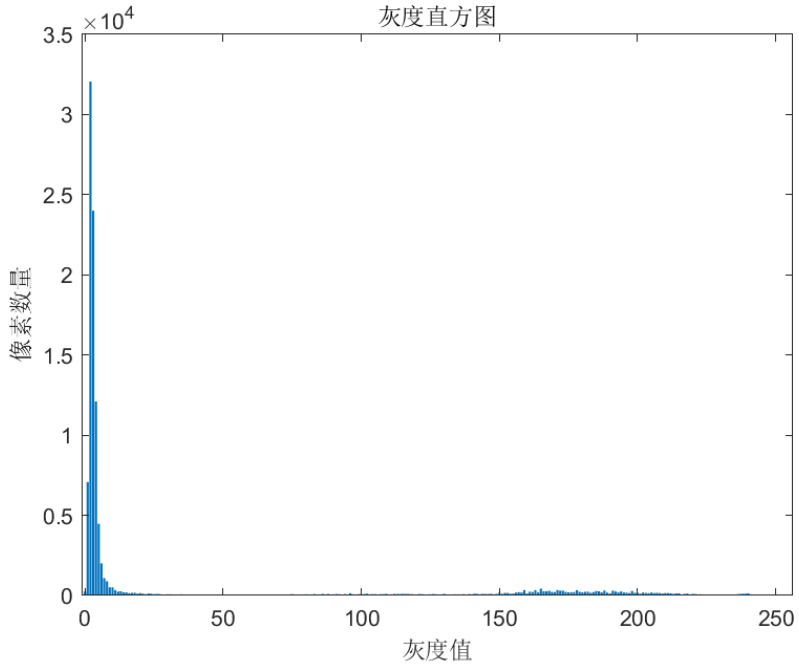

图 2: 原图的灰度直方图

2.3 获得离散傅里叶变换频谱幅度图

```
        % 3. 计算快速傅里叶变换频谱幅度图FFT
```

```
N = rows * cols;  % 图像的总像素数
fft_img = zeros(N, 1);  % 初始化FFT结果

% 将图像展开为列向量
gray_img_vector = gray_img(:);

for u = 1:N
        sum = 0;  % 初始化和
        for x = 1:N
                % 计算FFT公式
                sum = sum + double(gray_img_vector(x))
                    * exp(-2 * pi * 1i * (u-1) * (x-1)
                    / N);
        end
        fft_img(u) = sum;  % 存储FFT结果
end

% 频谱中心化
fft_img_shifted = zeros(rows, cols);  % 初始化中心化后
    的FFT结果
for u = 1:rows
        for v = 1:cols
                fft_img_shifted(u, v) = fft_img(mod(u +
                     rows/2 - 1, rows) + 1 + (mod(v +
                    cols/2 - 1, cols) + 1 - 1) * rows);
        end
end

% 计算频谱幅度
magnitude_spectrum = log(1 + abs(fft_img_shifted));  %
    计算频谱幅度

% 显示频谱幅度图
figure;
imshow(magnitude_spectrum, []);
title('FFT频谱幅度图');
saveas(gcf, 'fft_magnitude_spectrum.png');  % 保存频谱
    幅度图
```

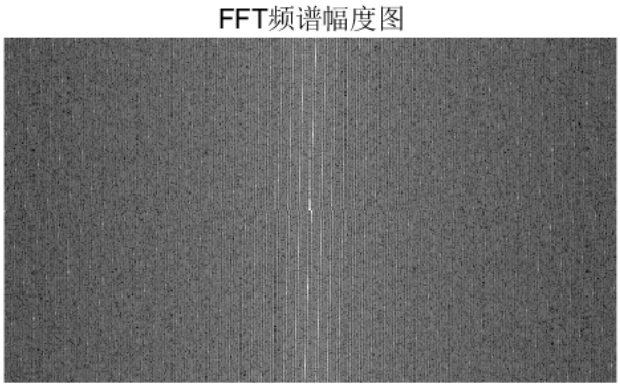

图 3: 离散傅里叶变换频谱幅度图

2.4 直方图均衡化

```
% 4. 直方图均衡化
cdf = cumsum(histogram);  % 计算累积分布函数
cdf_normalized = cdf / max(cdf);  % 归一化
equalized_img = zeros(size(gray_img));  % 初始化均衡化图像

% 根据CDF进行均衡化
for r = 1:rows
        for c = 1:cols
                % 应用均衡化
                equalized_img(r, c) = round(cdf_normalized(uint8(gray_img(r, c))) + 1) * 255);
        end
end

% 显示均衡化后的图像
figure;
imshow(uint8(equalized_img));
```

```
17          title('直方图均衡化');
18          saveas(gcf, 'equalized_image.png');  % 保存均衡化后的图
                像
19
20          % 计算并显示直方图均衡化后的灰度直方图
21          equalized_histogram = zeros(1, 256);  % 初始化均衡化后
                图像的直方图
22          for r = 1:rows  % 遍历每一行
23                  for c = 1:cols  % 遍历每一列
24                          intensity = uint8(equalized_img(r, c));
                                % 获取当前均衡化图像的灰度值
25                          equalized_histogram(intensity + 1) =
                                equalized_histogram(intensity + 1)
                                + 1;  % 更新直方图
26                  end
27          end
28
29          % 显示均衡化后的灰度直方图
30          figure;  % 创建新图形窗口
31          bar(0:255, equalized_histogram);  % 使用条形图显示均衡
                化后的灰度直方图
32          title('均衡化后的灰度直方图');  % 添加标题
33          xlabel('灰度值');  % x轴标签
34          ylabel('像素数量');  % y轴标签
35          saveas(gcf, 'equalized_histogram.png');  % 保存均衡化后
                的直方图为文件
```

图 4: 直方图均衡化图片

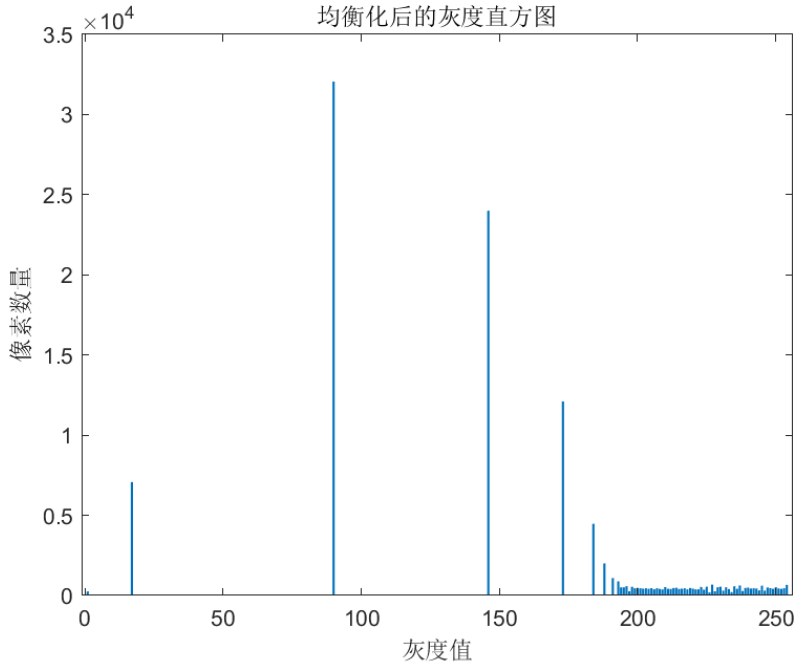

图 5: 直方图均衡化后的灰度直方图

2.5 同态滤波

```
% 5. 同态滤波处理
mean_val = mean(gray_img(:));  % 计算图像的平均值

% 将图像转换为频域
dft_equalized_img = zeros(N, 1);  % 初始化均值图的DFT
for u = 1:N
        sum = 0;  % 初始化和
        for x = 1:N
                sum = sum + double(equalized_img(x)) *
                    exp(-2 * pi * 1i * (u-1) * (x-1) /
                    N);
        end
        dft_equalized_img(u) = sum;  % 存储DFT结果
end

% 创建高通滤波器
H = zeros(rows, cols);  % 初始化高通滤波器
D0 = 30;  % 截止频率
for u = 1:rows
        for v = 1:cols
                Duv = sqrt((u - rows/2)^2 + (v - cols
                    /2)^2);  % 计算距离
                H(u, v) = 1 - exp(-(Duv^2) / (2 * D0^2)
                    );  % 生成高通滤波器
        end
end

% 应用同态滤波
filtered_dft = dft_equalized_img .* H(:);  % 进行滤波操
    作

% 进行逆FFT得到滤波后的图像
filtered_img = zeros(rows, cols);  % 初始化滤波后的图像
for x = 1:N
        sum = 0;  % 初始化和
        for u = 1:N
```

```
32                          sum = sum + filtered_dft(u) * exp(2 *
                                pi * 1i * (u-1) * (x-1) / N);
33                  end
34                  filtered_img(x) = abs(sum) / N;  % 存储逆FFT结
                        果
35          end
36
37          % 显示同态滤波后的图像
38          figure;
39          imshow(uint8(filtered_img));
40          title('同态滤波处理后的图像');
41          saveas(gcf, 'filtered_image.png');  % 保存同态滤波后的
                图像
42
43          % 计算同态滤波后的灰度直方图
44          filtered_histogram = zeros(1, 256);  % 初始化直方图
45          for r = 1:rows
46                  for c = 1:cols
47                          intensity = uint8(filtered_img(r, c));
                                % 获取灰度值
48                          filtered_histogram(intensity + 1) =
                                filtered_histogram(intensity + 1) +
                                1;  % 统计灰度值出现的频率
49                  end
50          end
51
52          % 显示同态滤波后的灰度直方图
53          figure;
54          bar(0:255, filtered_histogram);
55          title('同态滤波后的灰度直方图');
56          xlabel('灰度值');
57          ylabel('像素数量');
58          saveas(gcf, 'filtered_histogram.png');  % 保存同态滤波
                后的灰度直方图
```

图 6: 同态滤波图片

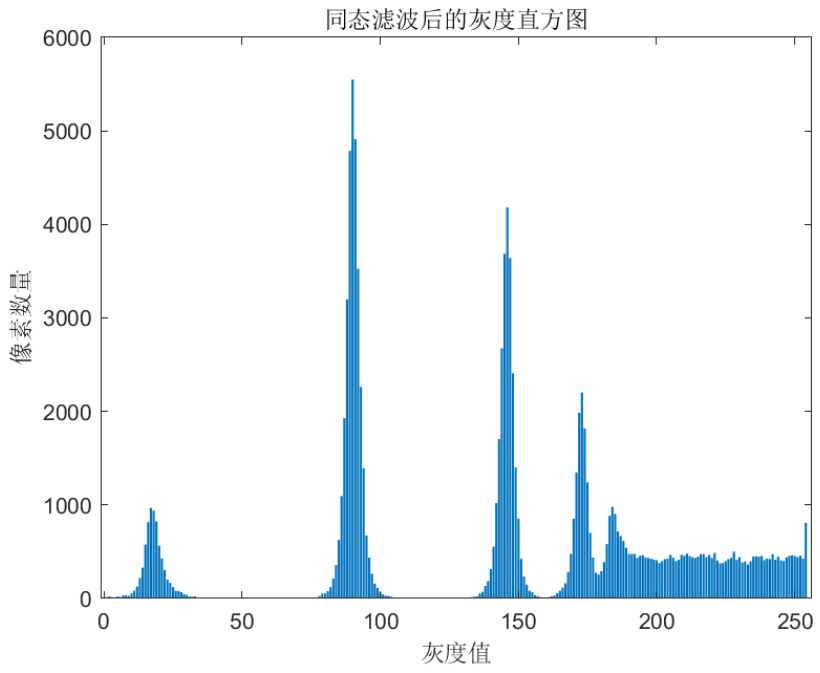

图 7: 同态滤波后的灰度直方图

2.6 效果比较

图 8: 两种算法的效果图比较

2.6.1 直方图均衡化

直方图均衡化是一种全局对比度增强技术，通过调整图像的灰度级分布，使图像的灰度直方图更加均匀，从而增加图像的对比度。它的优缺点如下：

- **优点：**
  - 简单易用，计算速度快。
  - 能有效增强低对比度图像中的细节，特别是暗部和亮部。
- **缺点：**
  - 对复杂图像的处理效果不佳，可能导致细节丢失或过度增强。
  - 全局调整可能不适合某些局部区域。
  - 可能会放大图像中的噪声。

2.6.2 同态滤波

同态滤波是一种基于频域的图像增强技术，通过分离图像的光照成分和反射成分，分别对它们进行处理，以改善图像的对比度。其优缺点如下：

- **优点：**
  - 局部增强，能自适应地调整对比度。
  - 有效抑制噪声，增强细节。
- **缺点：**
  - 计算复杂，处理速度较慢。
  - 需要根据具体图像调节参数。
  - 光照成分分离可能不完全。

2.6.3 比较总结

表 1: 直方图均衡化与同态滤波的比较

特性	直方图均衡化	同态滤波
处理方式	基于全局灰度分布，均衡化直方图	基于频域，分离光照与反射成分
优点	简单易实现；全局对比度提升	自适应性强，处理光照变化，去噪
缺点	不适合复杂光照变化；噪声可能放大	计算复杂，参数选择困难
应用场景	亮度低、对比度差的图像	光照不均、细节丰富的图像