

# 《数字图像处理与模式分析》平时作业

徐恺阳 523030910085

## 题目一：高斯滤波

### 算法描述

二维高斯滤波核的计算公式如下：

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

其中：

- $(x, y)$  表示卷积核中点的坐标。
- $\sigma$  (标准差) 是控制高斯分布宽度的关键参数。

高斯滤波的具体实现步骤如下：

- 输入：原始图像  $I$ , 卷积核大小  $k \times k$ , 标准差  $\sigma$ 。
  - 输出：平滑处理后的图像  $I'$ 。
1. **生成卷积核**: 根据给定的模板大小  $k$  和标准差  $\sigma$ , 利用上述数学公式计算出高斯卷积核矩阵  $G$ 。
  2. **遍历图像**: 对原始图像  $I$  中的每一个像素点  $(x, y)$  进行遍历。
  3. **初始化输出**: 将目标图像当前像素点的值  $I'(x, y)$  初始化为 0。
  4. **卷积运算**: 对于卷积核  $G$  内的每一个元素  $G(i, j)$ , 执行加权求和操作:

$$I'(x, y) = I'(x, y) + I(x - i, y - j) \times G(i, j)$$

这里  $I(x - i, y - j)$  代表对应邻域内的像素值。

5. **归一化处理**: 将计算得到的累加值  $I'(x, y)$  进行归一化处理, 确保存储到图像  $I'$  时数值处于有效的灰度范围内。
6. **返回结果**: 完成所有像素的处理后, 输出平滑后的图像  $I'$ 。

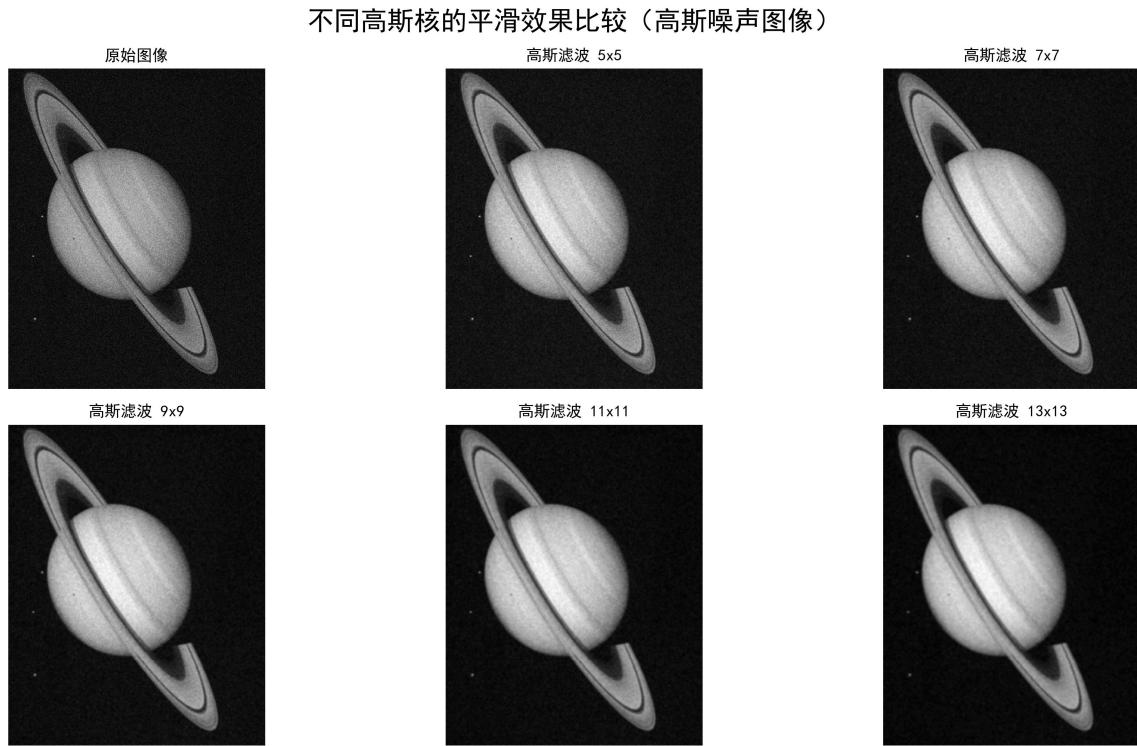
### 实验结果

对高斯多尺度平滑图像，应用  $5 \times 5$ ,  $7 \times 7$ ,  $9 \times 9$ ,  $11 \times 11$ ,  $13 \times 13$  的高斯卷积模板进行平滑，结果如下图所示。

不同高斯核的平滑效果比较（高斯多尺度平滑图像）



对高斯噪声图像，应用  $5 \times 5$ ,  $7 \times 7$ ,  $9 \times 9$ ,  $11 \times 11$ ,  $13 \times 13$  的高斯卷积模板进行平滑，结果如下图所示。



## 现象规律

### 1. 普通图像

- **平滑度与尺寸成正比**: 随着高斯卷积模板尺寸的增大，图像的平滑程度增强，图像越来越模糊；
- **小尺寸模板** (如  $5 \times 5$ ) : 轻微的平滑效果，较好地保留图像的大部分细节特征和边缘锐度；
- **大尺寸模板** (如  $13 \times 13$ ) : 图像变得非常模糊，大量的纹理细节和边缘信息丢失。

### 2. 噪声图像

- **去噪能力与尺寸成正比**: 增大高斯卷积模板的尺寸能够更有效地抑制和去除噪声；
- **小尺寸模板** (如  $5 \times 5$ ) : 对噪声的削减作用有限，只能轻微减少噪声干扰；
- **大尺寸模板** (如  $13 \times 13$ ) : 显著平滑掉噪声，但同时也导致图像原本的边缘和细节被模糊化。

## 题目二：阈值分割

### 算法描述

本实验采用 Otsu's 作为阈值分割方法。

Otsu's 方法通过最大化前景和背景类之间的方差来自动确定最佳的全局阈值。其优化的目标是寻找一个特定的阈值  $T$ ，使得分割后的前景和背景的类间方差 达到最大。

类间方差的计算公式如下：

$$\sigma_B^2(T) = \omega_0(T)\omega_1(T)(\mu_0(T) - \mu_1(T))^2$$

其中：

- $\omega_0, \omega_1$ : 分别代表前景和背景像素点占总像素点的比例；
- $\mu_0, \mu_1$ : 分别代表前景和背景像素点的平均灰度值。

Otsu's 阈值分割的具体实现步骤如下：

- 输入：灰度图像  $I$ 。
- 输出：二值化图像  $I_{binary}$ 。

1. **计算直方图**: 统计图像  $I$  的灰度直方图  $H$ ，以获取每个灰度级的像素分布情况；

2. **遍历阈值**: 对每一个可能的阈值  $T$  进行循环计算。

3. **像素分组**: 根据当前阈值  $T$ ，将图像像素分为两组：

- $C_0$ : 灰度值小于  $T$  的像素集合 ( $I(p) < T$ )；

- $C_1$ : 灰度值大于等于  $T$  的像素集合 ( $I(p) \geq T$ )。

#### 4. 计算统计量:

- 计算  $C_0$  组的像素比例  $\omega_0(T)$  和平均灰度  $\mu_0(T)$ ;
- 计算  $C_1$  组的像素比例  $\omega_1(T)$  和平均灰度  $\mu_1(T)$ 。

#### 5. 计算类间方差: 代入公式计算当前的类间方差。

$$\sigma_B^2(T) = \omega_0(T)\omega_1(T)(\mu_0(T) - \mu_1(T))^2$$

#### 6. 确定最佳阈值: 比较所有 $T$ 对应的方差值, 找到使 $\sigma_B^2(T)$ 最大的那个阈值, 记为 $T^*$ 。

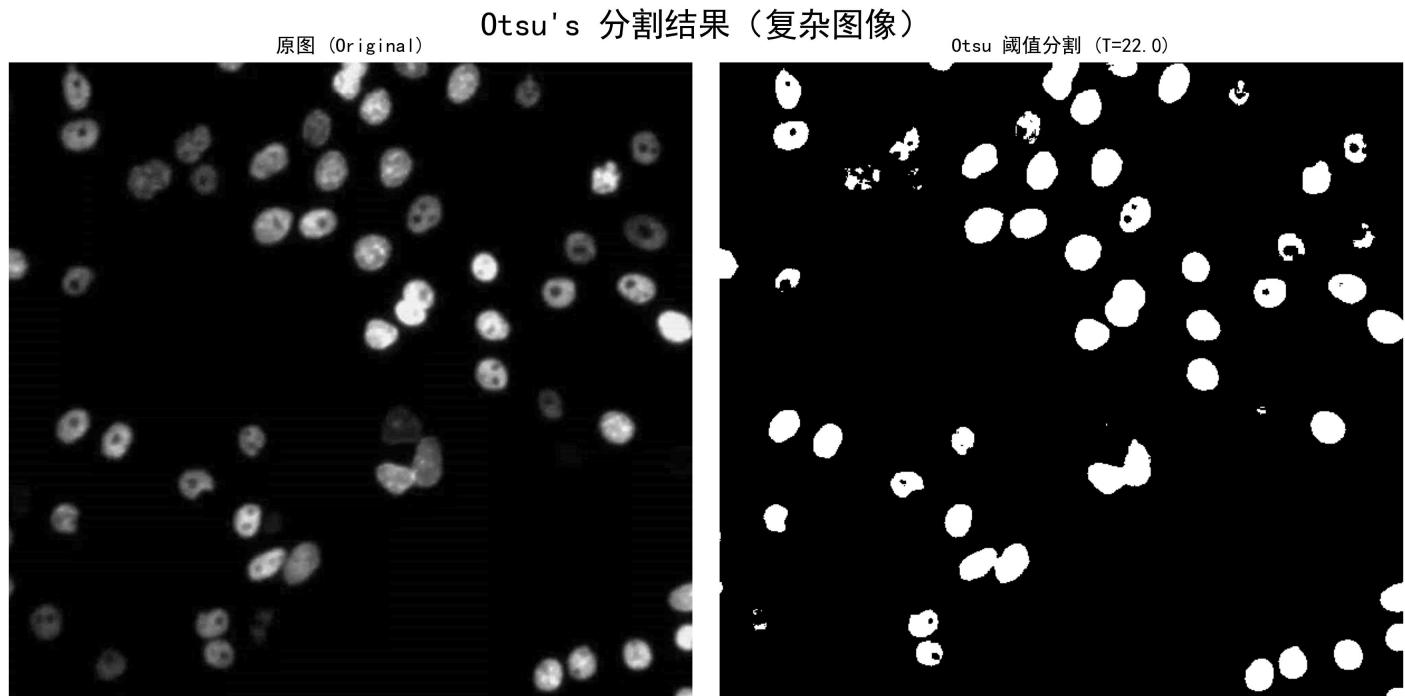
#### 7. 二值化处理: 根据最佳阈值 $T^*$ 对原图像 $I$ 进行二值化操作:

- 如果  $I(p) \geq T^*$ , 则  $I_{binary}(p) = 255$ ;
- 否则  $I_{binary}(p) = 0$ 。

#### 8. 输出: 返回最终的二值图像 $I_{binary}$ 。

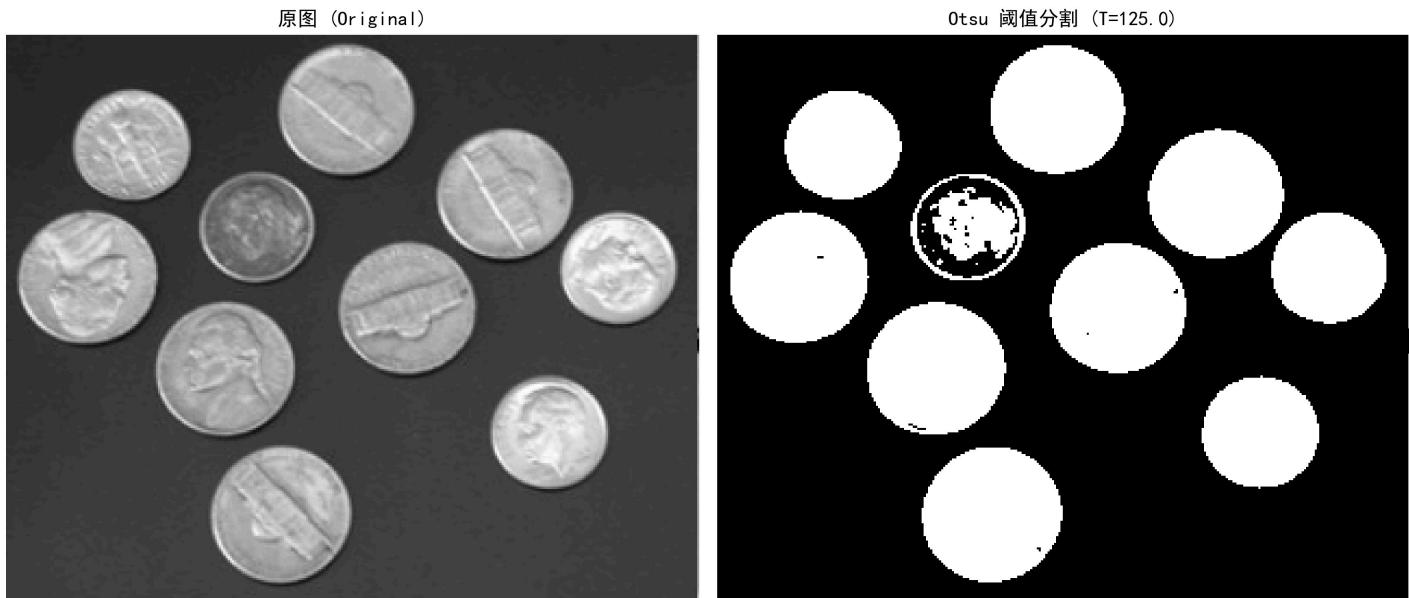
## 实验结果

对复杂图像进行 Otsu's 阈值分割, 结果如下图所示。



对简单图像进行 Otsu's 阈值分割, 结果如下图所示。

## Otsu's 分割结果 (简单图像)



## 现象规律

### 1. 复杂图像

- 当图像存在背景复杂、光照分布不均、包含多种纹理特征，或者前景与背景的灰度值出现重叠时，单纯依赖全局阈值进行分割往往难以获得理想结果；
- 过分割**：将原本完整的单一目标错误地切割成多个碎片；
- 欠分割**：未能将多个独立对象分开，或将目标与背景错误地合并。

### 2. 简单图像

- 对于前景与背景灰度差异显著、光照条件均匀且结构简单的图像，阈值分割法表现优异；
- Otsu 算法能够自动且精准地计算出最佳阈值，有效地实现前景与背景的分离，最终获得边缘清晰、形态完整的目标区域。

## 题目三：边缘检测

### 算法描述

本实验测试多种边缘检测算子，包括一阶微分算子 (Roberts, Prewitt, Sobel) 、二阶微分算子 (Laplacian, LoG) 以及多阶段检测算法 (Canny) 。

### 一阶微分算子 (Roberts, Prewitt, Sobel)

#### 1. Roberts 算子

- 原理**：Roberts 算子基于交叉差分，利用局部差分算子寻找边缘。
- 卷积核**：采用  $2 \times 2$  的离散差分算子，分别计算对角线方向的梯度。

$$G_x = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

- 梯度幅值**： $M = \sqrt{G_x^2 + G_y^2}$

#### 2. Prewitt 算子

- 原理**：Prewitt 算子基于  $3 \times 3$  模板通过对邻域像素进行加权平均来计算水平和垂直方向的梯度，具有一定的噪声平滑能力。
- 卷积核**：

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$$

- 梯度幅值**： $M = \sqrt{G_x^2 + G_y^2}$

#### 3. Sobel 算子

- **原理**: Sobel 算子基于  $3 \times 3$  模板对中心像素赋予高权重，从而能够提供更好的平滑效果，对噪声有更好的抑制作用。
- **卷积核**:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

- **梯度幅值**:  $M = \sqrt{G_x^2 + G_y^2}$

## 二阶微分算子 (Laplacian, LoG)

### 4. Laplacian 算子

- **原理**: Laplacian 是二阶导数算子，通过检测图像灰度变化的零交叉点来定位边缘，对细线和孤立点敏感。
- **数学定义**:  $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
- **卷积核**:

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{或} \quad L = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

### 5. LoG 算子

- **原理**: LoG 算子先对图像进行高斯平滑，然后应用 Laplacian 算子。
- **数学表达**:  $LoG(x, y) = \nabla^2(G(x, y, \sigma) * I(x, y))$ ，其中  $G(x, y, \sigma)$  是高斯函数，\* 表示卷积运算。

## 多阶段检测算法 (Canny)

Canny 边缘检测算法的具体实现步骤如下：

### 1. 高斯平滑:

- 使用高斯滤波器对原始图像进行平滑处理，去除噪声干扰；
- $I_{smooth} \leftarrow GaussianBlur(I, \sigma)$ 。

### 2. 计算梯度幅值和方向:

- 利用 Sobel 等算子计算每个像素点的水平梯度  $G_x$  和垂直梯度  $G_y$ ；
- 计算幅值  $M = \sqrt{G_x^2 + G_y^2}$  和方向  $\theta = \text{atan2}(G_y, G_x)$ 。

### 3. 非极大值抑制:

- 将边缘“细化”为单像素宽。
- 沿梯度方向  $\theta$  检查当前像素的梯度强度。
  - 如果当前像素  $M(r, c)$  是其梯度方向上两个邻居中的局部最大值，则保留；
  - 否则将其抑制。

### 4. 滞后双阈值:

- 使用两个阈值：高阈值  $T_H$  和低阈值  $T_L$ 。
- **强边缘** ( $E_{strong}$ ): 梯度值  $\geq T_H$  的像素，被确定为边缘。
- **弱边缘** ( $E_{weak}$ ): 梯度值在  $T_L$  和  $T_H$  之间的像素。
- **边缘连接**: 对于弱边缘像素，
  - 如果它与任何强边缘像素相连，则将其保留为边缘；
  - 否则将其丢弃。

## 实验结果

### 一阶微分算子 (Roberts, Prewitt, Sobel)

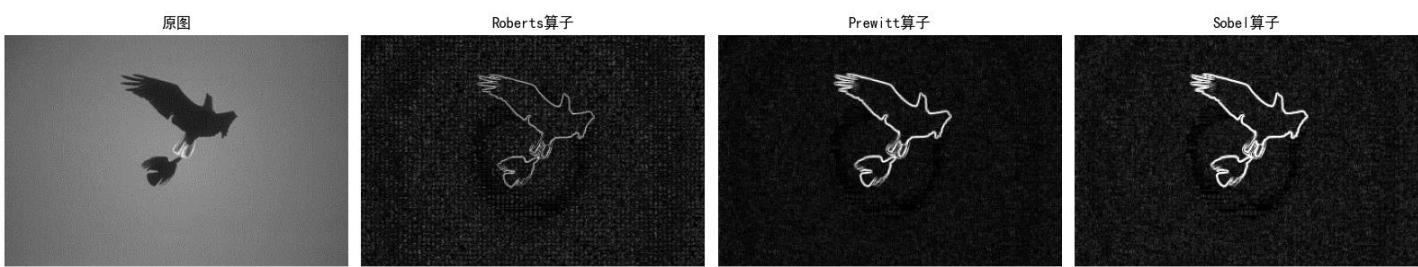
对复杂图像应用一阶微分算子，结果如下图所示。

一阶微分算子对比分析（复杂图像）



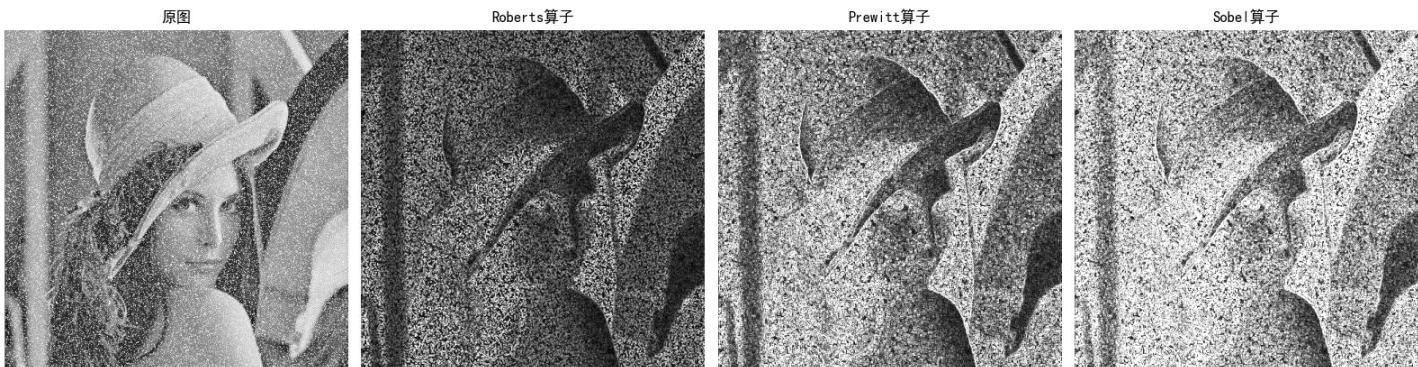
对简单图像应用一阶微分算子，结果如下图所示。

一阶微分算子对比分析（简单图像）



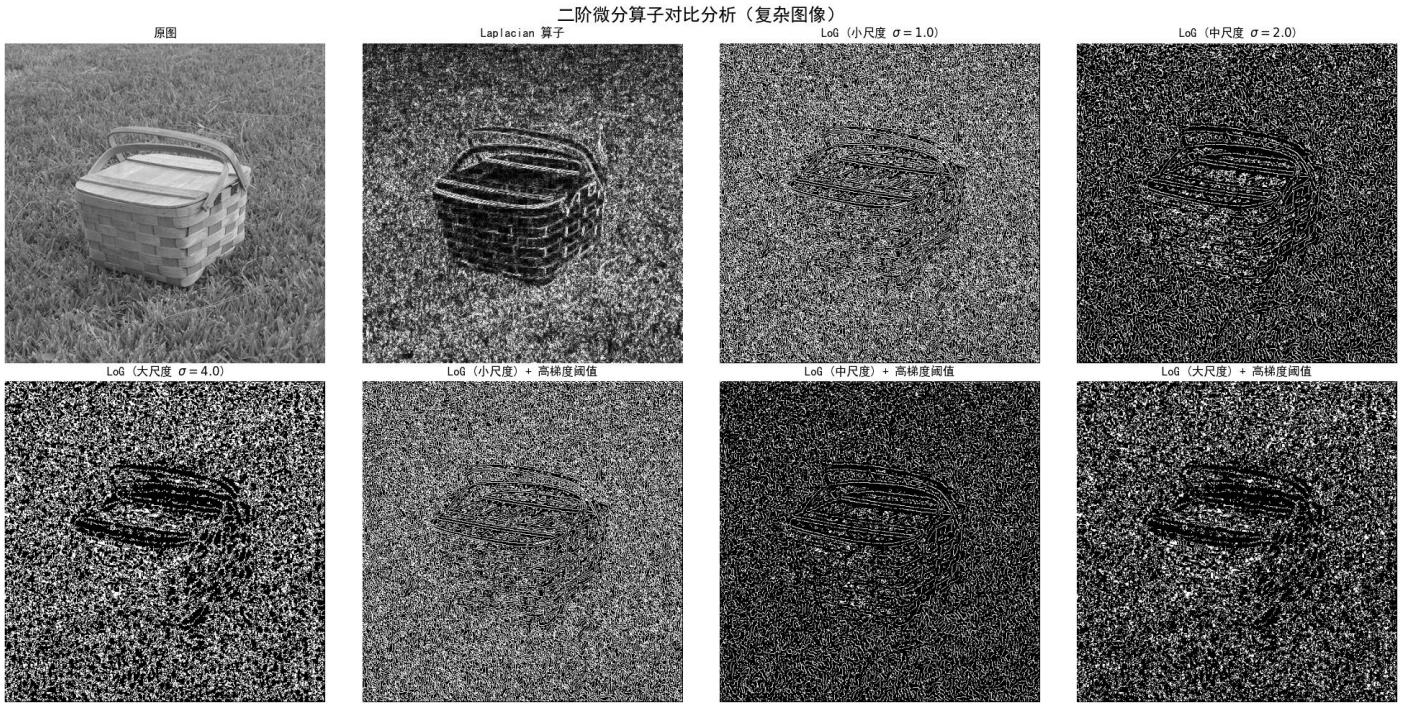
对噪声图像应用一阶微分算子，结果如下图所示。

一阶微分算子对比分析（噪声图像）

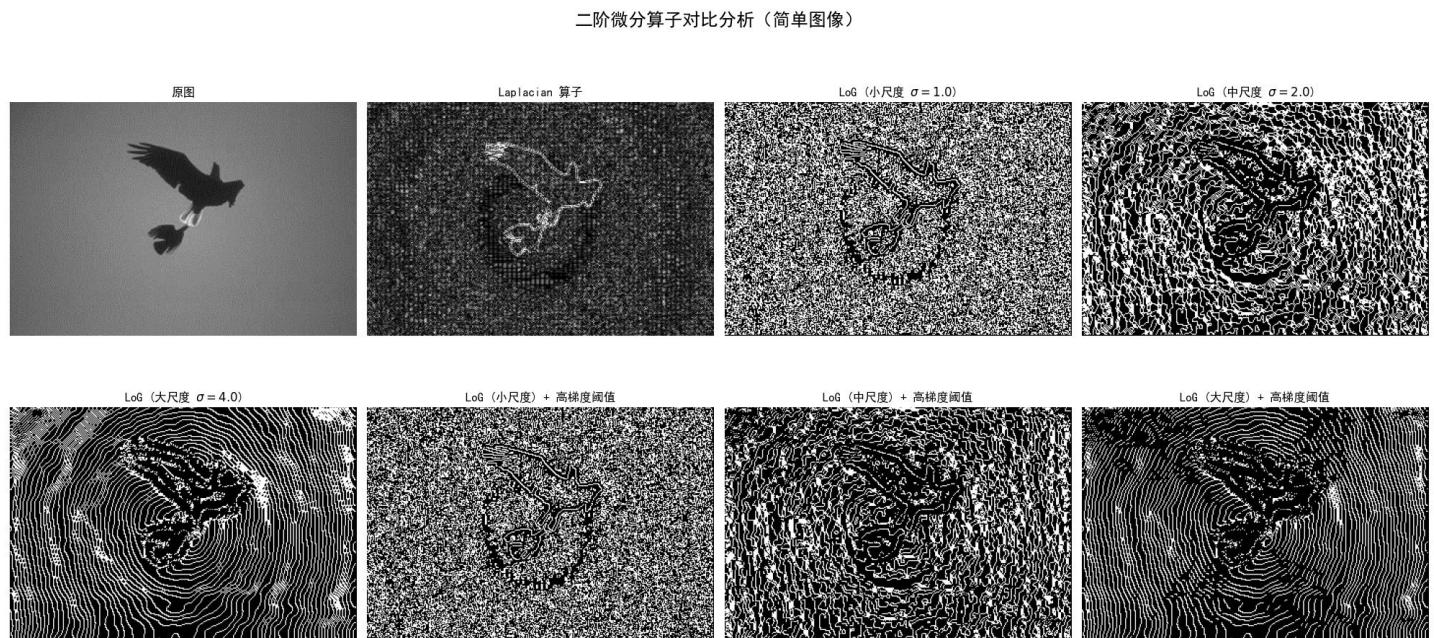


## 二阶微分算子 (Laplacian, LoG)

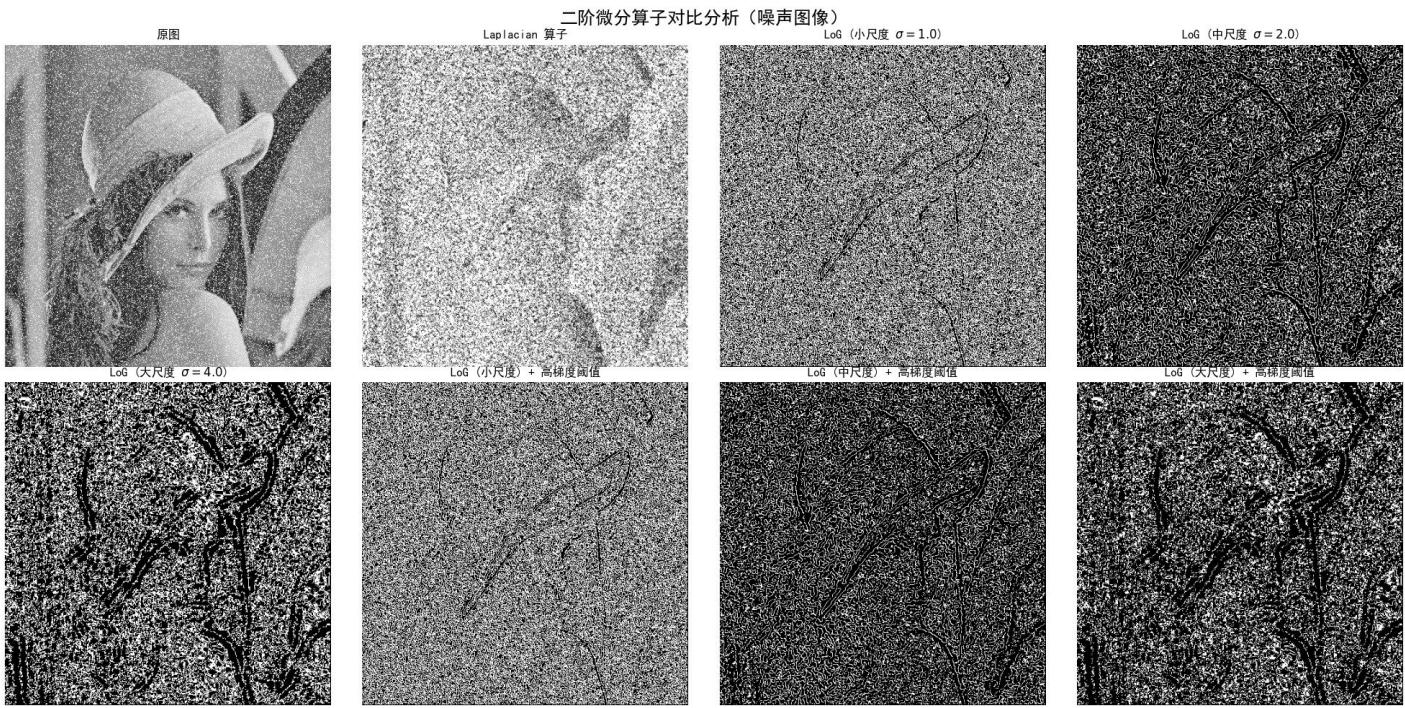
对复杂图像应用二阶微分算子以及高梯度阈值，结果如下图所示。



对简单图像应用二阶微分算子以及高梯度阈值，结果如下图所示。



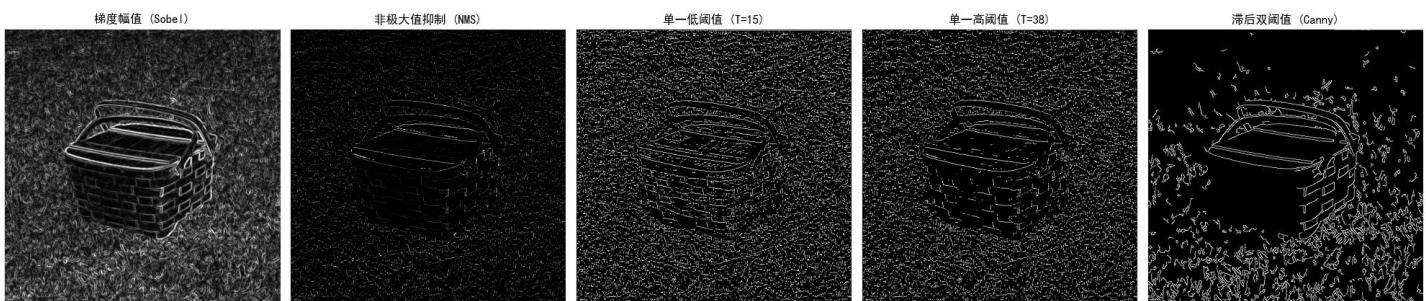
对噪声图像应用二阶微分算子以及高梯度阈值，结果如下图所示。



## 多阶段检测算法 (Canny)

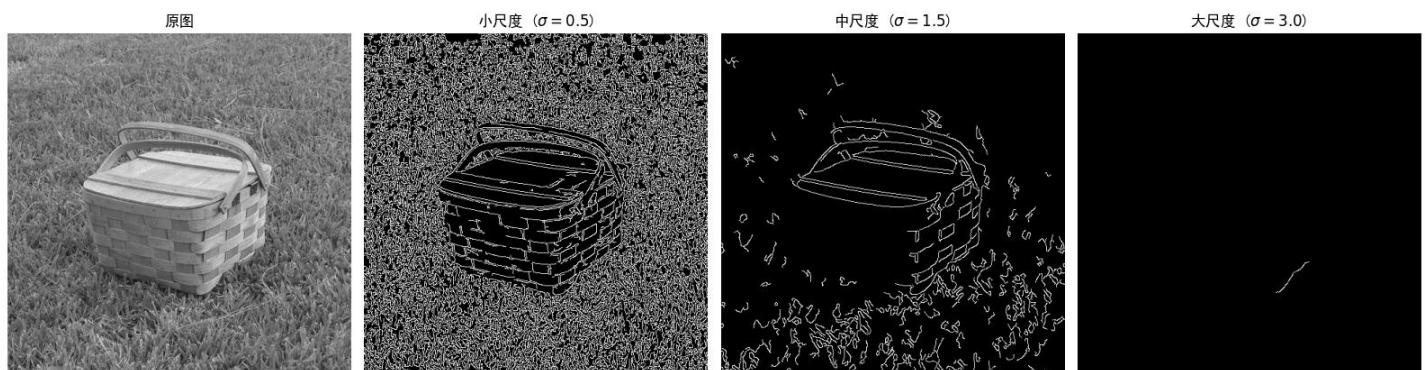
对复杂图像应用 NMS 实现与单双阈值的 Canny 算子，结果如下图所示。

Canny 算法NMS实现与单双阈值分析（复杂图像）



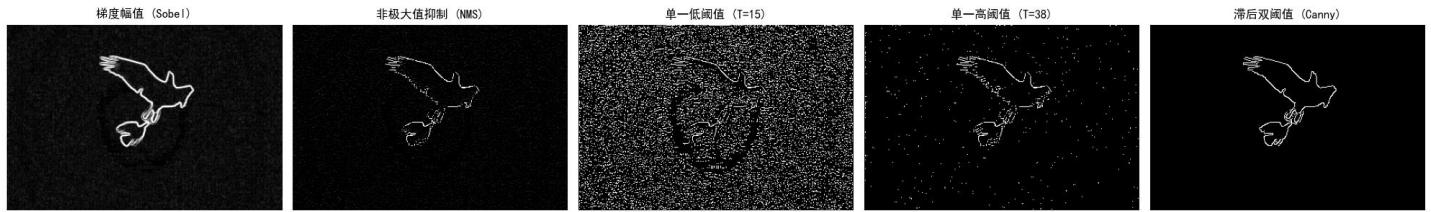
对复杂图像应用多尺度的 Canny 算子，结果如下图所示。

Canny 算法多尺度分析（复杂图像）



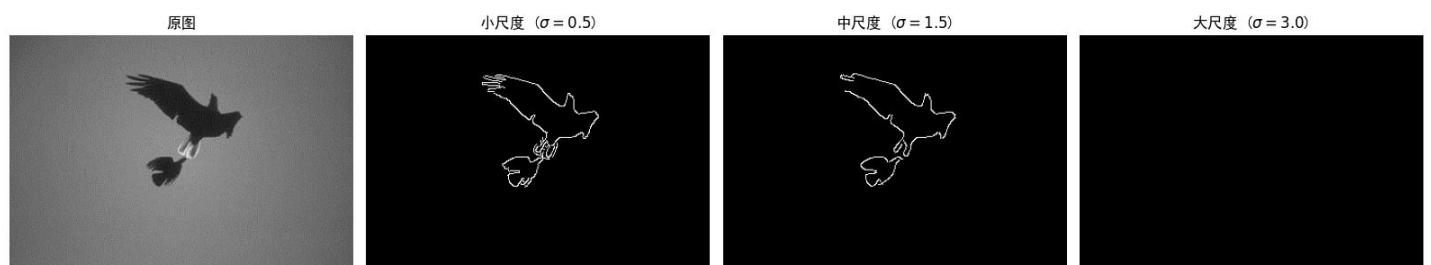
对简单图像应用 NMS 实现与单双阈值的 Canny 算子，结果如下图所示。

Canny 算法NMS实现与单双阈值分析（简单图像）



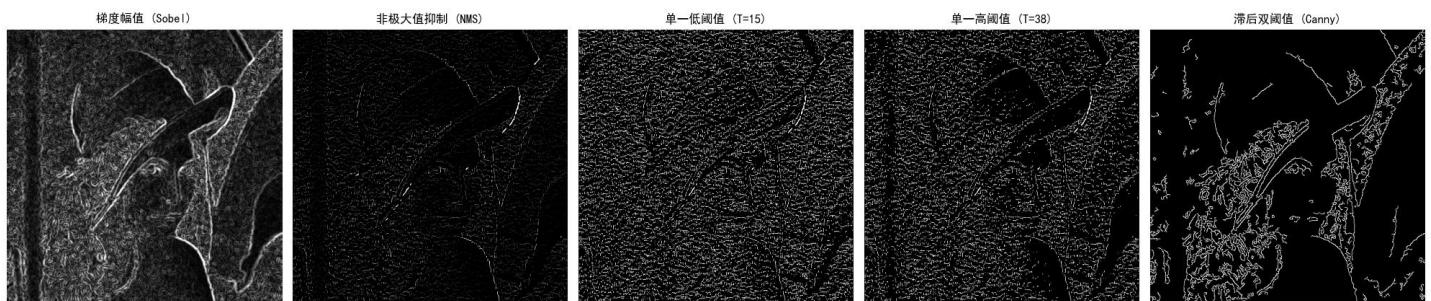
对简单图像应用多尺度的 Canny 算子，结果如下图所示。

Canny 算法多尺度分析（简单图像）



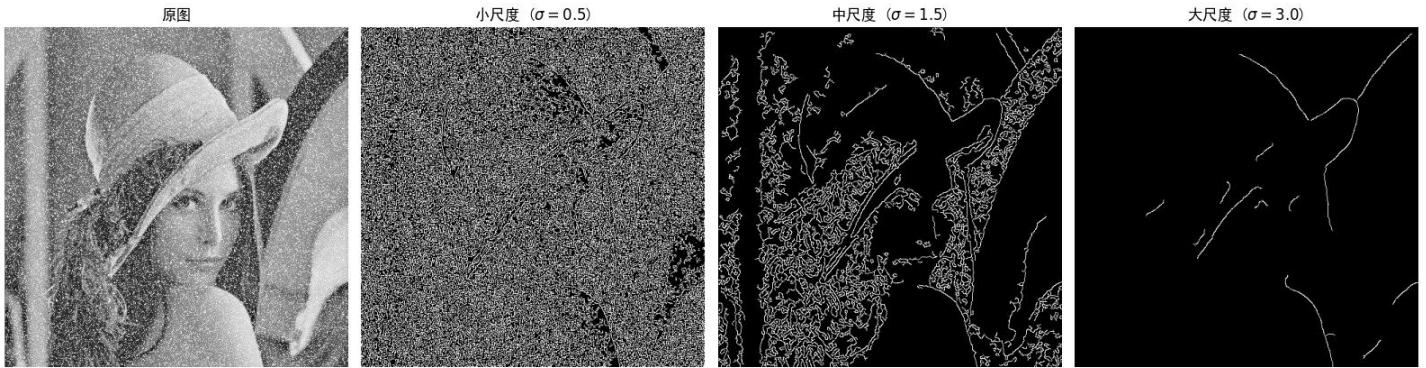
对噪声图像应用 NMS 实现与单双阈值的 Canny 算子，结果如下图所示。

Canny 算法NMS实现与单双阈值分析（噪声图像）



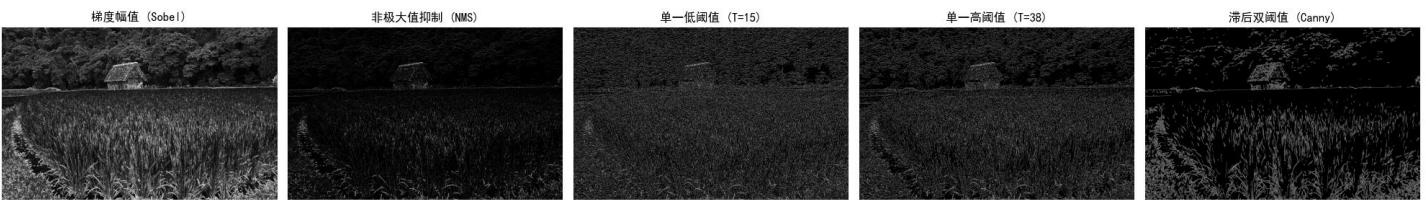
对噪声图像应用多尺度的 Canny 算子，结果如下图所示。

### Canny 算法多尺度分析（噪声图像）



对多尺度图像应用 NMS 实现与单双阈值的 Canny 算子，结果如下图所示。

### Canny 算法NMS实现与单双阈值分析（多尺度图像）



## 现象规律

### 一阶微分算子 (Roberts, Prewitt, Sobel)

#### 1. 厚边界现象：

- **观察：**一阶微分算子检测出的边缘通常表现为多像素宽的“粗线条”，而非理想的单像素细线；
- **原因：**一阶导数在图像的灰度渐变区域（边缘过渡区），导数值会在多个连续像素点上保持较高水平，导致检测出的边缘跨越多个像素宽度。

#### 2. 噪声敏感性：

- **Roberts：**对噪声最敏感。仅使用  $2 \times 2$  的微小核进行差分，缺乏平滑机制，极易将噪声点误判为边缘；
- **Prewitt & Sobel：**对噪声有一定抑制作用。Sobel 算子效果最佳，引入高斯加权平滑，在检测边缘前对邻域进行了微小的模糊处理。

#### 3. 阈值与连续性：

- **低阈值：**检测到丰富的细节和微弱边缘，同时引入大量噪声产生的伪边缘，导致边缘杂乱；
- **高阈值：**能有效滤除噪声，但会导致真实边缘断裂（不连续、不封闭），丢失重要结构信息。

### 二阶微分算子 (Laplacian, LoG)

#### 1. Laplacian 算子

- **全响应特性：**Laplacian 作为二阶导数算子，对图像中**任何**灰度变化（无论是强边缘、弱纹理还是噪声）都会产生响应。实验中观察到它不仅响应边缘，还对非边界区域的细节产生强烈反应；
- **极度敏感：**由于二阶微分会放大高频信号，Laplacian 对噪声极其敏感。在含噪图像中，它会产生密集的伪边缘，导致结果杂乱无章，因此不适合单独使用。

#### 2. 高斯-Laplacian (LoG) 算子

- **噪声抑制与细节保留：**通过引入高斯平滑，LoG 有效解决了 Laplacian 的噪声敏感问题；
- **多尺度 ( $\sigma$ ) 特性：**

- 小尺度(小 $\sigma$ ): 平滑弱, 能检测到细微的灰度变化和丰富纹理, 但抗噪能力差;
- 大尺度(大 $\sigma$ ): 平滑强, 能忽略纹理和噪声, 只保留显著的结构轮廓;
- 边缘偏移: 随着尺度 $\sigma$ 的增大, 高斯平滑会导致边缘位置(特别是角点和曲率大的地方)发生偏移, 不再位于真实的物理边界上;
- 双像素宽边缘: 由于LoG检测的是零交叉点, 直接取绝对值显示时, 边缘表现为“双像素宽”的双线结构;
- 单像素细化: 实验证明, 通过检测零交叉点并辅以高梯度阈值约束, 可以消除伪边缘, 并获得定位精准的单像素宽边缘。

## 多阶段检测算法(Canny)

Canny算子结合了上述方法的优点, 展现出最优的综合性能。

- **细化边界(非极大值抑制NMS):**
  - 效果: NMS算法成功消除了梯度算子的“厚边界”现象。它通过沿梯度方向寻找局部最大值, 将宽阔的边缘响应细化为精准的**单像素宽**线条。
- **滞后双阈值:**
  - **单一阈值的缺陷:** 低阈值导致伪边缘, 高阈值导致边缘断裂。
  - **双阈值的优势:** 利用高阈值确定“强边缘”, 利用低阈值寻找“弱边缘”并进行连接。这种策略完美解决了边缘断裂问题, 在保证边缘连续性的同时, 有效抑制了独立噪声点。
- **多尺度检测:** 与LoG类似, 调整Canny内部的高斯滤波器尺度 $\sigma$ 可以控制检测的细节程度。大尺度用于提取主轮廓, 小尺度用于提取精细纹理。

# 题目四

## 算法描述

### 区域生长

区域生长的具体实现步骤如下:

1. **初始化:**
  - 创建一个与原图同大小的全零图像 $I_{segmented}$ 用于存储分割结果。
  - 初始化一个空队列 $Q$ , 将所有选定的种子点 $S$ 入队。
  - 将种子点位置标记为已访问。
2. **迭代生长:**
  - 当队列 $Q$ 不为空时, 循环执行以下操作:
    - 从 $Q$ 中取出一个像素点 $P(x, y)$ 。
    - 检查 $P$ 的所有未访问邻域像素 $N(x', y')$ 。
    - **相似性判定:** 计算 $N$ 的灰度值 $I(N)$ 与种子点灰度 $I(S)$ (或当前区域的平均灰度)之差的绝对值。如果该差值小于预设阈值 $T$ , 即 $|I(N) - I(S)| \leq T$ , 则认为 $N$ 属于该区域。
    - **更新:** 将满足条件的 $N$ 加入队列 $Q$ , 并在 $I_{segmented}$ 中标记为已访问。
3. **结束:** 当队列清空时算法结束, 输出分割后的二值图像 $I_{segmented}$ 。

### 区域分裂与合并

区域分裂与合并算法的具体实现步骤如下:

1. **阶段一: 区域分裂(Splitting)**
  - 策略: 从整幅图像开始, 检查当前区域 $R$ 是否满足**均匀性准则**。
  - 递归:
    - 如果 $R$ 满足均匀性, 则停止分裂, 该节点成为叶子节点。
    - 如果 $R$ 不满足均匀性, 则将其均分为四个子象限区域 $R_1, R_2, R_3, R_4$ , 并对每个子区域递归执行上述检查。
2. **阶段二: 区域合并(Merging)**
  - 策略: 分裂完成后, 图像被破碎成许多小的均匀块。算法随后遍历这些区域, 检查相邻的区域是否满足**相似性准则**。
  - 操作: 如果满足条件, 则将 $R_A$ 和 $R_B$ 合并为一个新的大区域 $R_{AB}$ , 并更新区域邻接图, 直到无法再进行任何合并为止。

## 实验结果

### 区域生长

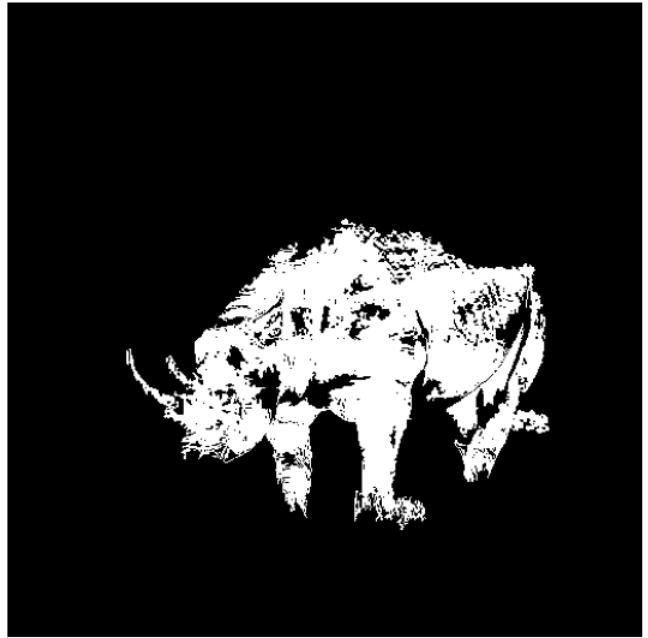
对复杂图像应用区域生长算法, 往往需要选择多个种子点, 结果如下图所示。

### 区域生长算法实现（简单图像）

原始图像



区域生长结果（阈值 T=15）



对简单图像应用区域生长算法，结果如下图所示。

### 区域生长算法实现（简单图像）

原始图像



区域生长结果（阈值 T=30）



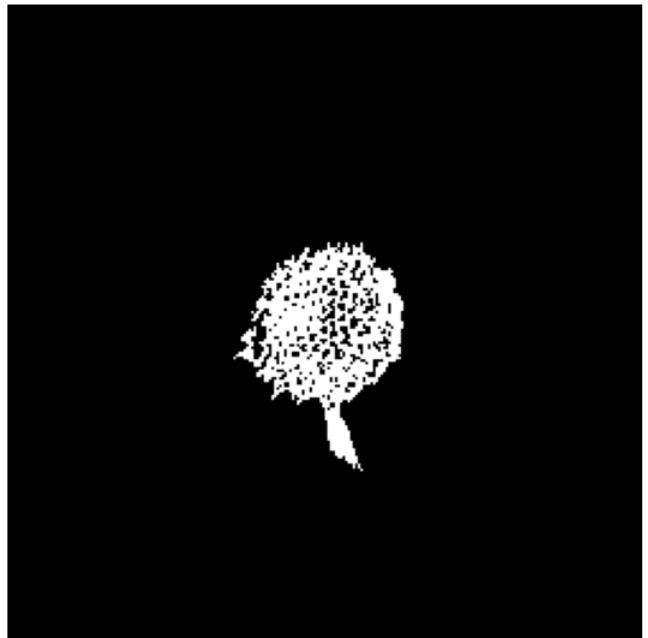
对于多区域的简单图像，往往需要选择多个种子点，结果如下图所示。

区域生长算法实现（简单图像）

原始图像



区域生长结果（阈值 T=30）



区域生长算法实现（简单图像）

原始图像



区域生长结果（阈值 T=30）



## 区域分裂与合并

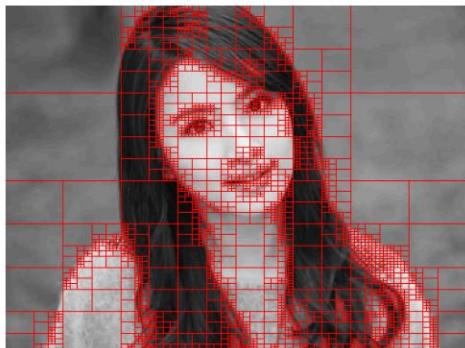
对复杂图像应用区域分裂与合并算法，结果如下图所示。

区域分裂与合并算法实现（复杂图像）

原图



分裂过程

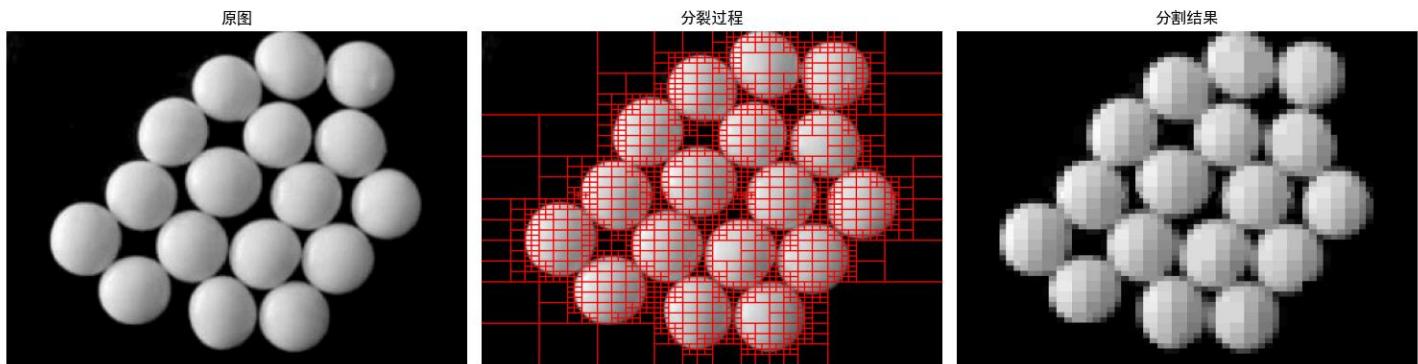


分割结果



对简单图像应用区域分裂与合并算法，结果如下图所示。

区域分裂与合并算法实现（简单图像）



## 现象规律

### 区域生长

- **简单图像**: 对于前景与背景对比度高、且内部灰度均匀的简单图像，算法表现优异。
  - 只需选取一个典型种子点并设定合理阈值，即可获得边缘平滑、完整且单连通的分割结果。
- **复杂图像**: 在纹理不均或存在噪声的图像中，算法对**种子点**和**阈值**的选择极度敏感。
  - 阈值过小会导致生长在真实边界前停止；
  - 阈值过大则会导致生长“溢出”到背景区域。
- **对光照与阴影的敏感性**: 光照与阴影会导致同一物体的灰度值发生显著变化，导致生长过程在阴影边界处错误截断，破坏了分割的整体性。
- **多区域图像的处理局限**: 区域生长算法一次只能提取一个连通区域。
  - 对于包含多个互不相连目标的图像，单次种子点选择无法覆盖全局；
  - 必须通过选取不同的种子点，才能将所有独立的目标区域逐一分割出来。

### 区域分裂与合并

- **复杂图像的表现**:
  - **分裂阶段**: 由于灰度变化剧烈，图像会被过度分割成大量细小的子块，初始结果呈现明显的“马赛克”效应。
  - **合并阶段**: 若合并准则不当，极易导致碎片无法正确聚合，最终结果往往显得支离破碎，难以获得具有良好语义的分割区域。
- **简单图像的表现**:
  - **分裂阶段**: 算法能快速在灰度均匀的区域停止分裂，形成较大的子块，仅在边缘处进行细分。
  - **合并阶段**: 相似的相邻子块容易满足合并条件，能够有效聚合成完整的对象。最终分割结果通常边界清晰，能够很好地分离主体与背景。