数字图像处理
(Digital Image Processing)

核心知识点精炼笔记 8-9

东南大学计算机科学与工程学院
09023419 严伟琛

授课教师：薛澄
参考教材：拉斐尔 · 冈萨雷斯 (Rafael C. Gonzalez) 第四版

2026 年 1 月 15 日

目录

8 图像压缩 (Image Compression)

8.1 数据冗余类型

在图像压缩中，若 n_1 代表原始图像的数据量，n_2 代表压缩后的数据量，则相关定义如下：

- **压缩比 (Compression Ratio)**：$C = n_1/n_2$。
- **相对数据冗余 (Relative Data Redundancy)**：$R = 1 - \frac{1}{C}$。

图像压缩的核心在于消除以下三种基本冗余：

8.1.1 编码冗余 (Coding Redundancy)

编码冗余发生在为灰度级分配码字时，没有考虑到灰度级出现的统计概率。

- **平均码长 (Average Length)**：设随机变量 r_k 表示灰度级，$p(r_k)$ 为其出现的概率，$l(r_k)$ 为分配给该灰度级的比特数，则平均码长为：

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k)p(r_k)$$

- **冗余判据**：若 L_{avg} 大于图像的熵（Entropy），则存在编码冗余。
- **消除方法**：为频繁出现的灰度级分配较短的码字，为不常出现的灰度级分配较长的码字（如 Huffman 编码）。

8.1.2 空间冗余和时间冗余 (Spatial and Temporal Redundancy)

空间冗余（针对静止图像）和**时间冗余**（针对视频序列）反映了像素之间的相关性。

- **原理**：由于相邻像素在空间上通常高度相关（灰度值相近），直接存储每个像素的绝对值会导致大量重复信息的记录。
- **消除方法**：
  - **映射 (Mapping)**：将像素值转换为另一种形式（如差分值），使数据分布更加集中。
  - **典型技术**：运行长度编码 (RLE)、预测编码 (DPCM) 等。

8.1.3 心理视觉冗余 (Psychovisual Redundancy)

心理视觉冗余与人类视觉系统 (HVS) 对不同视觉信息的敏感度不同有关。

- **定义**：人类大脑并不对图像中的所有比特信息进行等权处理。人眼对某些信息的忽略使得这部分数据在视觉表现上是多余的。
- **关键特征**：
  - 消除心理视觉冗余必然导致**信息丢失**，因此这种压缩是**不可逆**的（有损压缩）。
  - 这种方法利用了人眼对高频细节（如纹理）或某些颜色变化的不敏感性。
- **消除方法**：量化 (Quantization)。通过减少灰度级的精细程度，在视觉质量可接受的情况下显著降低数据量。

8.1.4 三种数据冗余的综合对比与总结

为了更清晰地理解三种冗余的区别，下表从处理目标、数学本质、处理手段及可恢复性四个维度进行了对比总结：

表 1: 编码冗余、空间/时间冗余与心理视觉冗余对比

维度	编码冗余 (Coding)	空间与时间冗余 (Spatial/Temporal)	心理视觉冗余 (Psychovisual)
处理目标	消除灰度级符号表示的冗余。	消除像素间的相关性（重复信息）。	消除人类视觉系统不敏感的信息。
数学本质	$L_{avg} > H$（平均码长大于熵）。	像素值可以由相邻像素预测。	图像中包含人眼感知不到的比特。
典型处理方法	**变长编码**：如 Huffman 编码、算术编码。	**映射与变换**：如预测编码 (DPCM)、变换编码 (DCT)。	**量化 (Quantization)**：减少灰度等级或频率分量精度。
可恢复性	**无损 (Lossless)**：解压后可完全恢复原始像素值。	**通常无损**：若映射过程可逆，则可完全恢复。	**有损 (Lossy)**：信息一旦去除不可恢复，图像质量有降级。

异同点核心深度解析：

1. **共同点**：三者的最终目的都是为了降低表示图像所需的总比特数，提高压缩比。
2. **不同点 - 理论极限**：
   - **编码冗余**和**空间冗余**的消除通常不涉及图像质量的损失，其压缩极限受图像本身的统计特征（熵）限制。
   - **心理视觉冗余**的消除则是人为地丢弃“次要”数据，它突破了熵的限制，是实现高压缩比（如 10:1 以上）的关键手段。
3. **处理逻辑的先后**：
   - 在典型的压缩模型中，通常先通过**映射器**消除空间冗余。
   - 接着通过**量化器**消除心理视觉冗余。
   - 最后通过**符号编码器**消除编码冗余。

8.1.5 信息与信源描述 (Information and Source Description)

在图像压缩中，我们需要一种数学方法来度量信息。通常将图像的灰度级看作是一个离散信源的输出。

- **自信息 (Self-information)**：设一个灰度级 r_k 出现的概率为 $p(r_k)$，则该灰度级所包含的信息量定义为：

$$I(r_k) = -\log p(r_k)$$

当对数底数为 2 时，信息的单位为比特 (bits)。这表明：出现概率越小的事件，包含的信息量越大。

- **信息熵 (Entropy)**：信源输出的平均信息量称为熵。对于一幅具有 L 个灰度级的图像，其熵定义为：

$$H = -\sum_{k=0}^{L-1} p(r_k) \log_2 p(r_k)$$

物理意义：熵 H 给出了对图像灰度级进行无损编码时，每个像素所需的最小平均比特数的理论极限。

- **信源描述**：
  - **零阶信源**：假设像素间相互独立，仅由灰度直方图决定熵（即一阶熵）。
  - **阶数扩展**：若考虑像素间的相关性（如相邻像素），则可以使用二阶或高阶熵。通常像素间相关性越强，条件熵越小，压缩潜力越大。

8.1.6 保真度准则 (Fidelity Criteria)

保真度准则用于衡量压缩后的图像与原始图像之间的接近程度，特别是在有损压缩中，它是评价算法性能的关键指标。主要分为两类：

1. **客观保真度准则 (Objective Fidelity Criteria)**：通过数学公式计算两幅图像之间的差异。
   - **均方根误差 (RMS Error)**：

$$e_{rms} = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [\hat{f}(x,y) - f(x,y)]^2}$$

   其中 f 为原图，$\hat{f}$ 为解压后的图像。
   - **峰值信噪比 (PSNR)**：

$$PSNR = 10 \log_{10} \frac{L^2}{e_{rms}^2}$$

   PSNR 值越高，表示图像失真越小。
2. **主观保真度准则 (Subjective Fidelity Criteria)**：由于客观准则（如 RMS）并不总是能完美反映人眼的视觉感受，因此需要观察者对图像质量进行评价。
   - **方法**：通常由一组观察者根据预设的评分标准（如：优秀、良好、差等）对图像进行打分，最后计算平均评价得分 (MOS)。
   - **必要性**：在涉及心理视觉冗余的消除时，主观准则是最终的衡量标准。

8.2 图像压缩模型

图像压缩系统通常由两个截然不同的逻辑块组成：**编码器 (Encoder)** 和 **解码器 (Decoder)**。由两者组成的整体被称为编解码器（Codec）。

8.2.1 信源编码器与解码器 (Source Encoder and Decoder)

信源编码器旨在通过消除图像中的冗余来减少输入图像的数据量。它主要由以下三个阶段组成：

1. **映射器 (Mapper)**：
   - **功能**：将输入图像变换为一种旨在减少空间冗余（或时间冗余）的格式。
   - **特性**：该步骤通常是可逆的。
   - **典型操作**：运行长度编码（Run-length coding）或离散余弦变换（DCT）等。
2. **量化器 (Quantizer)**：
   - **功能**：根据预设的保真度准则，减少心理视觉冗余。
   - **特性**：这是压缩模型中唯一的**不可逆**阶段。一旦执行，原始信息将无法完全恢复。
   - **注意**：在无损压缩模型中，该模块将被跳过。
3. **符号编码器 (Symbol Encoder)**：
   - **功能**：为量化器的输出分配码字，以消除编码冗余。
   - **典型方法**：使用变长编码，为出现频率高的值分配短码，出现频率低的值分配长码。
   - **特性**：该步骤是完全可逆的。

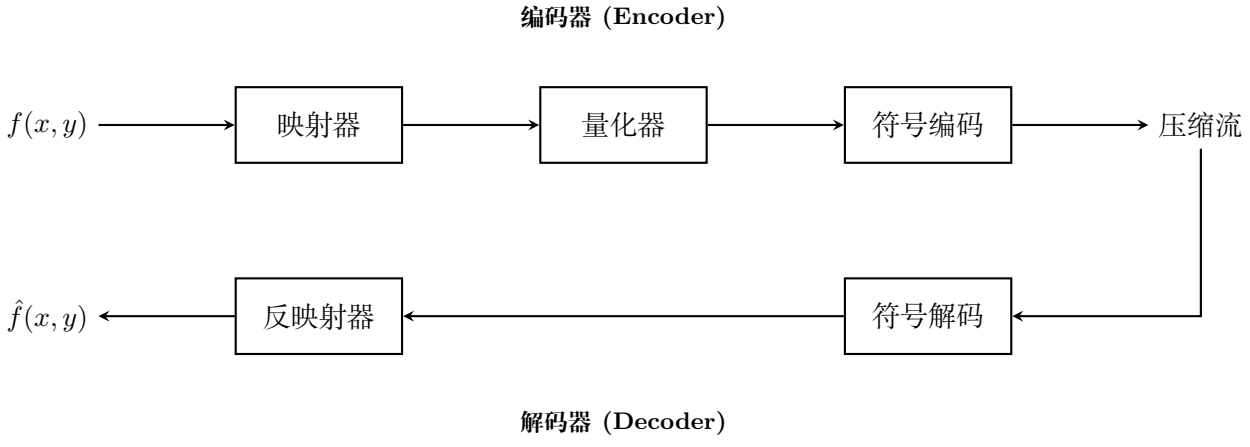
8.2.2 信道编码器与解码器 (Channel Encoder and Decoder)

当压缩后的图像需要通过通信信道传输时，信道编码器起到关键作用。

- **信道编码器**：以受控的方式在压缩数据中加入冗余（如纠错码），以减少信道噪声的影响。
- **信道解码器**：利用加入的冗余检测并纠正传输过程中产生的错误。

8.2.3 编解码器的对称性

信源解码器包含符号解码器和反映射器，它们分别执行与编码器相反的操作。由于量化过程是不可逆的，因此解码器中不存在“反量化器”，而是使用一个根据量化规则重建灰度级的模块。



8.3 一些基本的压缩方法

图像压缩算法根据其是否能完全恢复原始数据，主要分为两大类：

- **无损压缩 (Lossless Compression)**：能够从压缩数据中精确还原原始图像，不丢失任何信息。适用于医学图像、法律文档或科学研究。
- **有损压缩 (Lossy Compression)**：在重建时会丢失部分信息，但能获得远高于无损压缩的压缩比。适用于视频、流媒体及普通数码照片。

8.3.1 基本编码技术概述

在详细讨论具体算法前，以下是图像压缩中常用的几种核心技术及其消除的冗余类型：

1. **消除编码冗余的技术**：
   - **霍夫曼编码 (Huffman Coding)**：根据符号出现概率分配变长码，是最常用的变长编码。
   - **算术编码 (Arithmetic Coding)**：将整个消息序列映射为 $[0, 1)$ 区间的一个实数。相比霍夫曼编码，它在处理极高概率符号时效率更高。
   - **LZW 编码 (Lempel-Ziv-Welch Coding)**：基于字典的编码方法，通过为重复出现的字符串分配固定长度的代码来压缩，广泛用于 GIF 格式。
2. **消除空间冗余的技术**：
   - **行程编码 (Run-Length Coding)**：又称码长编码。通过记录相同像素值连续出现的长度（Run）来压缩，对二值图像（如传真）极其有效。
   - **比特平面编码 (Bit-Plane Coding)**：将多位图像分解为一系列二值平面，分别进行压缩。
   - **基于符号的编码 (Symbol-based Coding)**：将图像表示为预定义符号库中的索引，常用于文档压缩。
   - **预测编码 (Predictive Coding)**：仅编码相邻像素间的差异（残差），使数据分布大幅向 0 集中（如 DPCM）。
3. **消除心理视觉冗余的技术 (有损)**：
   - **块变换编码 (Block Transform Coding)**：将图像分割为小块（如 8×8），通过离散余弦变换 (DCT) 将能量集中到低频系数，并对高频系数进行量化舍弃（JPEG 核心）。
   - **小波编码 (Wavelet Coding)**：在多尺度下对图像进行分解。它不仅消除了空间和心理视觉冗余，还避免了变换编码在低码率下的“块效应”（JPEG2000 核心）。

8.3.2 霍夫曼编码 (Huffman Coding)

霍夫曼编码是消除**编码冗余**最常用的技术之一。它是一种变长编码（Variable-length Coding）算法，在不考虑符号间相关性的前提下，对于给定的信源符号集，霍夫曼编码能实现最短的平均码长。

1. **核心原理**：

- **变长策略**：根据符号出现的概率分配不同长度的码字。出现频率高的符号分配较短的码字，出现频率低的符号分配较长的码字。
- **前缀码特性**：霍夫曼码是一种前缀码（Prefix Code），即任何一个码字都不是另一个码字的前缀。这保证了在解码时无需分隔符即可实现唯一解码（Unique Decodability）。

2. **算法执行步骤**：

- **步骤 1：源归约 (Source Reduction)**
  (a) 将信源符号按概率大小进行降序排列。
  (b) 将概率最小的两个符号合并为一个复合符号，其概率为两者概率之和。
  (c) 将合并后的新符号集重新进行排序。
  (d) 重复上述过程，直到最后只剩下两个符号为止。
- **步骤 2：码字分配 (Code Assignment)**
  (a) 从归约过程的最后一步出发，分别为最后的两个符号分配二进制数字“0”和“1”。
  (b) 沿归约路径向回追溯，在每一次合并的节点处，分别为其分支分配“0”和“1”，直到回到所有原始符号。

3. **性能衡量指标**：

- **平均码长 (Average Word Length)**：

$$L_{avg} = \sum_{k=1}^{L} p(r_k) l(r_k)$$

  其中 $p(r_k)$ 是符号 r_k 出现的概率，$l(r_k)$ 是为其分配的码字长度。
- **编码效率 (Coding Efficiency)**：

$$\eta = \frac{H}{L_{avg}} \times 100\%$$

  其中 H 是信源的熵（Entropy）。霍夫曼编码被证明是最优的单符号编码，其 L_{avg} 满足：$H \le L_{avg} < H + 1$。

4. **方法总结与局限性**：

- **优点**：实现简单，计算效率高，能有效消除编码冗余。
- **局限性**：它针对单个符号进行编码，没有利用像素间的空间相关性；且在处理概率分布极不均匀（如某一概率接近 1）的情况时，其压缩效率不如算术编码。

8.3.3 例题 19：基于 4×8 图像的霍夫曼编码全过程推导

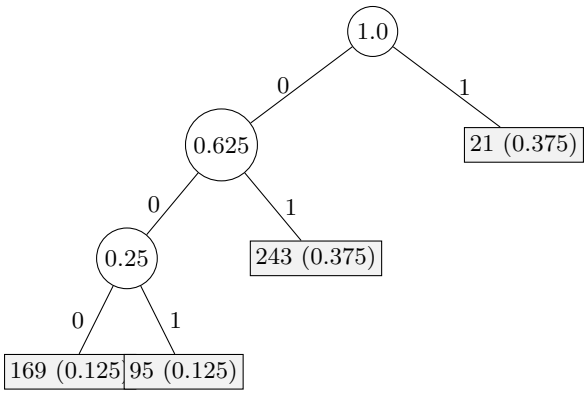
题目：已知一幅 4×8 的 8 比特灰度图像 $\mathbf{f}$ 如下。请：(1) 统计概率；(2) 绘制霍夫曼树并分配码字；(3) 计算该信源的熵 H；(4) 计算平均码长 L_{avg}、压缩比 C 及编码效率 η。

$$\mathbf{f} = \begin{bmatrix} 21 & 21 & 21 & 95 & 169 & 243 & 243 & 243 \\ 21 & 21 & 21 & 95 & 169 & 243 & 243 & 243 \\ 21 & 21 & 21 & 95 & 169 & 243 & 243 & 243 \\ 21 & 21 & 21 & 95 & 169 & 243 & 243 & 243 \end{bmatrix}$$

1. 统计概率分布 图像总像素数 $n = 4 \times 8 = 32$。各灰度级出现的次数 n_k 及概率 p_k 为：

- $r_1 = 21$: 出现 12 次, $p_1 = 12/32 = 0.375$
- $r_2 = 243$: 出现 12 次, $p_2 = 12/32 = 0.375$
- $r_3 = 95$: 出现 4 次, $p_3 = 4/32 = 0.125$
- $r_4 = 169$: 出现 4 次, $p_4 = 4/32 = 0.125$

2. 霍夫曼树与码字分配 通过不断合并概率最小的节点构建最优二叉树：



码字分配结果：

- $r_k = 21 \rightarrow \mathbf{1}$ (长度 1)
- $r_k = 243 \rightarrow \mathbf{01}$ (长度 2)
- $r_k = 95 \rightarrow \mathbf{001}$ (长度 3)
- $r_k = 169 \rightarrow \mathbf{000}$ (长度 3)

3. 性能评价计算

1. **计算信源熵** H：

$$H = -\sum_{k=1}^{4} p_k \log_2 p_k$$

代入数据：

$$H = -[0.375 \log_2 0.375 + 0.375 \log_2 0.375 + 0.125 \log_2 0.125 + 0.125 \log_2 0.125]$$

$$H = -[0.75 \times (\log_2 3 - 3) + 0.25 \times (-3)] \approx -[0.75 \times (1.585 - 3) - 0.75]$$

$$H \approx 1.811 \text{ bits/pixel}$$

2. **计算平均码长** L_{avg}：

$$L_{avg} = \sum p_k l_k = 0.375(1) + 0.375(2) + 0.125(3) + 0.125(3)$$

$$L_{avg} = 0.375 + 0.75 + 0.375 + 0.375 = 1.875 \text{ bits/pixel}$$

3. **计算压缩比** C：原始图像为 8-bit 编码，即 $L_{orig} = 8$。

$$C = \frac{L_{orig}}{L_{avg}} = \frac{8}{1.875} \approx 4.267$$

4. **计算编码效率** η：

$$\eta = \frac{H}{L_{avg}} \times 100\% = \frac{1.811}{1.875} \times 100\% \approx 96.59\%$$

结论：该压缩方案通过消除编码冗余，在无损的前提下将数据量减少了约 76.5% $(1 - 1/4.267)$。

8.3.4 算术编码 (Arithmetic Coding)

算术编码是一种非块前缀编码技术。与霍夫曼编码为每个信源符号分配固定码字不同，**算术编码**将整个输入符号序列映射为实数轴上 $[0, 1)$ 区间内的一个小数。

1. **基本原理**：

   - **整体编码**：它不把符号看作独立的个体，而是把整个消息序列看作一个整体。
   - **区间划分**：初始区间为 $[0, 1)$。随着序列中符号的逐个输入，编码器根据每个符号出现的概率，不断地将当前区间缩小为一个子区间。
   - **数值表示**：序列处理结束后，最终落在一个极小的区间内。该区间内的任何一个实数都可以代表整个原始序列，通常选择二进制表示最短的那个数作为编码结果。

2. **算法执行逻辑**：

   - **初始化**：根据信源符号的概率分布，将 $[0, 1)$ 区间划分为若干个互不重叠的子区间，每个子区间的长度等于对应符号的概率。
   - **区间更新过程**：
     (a) 读取一个符号，找到它在当前区间中所占的子区间。
     (b) 将该子区间设为“当前搜索区间”。
     (c) 按照原始概率比例，对这个新区间进行再次划分。
     (d) 重复上述步骤，直到所有符号处理完毕。
   - **输出**：记录最终区间的下限（Base）和长度（Range），或者输出该区间内的一个特定浮点值。

3. **主要优势与特点**：

   - **效率极高**：对于概率非常大的符号（例如概率为 0.99），霍夫曼编码至少要给它 1 位，而算术编码可以给它远小于 1 位的平均长度，理论上能无限接近信源熵。
   - **性能**：尤其在符号概率分布极不均匀时，算术编码的压缩比远优于霍夫曼编码。

4. **局限性**：

   - **计算复杂度**：涉及高精度的浮点运算（或者复杂的整数缩放实现），计算开销比霍夫曼编码大。
   - **错误敏感**：由于是整体编码，序列中任何一位比特错误都会导致解码过程彻底失效，具有显著的错误扩散效应。

8.3.5 例题 20：算术编码过程

题目：已知信源符号概率如下表，请对序列 $a_1a_2a_3$ 进行编码。

符号	a_1	a_2	a_3	a_4
概率 P	0.2	0.2	0.4	0.2
初始子区间	$[0.0, 0.2)$	$[0.2, 0.4)$	$[0.4, 0.8)$	$[0.8, 1.0)$

1. **第一步：处理 a_1**
   - 初始空间是 $[0, 1)$，长度 $L = 1$。
   - 根据初始表，a_1 占据了前 20% 的空间。
   - **锁定区间**：$[0.0, 0.2)$。此时新起点 $Base = 0.0$，新长度 $Range = 0.2$。
2. **第二步：处理 a_2**
   - 此时在 $[0.0, 0.2)$ 区间中，开始计算 a_2 的位置。
   - a_2 在原始比例中是从 20% 开始到 40% 结束。
   - **计算新起点**：$0.0 + (0.2 \times 0.2) = 0.04$ （老起点 + 老长度 $\times$ a_2 的相对起点）。
   - **计算新终点**：$0.0 + (0.2 \times 0.4) = 0.08$ （老起点 + 老长度 $\times$ a_2 的相对终点）。
   - **锁定区间**：$[0.04, 0.08)$。此时新长度 $Range = 0.08 - 0.04 = 0.04$。
3. **第三步：处理 a_3**
   - 现在的区间缩减到了 $[0.04, 0.08)$。我们要在这里面找 a_3。
   - a_3 在原始比例中是从 40% 开始到 80% 结束。
   - **计算新起点**：$0.04 + (0.04 \times 0.4) = 0.056$。
   - **计算新终点**：$0.04 + (0.04 \times 0.8) = 0.072$。
   - **锁定区间**：$[0.056, 0.072)$。

结论：序列 $a_1a_2a_3$ 的编码结果就是落在 $[0.056, 0.072)$ 之间的任何一个数。

8.3.6 例题 21：算术解码过程

题目：已知接收到的编码值为 0.068，序列长度为 3，概率表同上。请还原原始符号。

核心逻辑：看这个数落在哪，就判定是哪个符号；判定完后，把该符号占据的空间“切掉”并将剩余空间“拉伸”回 $[0, 1)$ 寻找下一个。

1. **解码第一个符号**
   - 0.068 落在初始表的 $[0.0, 0.2)$ 之间。
   - **判定**：第一个符号是 a_1。
   - **归一化（拉伸）**：我们要看看 0.068 在 a_1 的初始子区间 $[0.0, 0.2)$ 处于什么相对位置?
   - 公式：$v_2 = (0.068 - 0.0)/0.2 = 0.34$。

2. **解码第二个符号**

   - 拿新值 0.34 去对初始表，它落在 $[0.2, 0.4)$ 之间。
   - **判定**：第二个符号是 a_2。
   - **归一化（拉伸）**：看看 0.34 在 a_2 的子区间 $[0.2, 0.4)$ 里的相对位置。
   - 公式：$v_3 = (0.34 - 0.2)/0.2 = 0.7$。（解释：减去 a_2 的起点 0.2，除以 a_2 的宽度 0.2）。

3. **解码第三个符号**

   - 拿新值 0.7 去对初始表，它落在 $[0.4, 0.8)$ 之间。
   - **判定**：第三个符号是 a_3。

结论：原始序列为 $a_1a_2a_3$。

8.3.7 行程编码 (Run-Length Coding, RLC)

行程编码是一种简单的无损压缩技术，主要用于消除**空间冗余**。

- **核心思想**：如果图像中存在大量连续重复的像素，不再逐个记录像素值，而是记录“像素值”和“该值连续出现的次数（行程长度）”。
- **应用**：在二值图像（如传真、扫描文档）中效果极佳。
- **示例**：序列 `AAAAABBBCC` 可编码为 `(A,5), (B,3), (C,2)`。

8.3.8 比特平面编码 (Bit-Plane Coding)

比特平面编码通过将一幅多比特图像分解为一系列二值图像来处理。

- **分解逻辑**：一个 2^n 灰度级的图像可以分解为 n 个二值平面。高阶平面包含了图像的主要视觉特征，低阶平面则多为噪声。
- **处理方法**：通常结合行程编码或算术编码对每一个二值平面独立压缩。
- **格雷码 (Gray Code) 的应用**：为了减少相邻灰度值变动引起的比特位剧烈翻转，通常先将图像转换为格雷码，再进行平面分解，这样可以增加行程长度，提高压缩率。

8.3.9 JPEG 编码 (块变换编码, Block Transform Coding)

JPEG 是最经典的**有损压缩**标准，其核心是消除**心理视觉冗余**。它采用的是典型的**块编码**策略。

什么是块编码？ **块编码**是指将整幅图像分割成若干个大小固定的非重叠像素块（Blocks），并以这些“块”为基本单位进行处理和编码。这种方法的逻辑是：

- **相关性利用**：图像在局部区域（如 8×8 范围内）通常具有极强的像素相关性，分块处理便于提取局部特征。
- **简化计算**：对全图进行整体变换（如大尺度 DCT）计算量巨大且效率低，分块可以将复杂的全局计算简化为多个并行的局部计算。

核心步骤 ：

1. **分块**：将图像分为 8×8 的非重叠小块。这是块编码的起点，后续所有步骤均以此块为单位独立进行。
2. **变换 (DCT)**：对每块执行离散余弦变换。其目的是将图像从空间域转换到频率域，将能量集中在左上角的低频系数（直流分量 DC）上。
3. **量化 (Quantization)**：**关键步骤**。根据量化表对变换系数除以步长并取整。由于人眼对高频不敏感，高频系数（交流分量 AC）通常被量化为 0，从而实现大幅度压缩。
4. **符号编码**：
   - 对直流系数 (DC) 进行差分编码。
   - 对交流系数 (AC) 进行“之”字形 (Zig-zag) 扫描。
   - 最后通过哈夫曼编码进一步消除编码冗余。

8.3.10 小波编码 (Wavelet Coding)

小波编码是 JPEG2000 标准的基础，相比 DCT 变换编码具有明显的优势。

- **多尺度特性**：通过小波变换将图像分解为不同尺度和方向的子带。
- **优点**：
  - **无块效应**：由于是在全图或大块上进行变换，避免了 JPEG 在低码率下的“方块效应”。
  - **渐进传输**：支持从低分辨率到高分辨率的逐级重构。

8.3.11 基于符号的编码 (Symbol-based Coding)

这种方法常用于包含大量重复符号的文档图像压缩（如 JBIG2 标准）。

- **原理**：将图像表示为预定义符号库（如字母、数字）的集合。
- **过程**：
  1. 建立一个包含所有唯一符号的“字典”。
  2. 记录每个符号在图像中出现的位置（坐标）以及它在字典中的索引。
- **优势**：对于扫描文本，压缩比远高于传统的二值图像压缩。

8.4 图像编码技术综合对比与总结

下表对各种编码技术进行了系统化的对比，涵盖了它们的多重特征及其在实际应用中的优缺点。

编码方法	应用场景	主要特征	优点	缺点
霍夫曼编码	通用无损压缩	1. **块编码** 2. 变长前缀码 3. 统计概率分配	实现简单，无损还原	每个符号至少占 1bit
算术编码	JPEG2000	1. **非块编码** 2. 全局区间缩放映射	压缩比逼近熵极限	计算开销大，容错性差
LZW 编码	GIF, TIFF	1. **块编码**（字典项） 2. 动态字典自适应	无需预知概率分布	字典维护占用内存
行程编码	二值图, 传真	1. **非块/流式编码** 2. 游程计数统计	结构简单，处理极快	像素变化剧烈时效率低
比特平面	灰度图压缩	1. **分层编码** 2. 位深度分解映射	突出视觉重要比特层	低位平面压缩效果差
预测编码	视频, DPCM	1. **非块/递归编码** 2. 消除相邻相关性	显著降低数据动态范围	存在预测误差扩散风险
JPEG (DCT)	数码摄影	1. **块编码**（8×8 块） 2. 频域变换与量化	压缩比高，人眼适配	低码率下产生块效应
小波编码	JPEG2000	1. **非块/全局变换** 2. 多分辨率分解	无块效应，支持渐进传输	算法复杂度较高

8.4.1 块效应 (Blocking Artifacts)

在 JPEG 等基于块变换的有损压缩中，经常会出现“块效应”，这是评价有损压缩质量的重要指标。

- **定义**：图像中 8×8 分块的边界处出现明显的不连续跳变，看起来像是许多小方格拼凑而成。
- **成因**：
  1. **独立变换**：每个 8×8 块是独立进行离散余弦变换 (DCT) 的，块与块之间没有考虑平滑过渡。
  2. **量化误差**：这是核心原因。在压缩过程中，为了节省空间，量化器会粗略地舍弃高频系数。当压缩率过高时，相邻两个块在量化后的直流 (DC) 和低频系数上出现较大偏差，导致块边界处灰度不连续。
- **解决方法**：采用小波编码（全局或大区域变换）或者在解码后使用去块滤波器 (De-blocking Filter)。

8.4.2 冗余消除的针对性总结

为了更清晰地理解算法的本质，我们将各方法与其针对的冗余类型进行归类：

- **消除编码冗余**：霍夫曼编码、算术编码（主要通过概率分布优化码字长度）。
- **消除空间冗余**：行程编码 (RLC)、预测编码 (DPCM)、比特平面编码、LZW 编码（主要通过利用像素间的相关性）。

- **消除心理视觉冗余**：量化 (Quantization) ——这是 JPEG 和所有有损压缩的核心，通过牺牲人眼不敏感的信息换取压缩空间。

8.5 第八章核心重点问题总结

1. **图像压缩（Image Compression）中的三类主要数据冗余是什么？**
   - **编码冗余（Coding Redundancy）**：当为表示一组符号（如像素灰度）而分配的码字所用的平均比特数超过了该信源的熵时，就存在编码冗余。通常通过变长编码（如哈夫曼编码）解决。
   - **空间/时间冗余（Interpixel/Interframe Redundancy）**：由于图像中相邻像素之间（或视频帧间）存在高度的相关性，导致数据重复。通常通过预测编码、变换编码或行程编码解决。
   - **心理视觉冗余（Psychovisual Redundancy）**：人眼对不同频率和亮度的敏感度不同，某些信息虽然存在但人类视觉系统无法感知。通过有损压缩（量化）去除这部分信息，可以显著提高压缩比。
2. **如何评价图像压缩算法的性能（保真度准则）？**
   - **客观保真度准则**：通过数学公式度量原始图像与重建图像的误差。常用指标包括均方根误差（RMS Error）和峰值信噪比（PSNR）。PSNR 值越高，代表重构质量越好。
   - **主观保真度准则**：通过观察者对图像质量的直接评价（如平均评价得分 MOS）来衡量。在有损压缩中，主观准则往往比客观准则更贴合实际应用需求。
3. **霍夫曼编码（Huffman Coding）与算术编码（Arithmetic Coding）的主要区别是什么？**
   - **霍夫曼编码**：属于块编码，为每个信源符号分配一个唯一的码字。其限制在于每个符号至少要占 1 bit，在符号概率极大时效率无法达到最优。
   - **算术编码**：属于非块编码，将整个输入序列映射为 $[0,1)$ 区间内的一个单一浮点数。它能够实现分数值的平均比特数，理论上能无限接近信源熵，是性能最强的无损编码之一。
4. **在 JPEG 压缩标准中，量化（Quantization）步骤起到了什么作用？**量化是 JPEG 流程中唯一的**不可逆**步骤，也是产生压缩收益和图像降质的根本原因。
   - **作用**：通过将 DCT 变换后的系数除以量化表并取整，舍弃掉大量人眼不敏感的高频分量。
   - **结果**：使得变换矩阵中出现大量的零值，为后续的“之”字形扫描和行程编码创造了极高的压缩条件。
5. **JPEG 压缩中的“块效应（Blocking Artifacts）”是如何产生的？**块效应表现为图像中 8×8 分块边缘的不连续跳变。
   - **成因**：因为 JPEG 是对每个 8×8 像素块独立进行 DCT 变换和量化的。当压缩率过高时，量化过程会导致相邻块在直流（DC）分量和低频分量上的偏差增大，由于块之间没有平滑衔接，边界处便产生了视觉上的阶跃。
6. **小波编码（Wavelet Coding）相比传统的 DCT 变换编码有哪些优势？**

- **无块效应**：小波变换是在全图或较大的瓦片（Tiles）上进行的，重构图像边缘平滑，在低码率下表现远优于 JPEG。
- **多尺度特性**：提供图像在不同分辨率下的描述，支持“渐进传输（Progressive Transmission）”，即用户可以先接收低分辨率缩略图，再逐步接收细节。
- **高压缩比**：在保持同等画质的前提下，JPEG2000（基于小波）的压缩率通常比普通 JPEG 提高 20%–50%。

7. **LZW 编码的自适应性体现在哪里？** LZW 编码不需要预先分析数据的统计概率。它在扫描数据的过程中动态地构建字典，将重复出现的序列（字符串）映射为短索引。对于具有大量重复模式的图像（如带大面积单一背景的 GIF 或 TIFF 图像），随着字典的不断充实，其压缩性能会越来越强。

9 形态学图像处理 (Morphological Image Processing)

形态学（Morphology）一词源于生物学，意为“研究形状的学科”。在数字图像处理中，数学形态学（Mathematical Morphology）是一种用于提取图像分量的工具，对于表示和描述区域形状（如边界提取、骨架提取、凸包计算）非常有用。

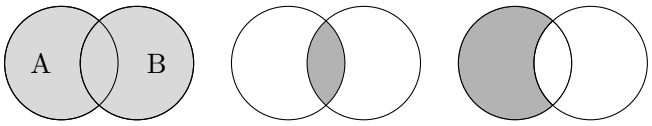
9.1 预备知识

数学形态学的理论基础是集合论。在二值形态学中，我们将图像看作是二维整数空间 Z^2 中的点集。

9.1.1 集合论基本概念

在二值形态学中，集合 A 代表图像中的前景像素点。设 $a = (a_1, a_2)$ 是集合中的元素，表示像素的坐标。

- **子集 ($A \subseteq B$)**：若 A 的每个元素都是 B 的元素，则称 A 是 B 的子集。
- **并集 ($C = A \cup B$)**：两个集合合并后的总区域。
- **交集 ($D = A \cap B$)**：两个集合重叠的共有区域。
- **补集 (A^c)**：相对于全集 Z^2，不属于 A 的所有点（即背景）。
- **差集 ($A - B$)**：属于 A 但剔除掉属于 B 的部分，即 $A - B = A \cap B^c$。

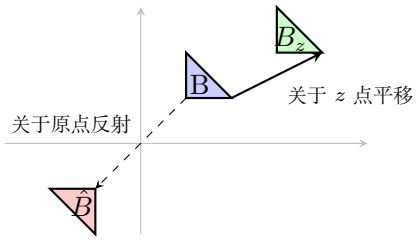


并集 $A \cup B$　　交集 $A \cap B$　　差集 $A - B$

9.1.2 反射与平移 (Reflection and Translation)

这两个操作是定义“腐蚀”与“膨胀”的数学基石。

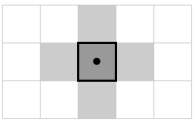
1. **反射 (Reflection)**：集合 B 关于原点的反射记为 $\hat{B}$，定义为：$\hat{B} = \{w \mid w = -b, b \in B\}$。若 B 是一个结构元，$\hat{B}$ 相当于将 B 绕其原点旋转 180°。
2. **平移 (Translation)**：集合 A 沿位移向量 $z = (z_1, z_2)$ 的平移记为 $A_z = \{c \mid c = a + z, a \in A\}$。



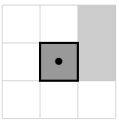
9.1.3 3. 结构元 (Structuring Element, SE)

结构元是探测图像形状的“探针”。

- **原点**：决定了当前探测的位置，结果图像在该位置的取值由结构元覆盖范围内的计算结果决定。
- **形状设计**：形态学处理的效果高度取决于结构元的形状和大小。
- **“不关心” 元素 (Don't-care elements)**：在表示结构元时，有时某些位置并不影响计算结果。这些位置通常在网格中留空（或不加标记），表示在进行集合运算时，这些位置的图像像素值无论是 0 还是 1 都不参与匹配判定。这在“击中或击不中变换”中尤为重要。



十字型 SE (中心为原点)



非对称 SE (原点在角点)

9.2 二值图像形态学处理

二值形态学处理的对象是集合。在二值图像中，我们通常将感兴趣的物体像素（灰度值为 1）看作集合 A，而背景（灰度值为 0）看作其补集。

9.2.1 腐蚀 (Erosion)

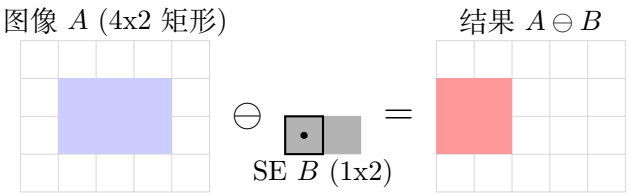
腐蚀操作类似于一种“收缩”或“细化”操作。它会消除图像中比结构元小的物体，并削减物体的边界。

数学定义 结构元 B 对集合 A 的腐蚀表示为 $A \ominus B$，定义为：

$$A \ominus B = \{z \mid (B)_z \subseteq A\}$$

该公式的物理意义是：将结构元 B 平移，使得其原点位于 z 点。如果此时平移后的 B **完全包含**在集合 A 中，那么该点 z 就属于腐蚀后的结果。

图解说明



注：腐蚀的结果取决于 SE 的原点位置。在上图中，只有当 SE 完全进入蓝色区域时，对应的原点位置才会被保留。

9.2.2 膨胀 (Dilation)

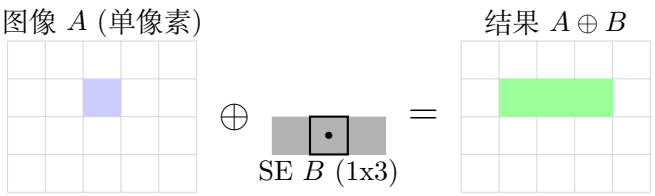
膨胀是腐蚀的逆过程，类似于“增长”或“粗化”操作。它会填充物体内的小孔洞，并连接距离较近的物体。

数学定义 结构元 B 对集合 A 的膨胀表示为 $A \oplus B$，定义为：

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

物理意义是：将结构元 B 关于原点旋转 180 度得到 $\hat{B}$，将其平移到 z 点。只要此时 $\hat{B}$ 与集合 A **至少有一个像素重叠**（交集不为空），那么该点 z 就属于膨胀后的结果。

图解说明



注：膨胀会使物体向四周扩展。如果 SE 是 1x3 且原点在中心，单像素物体将变成 1x3 的横条。

9.2.3 腐蚀与膨胀的对偶性 (Duality)

形态学处理中一个极重要的性质是腐蚀和膨胀互为对偶。这意味着我们可以通过对背景进行膨胀来间接实现对前景的腐蚀。

对偶性公式

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

$$(A \oplus B)^c = A^c \ominus \hat{B}$$

其中 A^c 是 A 的补集（背景），$\hat{B}$ 是结构元的反射。

直观理解 如果你想“收缩”白色的物体，你可以选择直接“腐蚀”白色区域，也可以选择去“膨胀”黑色的背景，两者的结果在逻辑上是一致的。

9.2.4 腐蚀与膨胀的应用及效果对比

在实际的图像处理任务中，腐蚀和膨胀通常被用来消除噪声或提取特征。

- **腐蚀的效果**：
  - **消除细小物体**：可以移除图像中面积小于结构元的噪声点（如“胡椒”噪声）。
  - **断开窄连接**：如果两个物体之间有细微的桥接，腐蚀可以将其切断。
  - **物体收缩**：整体缩小前景物体的轮廓。
- **膨胀的效果**：
  - **填充小孔洞**：可以填充物体内部小于结构元的空隙（如“盐”噪声）。
  - **连接断裂处**：如果一个物体因为噪声或采样问题断开，膨胀可以将其重新连接。
  - **物体扩张**：整体加粗前景物体的轮廓。

9.2.5 开操作 (Opening)

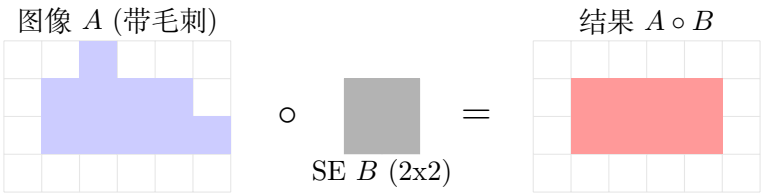
开操作是先腐蚀后膨胀的过程。

数学定义 使用结构元 B 对集合 A 进行开操作表示为 $A \circ B$，定义为：

$$A \circ B = (A \ominus B) \oplus B$$

物理意义与几何解释 开操作具有一种“平滑轮廓”的作用。它能切断窄的间断，消除细长的突出物，并移除小于结构元的孤立亮点。从几何上看，$A \circ B$ 是由结构元 B 能够完全平移并包含在 A 内部的所有点的并集。这就像是用一个圆形的刷子在图形内部滚过，刷子够不到的地方（细刺、毛边）都会被抹除，但物体的整体面积不会像单纯腐蚀那样大幅缩减。

图解说明



9.2.6 闭操作 (Closing)

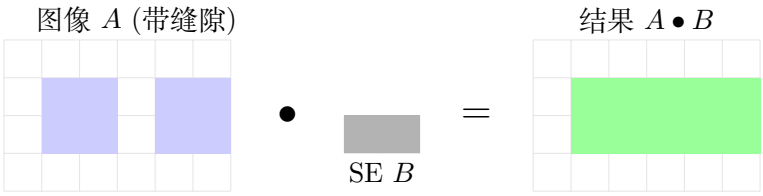
闭操作是先膨胀后腐蚀的过程。

数学定义 使用结构元 B 对集合 A 进行闭操作表示为 $A \bullet B$，定义为：

$$A \bullet B = (A \oplus B) \ominus B$$

物理意义与几何解释 闭操作同样具有“平滑轮廓”的作用，但它主要针对的是内部。它能熔合窄的间断和长而窄的沟壑，消除细小的孔洞，并填补轮廓线上的断裂。从几何上看，$A \bullet B$ 的结果包含所有不在“结构元 B 在集合 A 的外部滚动时所覆盖范围”内的点。

图解说明



9.2.7 开闭运算的性质与对偶性

开运算和闭运算也满足对偶关系：

$$(A \bullet B)^c = A^c \circ \hat{B}$$

$$(A \circ B)^c = A^c \bullet \hat{B}$$

幂等性 (Idempotence) 这是一个非常重要的性质：对图像进行多次开操作（或闭操作），其结果与进行一次操作完全相同。

$$(A \circ B) \circ B = A \circ B$$

$$(A \bullet B) \bullet B = A \bullet B$$

这说明开闭操作是一种稳定的变换，一旦消除了特定尺寸的噪声或填补了缝隙，重复操作不会进一步改变图像。

9.2.8 开闭操作的视觉效果总结

为了区分这两个操作对边界和内部的影响，我们可以总结如下：

操作名称	对边界的影响	对细节的影响
开操作 $(A \circ B)$	平滑轮廓，但**基本维持**原物体大小。	消除细小的突出物（刺）、断开窄桥。
闭操作 $(A \bullet B)$	平滑轮廓，但**基本维持**原物体大小。	填充细小的孔洞、熔合窄的缝隙（沟壑）。

关键结论 ：开操作和闭操作之所以比单纯的腐蚀和膨胀更常用，正是因为它们在消除噪声的同时，能够恢复出物体原本的尺寸（因为先缩后涨或先涨后缩，大部分边界位置相互抵消了）。

9.2.9 击中或击不中变换 (Hit-or-Miss Transform)

击中或击不中变换是形态学中用于物体识别和模式匹配的基础工具。它的核心任务是：在图像 A 中定位一个特定的局部形状。

基本原理 如果我们想在图像中寻找一个特定的形状（如某个特定的字母或零件），我们需要满足两个条件：

1. **“击中”目标**：结构元的前景部分必须与图像中的物体完全重叠。
2. **“击不中”背景**：结构元的背景部分必须与图像中的背景完全重合。

只有当这两个条件同时满足时，才能确定找到了目标。

数学定义 设 $B = (B_1, B_2)$ 是一个复合结构元，其中 B_1 是由我们想要找的形状（目标）组成的集合，B_2 是由目标周围的背景组成的集合。集合 A 被 B 击中或击不中变换表示为 $A \circledast B$，定义为：

$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

其中：

- $A \ominus B_1$ 得到的是所有可能包含目标 B_1 的候选位置（击中部分）。
- $A^c \ominus B_2$ 得到的是所有背景环境匹配 B_2 的候选位置（击不中部分）。
- 两者的**交集**即为目标准确出现的位置（通常结果为单像素点）。

结构元的简化表示 在实际应用中，我们常将 B_1 和 B_2 合并成一个单一的结构元。在这个结构元中：

- 值为 **1** 的位置属于 B_1（必须匹配前景）。
- 值为 **0** 的位置属于 B_2（必须匹配背景）。
- **不关心 (Don't care)** 的位置通常标记为 "X"，表示该位置无论是前景还是背景都不影响匹配结果。

图解说明

0	0	0
0	1	0
0	0	0

用于检测孤立点的 SE

X	X	X
0	1	1
X	X	X

用于检测左侧端点的 SE

应用效果 该变换是形态学识别的基础，广泛应用于：

- **特定形状提取**：如在复杂的电路板图中寻找特定的焊盘形状。
- **端点检测**：寻找线段的终点。
- **细化和骨架算法**：作为这些高级算法的基本迭代步骤。

9.2.10 基本形态学算法

本节介绍如何通过组合基础算子解决实际问题。在以下算法中，A 表示原始二值图像集合，B 表示结构元，A^c 表示图像背景（补集）。

1. 边界提取 (Boundary Extraction)

- **数学表达式**：$\beta(A) = A - (A \ominus B)$
- **变量含义**：A 为原图像，B 为位移对称的结构元（如 3×3 全 1 矩阵）。
- **算法结果**：得到物体 A 的**内边界**。原理是从原图中减去收缩后的内核，剩下的边缘即为边界。

2. 孔洞填充 (Region Filling)

- **数学表达式**：$X_k = (X_{k-1} \oplus B) \cap A^c, \quad k = 1, 2, 3...$
- **变量含义**：X_0 是位于孔洞内部的一个已知像素点（种子点）；B 是十字型结构元；A^c 是原图的背景。
- **算法结果**：当 $X_k = X_{k-1}$ 时迭代停止，最终输出 $X_k \cup A$ 得到**完全填充孔洞后的图像**。

3. 连通分量提取 (Extraction of Connected Components)

- **数学表达式**：$X_k = (X_{k-1} \oplus B) \cap A, \quad k = 1, 2, 3...$
- **变量含义**：X_0 是属于某个连通区域的种子点；B 为 8-连通或 4-连通结构元；A 是原图。
- **算法结果**：当迭代收敛时，X_k 即为图像中包含该种子点的**独立连通物体**。

4. 凸包 (Convex Hull)

- **数学表达式**：$X_k^i = (X_{k-1} \circledast B^i) \cup A$
- **变量含义**：$\circledast$ 为击中或击不中变换；B^i 是四个方向的结构元序列。
- **算法结果**：令 $D^i = X_{k=converge}^i$，最终输出 $C(A) = \bigcup_{i=1}^4 D^i$。得到包含物体的**最小凸多边形区域**。

5. 细化 (Thinning)

- **数学表达式**：$A \otimes B = A - (A \circledast B)$
- **变量含义**：$\otimes$ 表示细化符号；B 是击中或击不中变换的结构元序列。
- **算法结果**：迭代剔除符合条件的边界像素，最终得到物体的**单像素宽度轮廓**，常用于 OCR 预处理。

6. 粗化 (Thickening)

- **数学表达式**：$A \odot B = A \cup (A \circledast B)$
- **变量含义**：$\odot$ 表示粗化符号。
- **算法结果**：在物体边缘外增加像素，输出**加粗后的物体**。通常用于补齐断裂的线段。

7. 骨架 (Skeleton)

- **数学表达式**：$S(A) = \bigcup_{k=0}^K \{(A \ominus kB) - [(A \ominus kB) \circ B]\}$
- **变量含义**：kB 表示对 A 连续进行 k 次腐蚀；K 是 A 被腐蚀为空集前的最后一次迭代。
- **算法结果**：输出物体的**中轴线集合**，能够以最简洁的线条保留物体的拓扑结构。

8. 裁剪 (Pruning) 裁剪是细化操作的后处理过程，用于去除多余的毛刺（Parasitic components）。

- **第一阶段 (X_1)**：$X_1 = A \otimes \{B\}$。消除长度小于指定阈值的毛刺，但也缩短了主干。
- **第二阶段 (X_2)**：$X_2 = \bigcup_{k=1}^8 (X_1 \circledast B^k)$。检测 X_1 中被缩短后的主干端点。
- **第三阶段 (X_3)**：$X_3 = (X_2 \oplus B) \cap A$。通过限制膨胀，让主干线条沿着原图 A 的路径恢复长度。
- **最终结果**：输出 $X_1 \cup X_3$。得到既无毛刺又保持主干完整的几何骨架。

9.2.11 形态学操作符号汇总

下表总结了本章涉及的所有数学符号、对应的形态学操作及其核心功能，以便于在复杂的组合算法中进行快速查阅。

符号	操作名称	数学定义/表达式	主要功能与效果
A^c	**补集**	$\{w \mid w \notin A\}$	定义图像的背景区域。
$\hat{B}$	**反射**	$\{w \mid w = -b, b \in B\}$	将结构元绕原点旋转 180°。
A_z	**平移**	$\{c \mid c = a + z, a \in A\}$	将集合 A 沿向量 z 移动。
$A \ominus B$	**腐蚀**	$\{z \mid (B)_z \subseteq A\}$	收缩物体，消除小于结构元的细节。
$A \oplus B$	**膨胀**	$\{z \mid (\hat{B})_z \cap A \neq \emptyset\}$	扩张物体，填充空隙，连接邻近物体。
$A \circ B$	**开操作**	$(A \ominus B) \oplus B$	**先腐蚀后膨胀**。平滑外部轮廓，断开窄桥，消除细长突出物。
$A \bullet B$	**闭操作**	$(A \oplus B) \ominus B$	**先膨胀后腐蚀**。平滑内部轮廓，熔合窄缝隙，填充小孔洞。
$A \circledast B$	**击中或击不中**	$(A \ominus B_1) \cap (A^c \ominus B_2)$	**形状检测**。同时匹配前景目标和背景环境。
$A \otimes B$	**细化**	$A - (A \circledast B)$	剥离边界像素，提取单像素宽度的轮廓。
$A \odot B$	**粗化**	$A \cup (A \circledast B)$	在特定结构处增加像素，加粗线条。
$\beta(A)$	**边界提取**	$A - (A \ominus B)$	提取物体的内部边缘轮廓。
$S(A)$	**骨架提取**	$\bigcup_{k=0}^{K} S_k(A)$	提取能代表物体形状特征的中轴线。

9.2.12 例题 22：基于形态学与逻辑运算的垫圈形状检测

题目：某高技术制造工厂与当地政府签订了一份合同，制造高精度垫圈（如下图所示）。合同要求使用图像系统来检测所有垫圈的形状偏差（内边缘和外边缘）。假设：(1) 存在一幅能够满足要求的垫圈“金（Golden）”图像（即完美无暇的标准图像）；(2) 系统成像和定位精度足够高，可以忽略数字化和定位引起的误差。请根据形态学/逻辑运算提出一种检测方案。

1. **第一步：图像获取与预处理**

   - **照明设计**：由于关注点在边界缺陷，采用背光照明系统（Back-lighting），以产生高质量的二值图像。
   - **二值化处理**：将采集到的待测图像转换为二值图像 A，目标（垫圈实体）像素值为 1，背景为 0。

2. **第二步：执行逻辑对比运算**

   - **定义参考图**：令“金”图像为 G。根据定义，其补集 G^c 代表了标准垫圈以外的所有区域。
   - **逻辑运算**：将待测图像 A 与标准图的补集 G^c 进行“逻辑与”操作：

     $$R = A \cap G^c \quad （或表示为 R = A \text{ AND } G^c）$$

   - **原理分析**：如果垫圈完美，则 A 中为 1 的像素在 G^c 中对应的位置必然全部为 0，结果 R 为全全黑（全 0）。若 A 在标准范围之外有突出（尺寸偏大或形变），R 中会出现值为 1 的区域。

- **双向检测**：同理，通过 $G \cap A^c$ 可检测待测件是否存在缺口或尺寸偏小（内/外径收缩）。

3. **第三步：连通分量提取与分析**

- **提取差异区域**：利用连通分量算法提取结果图像 R 中所有值为 1 的连通区域。
- **计算特征**：统计每个连通分量的数量及面积（像素个数）。

结论：根据提取出的连通分量判定检测结果：

- **完美匹配**：连通分量个数为 0。
- **容差判定**：若允许微小偏差，可设置阈值（例如：连通分量面积小于 T 个像素则忽略）。
- **不合格**：若存在面积超过预设值的连通分量，则判定该垫圈存在形状偏差，予以剔除。

9.3 灰度级形态学

灰度级形态学将形态学操作扩展到灰度图像。此时，图像被视为三维空间中的曲面，$f(x, y)$ 代表坐标 (x, y) 处的灰度高度。

9.3.1 灰度腐蚀与膨胀

在灰度级形态学中，结构元 b 通常是一个小型平坦矩阵（类似于滤波掩模）。

- **灰度腐蚀 (Erosion)**：
$$(f \ominus b)(x, y) = \min_{(s,t)\in B} \{f(x + s, y + t)\}$$
效果：在结构元范围内取**最小值**。它会使图像整体变暗，减小明亮区域，并消除较小的亮细节。

- **灰度膨胀 (Dilation)**：
$$(f \oplus b)(x, y) = \max_{(s,t)\in B} \{f(x - s, y - t)\}$$
效果：在结构元范围内取**最大值**。它会使图像整体变亮，扩大明亮区域，并填充较小的暗缝隙。

9.3.2 灰度开运算与闭运算

定义与二值形态学一致，但操作基于灰度腐蚀和膨胀。

- **开运算**：$f \circ b = (f \ominus b) \oplus b$。用于去除较小的亮细节，同时保持背景和较大的亮区域基本不变。
- **闭运算**：$f \bullet b = (f \oplus b) \ominus b$。用于去除较小的暗细节，同时保持明亮区域基本不变。

9.3.3 典型灰度形态学算法

通过这些基本操作，可以实现图像增强和特征提取：

- **形态学梯度 (Morphological Gradient)**：
$$g = (f \oplus b) - (f \ominus b)$$
利用膨胀和腐蚀的差值提取图像边缘，能够增强区域间的灰度过渡。

- **顶帽变换 (Top-hat Transform)**：

$$T_{hat}(f) = f - (f \circ b)$$

用原图减去开运算结果。主要用于**提取暗背景上的亮物体**，在校正不均匀光照（阴影校正）方面非常有效。

- **底帽变换 (Bottom-hat Transform)**：

$$B_{hat}(f) = (f \bullet b) - f$$

用闭运算结果减去原图。主要用于**提取亮背景上的暗物体**。

9.4 第九章核心重点问题总结

1. **数学形态学（Mathematical Morphology）的基本思想是什么？**
   - 形态学不是基于像素值的算术运算，而是基于**集合论**的。其核心思想是利用一个具有特定形状的“探针”——**结构元（Structuring Element）**在图像中移动，通过检查结构元与图像之间的包含关系或相交关系，来提取图像的几何结构特征（如边界、骨架、孔洞等）。
2. **腐蚀（Erosion）与膨胀（Dilation）在二值图像处理中的视觉效果有何本质区别？**
   - **腐蚀**：是一个“收缩”过程。它会消除图像中比结构元小的物体，削减物体的边界，并断开物体之间微弱的连接。
   - **膨胀**：是一个“增长”过程。它会填充物体内部的小孔洞，桥接物体间的微小间断，并增加物体的边界厚度。
   - **对偶性**：对前景的腐蚀等价于对背景的膨胀再取补集。
3. **开操作（Opening）与闭操作（Closing）的应用场景分别是什么？**
   - **开操作**（先腐蚀后膨胀）：主要用于**去除明亮的细小噪声**（如胡椒噪声）或平滑物体的外部轮廓，它能断开狭窄的连接而基本保持物体面积不变。
   - **闭操作**（先膨胀后腐蚀）：主要用于**填充暗色的细小裂缝**或物体内部的孔洞，它能熔合狭窄的沟壑，平滑物体的内部轮廓。
4. **击中或击不中变换（Hit-or-Miss Transform）的判定标准是什么？**击中或击不中变换是用于检测特定形状的工具。它要求两个条件同时满足：
   - 结构元的前景部分 B_1 必须完全包含在图像的前景中（“击中”目标）；
   - 结构元的背景部分 B_2 必须完全包含在图像的背景中（“击不中”目标）。
   - 数学表示为：$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$。
5. **在基本形态学算法中，如何利用腐蚀和膨胀提取图像边界？**边界提取的核心在于利用腐蚀缩小物体。内边界提取公式为：$\beta(A) = A - (A \ominus B)$。即原图减去腐蚀后的图像，剩下的部分就是原图的最外层像素。结构元 B 的大小决定了边界的粗细。

6. **孔洞填充（Region Filling）算法中，为什么要与原图的补集 A^c 取交集？**在迭代公式 $X_k = (X_{k-1} \oplus B) \cap A^c$ 中，膨胀操作负责让种子点向四周扩散填充孔洞，而**与 A^c 取交集**起到了掩模约束的作用。它确保填充过程只发生在背景区域（即孔洞内部），防止种子点的扩张越过原有的边界而“溢出”到整个图像。

7. **灰度级形态学（Grayscale Morphology）与二值形态学的主要运算差异是什么？**二值形态学的集合运算在灰度级形态学中演变为**统计运算**：

   - **灰度腐蚀**：对应局部区域内的**最小值**运算，效果是增强暗色区域，抑制亮细节。
   - **灰度膨胀**：对应局部区域内的**最大值**运算，效果是增强亮区域，抑制暗细节。

8. **顶帽变换（Top-hat）与底帽变换（Bottom-hat）在实际工程中有什么用途？**

   - **顶帽变换**（$f - f \circ b$）：用于提取暗背景上的亮物体，最关键的用途是**校正不均匀光照**的影响。通过减去开运算得到的背景估计，可以使背景趋于均匀。
   - **底帽变换**（$f \bullet b - f$）：用于提取亮背景上的暗物体。