

**学生实验报告**

实验课名称：数字图像处理

实验项目名称：数学形态学(python实现)

专业名称：人工智能

班级： 2022240401

学号： 2022905226

学生姓名：

教师姓名：

实验地点:计算中心机房WM2204

实验日期：2024.10.21

[**一、实验名称** 3](#_Toc180257856)

[**二、实验目的与要求** 3](#_Toc180257857)

[**三、实验内容** 3](#_Toc180257858)

[**四、实验仪器与设备** 4](#_Toc180257859)

[**五、实验原理** 4](#_Toc180257860)

[**5.1** **二值形态学** 4](#_Toc180257861)

[**5.1.1** **基本概念** 4](#_Toc180257862)

[**5.1.2** **实现原理** 5](#_Toc180257863)

[**5.1.2.1 腐蚀操作** 5](#_Toc180257864)

[**5.1.2.2 膨胀操作** 5](#_Toc180257865)

[**5.1.2.3 开运算** 6](#_Toc180257866)

[**5.1.2.4 闭运算** 7](#_Toc180257867)

[**5.2** **灰值形态学 - 膨胀、腐蚀、开/闭运算** 8](#_Toc180257868)

[**5.2.1** **基本概念** 8](#_Toc180257869)

[**5.2.2** **实现原理** 9](#_Toc180257870)

[**5.2.2.1 灰值膨胀** 9](#_Toc180257871)

[**5.2.2.2 灰值腐蚀** 10](#_Toc180257872)

[**5.2.2.3 灰值开运算** 11](#_Toc180257873)

[**5.2.2.4 灰值闭运算** 13](#_Toc180257874)

[**5.3** **灰值形态学-梯度求取（边缘）** 14](#_Toc180257875)

[**5.3.1** **基本原理** 14](#_Toc180257876)

[**5.3.2** **实现过程** 14](#_Toc180257877)

[**5.3.3** **结构元素大小的影响** 15](#_Toc180257878)

[**5.3.4** **思考：为什么结构元素变大，边缘会更亮，但是整体图像会变得更模糊** 15](#_Toc180257879)

[**5.4** **灰值形态学-快速细化（骨架提取）** 15](#_Toc180257880)

[**5.4.1** **实现过程** 15](#_Toc180257881)

[**六、实验过程及代码** 16](#_Toc180257882)

[**6.1** **二值形态学处理** 16](#_Toc180257883)

[**6.1.1** **二值腐蚀** 16](#_Toc180257884)

[**6.1.2** **二值膨胀** 17](#_Toc180257885)

[**6.1.3** **开运算和闭运算** 17](#_Toc180257886)

[**6.2** **灰值形态学处理** 17](#_Toc180257887)

[**6.2.1** **灰值腐蚀** 17](#_Toc180257888)

[**6.2.2** **灰值膨胀** 18](#_Toc180257889)

[**6.2.3** **开运算和闭运算** 18](#_Toc180257890)

[**6.3** **灰值形态学-梯度求取（边缘）** 19](#_Toc180257891)

[**6.4** **灰值形态学-快速细化（骨架提取）** 19](#_Toc180257892)

[**6.4.1** **实现代码** 19](#_Toc180257893)

[**七、实验结果与分析** 20](#_Toc180257894)

[**7.1** **二值形态学** 20](#_Toc180257895)

[**7.2** **灰值形态学** 22](#_Toc180257896)

[**7.3** **快速细化（提取骨架）** 24](#_Toc180257897)

[**八、实验总结及心得体会** 25](#_Toc180257898)

# **一、实验名称**

数学形态学(python实现)

# **二、实验目的与要求**

本次实验主要实现并掌握数学形态学的相关算法。编程实现二值形态学膨胀、腐蚀、开/闭运算，编程实现灰值形态学膨胀、腐蚀、开/闭运算，掌握击中/击不中变换、骨架提取算法。能分析图像处理效果，理解二值形态学不同结构元素对结果的影响，会用数学形态学方法解决实际工程问题。

# **三、实验内容**

1. **二值形态学算法设计与实现（40分）**
   1. 编程实现二值形态学膨胀效果
   2. 编程实现二值形态学腐蚀效果
   3. 编程实现二值形态学开/闭运算 注：要在实验报告中体现不同的二值形态学操作对于图像最终的处理结果有何影响！
2. **灰值形态学算法设计与实现（40分）**
   1. 编程实现灰值形态学膨胀效果
   2. 编程实现灰值形态学腐蚀效果
   3. 编程实现灰值形态学开/闭运算
   4. 编程实现灰值形态学边缘提取 注：要在实验报告中体现不同的灰值形态学操作对于图像最终的处理结果有何影响！
3. **快速形态学细化方法提取骨架(20, 可选做)**

# **四、实验仪器与设备**

联想笔记本电脑。 13代i7处理器，32G内存。 4070显卡。

# **五、实验原理**

## **5.1 二值形态学**

### **5.1.1 基本概念**

二值形态学是数学形态学的基础，用于处理二值图像。在二值图像中，像素只有两种可能的值：通常用 0 表示背景，255 表示前景（或目标）。

1. **结构元素**：形态学操作使用一个称为结构元素（Structuring Element）的小型模板来处理图像。结构元素可以看作是一个小的二值图，其形状和大小决定了形态学操作的效果。
2. **膨胀（Dilation）**：膨胀操作会扩大图像前景区域。其数学定义为：

* 其中 A 是输入图像，B 是结构元素， 是 B 的反射。

1. **腐蚀（Erosion）**：腐蚀操作会缩小图像中的前景区域。其数学定义为：
2. **开运算（Opening）**：开运算是先腐蚀后膨胀的组合操作。其数学定义为：
3. **闭运算（Closing）**：闭运算是先膨胀后腐蚀的组合操作。其数学定义为：

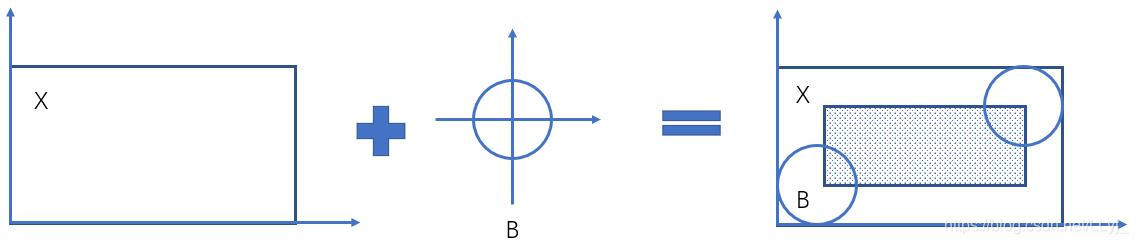
### **5.1.2 实现原理**

#### **5.1.2.1 腐蚀操作**

实现步骤：

* 将结构元素的中心放置在图像的每个像素上。
* 只有当结构元素完全覆盖在前景区域内时，才保留当前像素为前景。

把结构元素B平移a后得到，若包含于X，我们记下这个a点，所有满足上述条件的a点组成的集合称做X被B腐蚀(Erosion)的结果。如下图所示:



**图1：腐蚀操作**

其中X是被处理的对象，B是结构元素。对于任意一个在阴影部分的点a，包含于X，所以X被B腐蚀的结果就是那个阴影部分。阴影部分在X的范围之内，可以看出比之前的X要小了，被撕了一层似的，是将图像中的目标进行**细化的操作**。

结果分析：

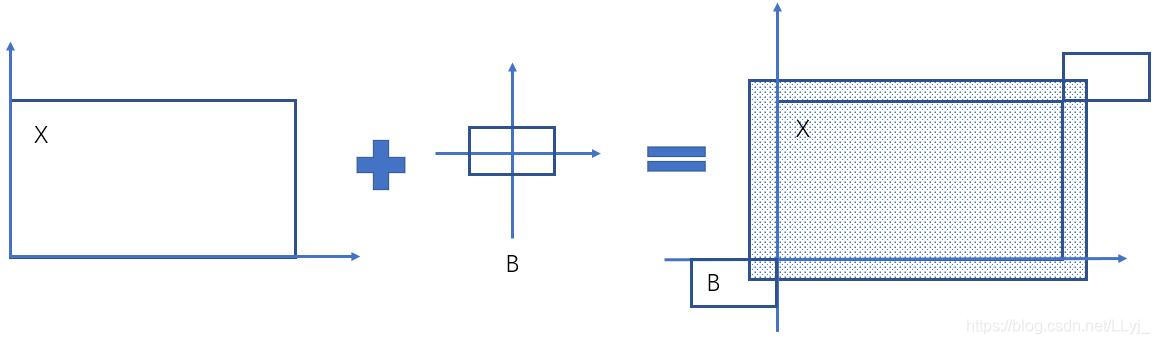
**腐蚀之后的图像通俗理解会变细**，且随着滤波器的大小改变。当滤波器的尺寸越大，则腐蚀效果越强，图像就变得越细；弱滤波器的大小较小，则腐蚀效果较弱。 腐蚀将图像中的高亮区域或白色部分进行缩减细化，其运行结果图比原图的高亮区域更小。主要用于**去除较小的毛刺以及粘连**。

#### **5.1.2.2 膨胀操作**

实现步骤：

* 将结构元素的中心放置在图像的每个像素上。
* 如果结构元素覆盖的区域内有任何前景像素，则将当前像素设为前景。

把结构元素B平移a后得到，若与X至少有一个像素相交，我们记下这个a点，所上述条件的a点组成的集合称做X被B膨胀(Dilation)的结果。如下图所示:



**图2：膨胀操作**

其中X是被处理的对象，B是结构元素，不难知道，对于任意一个在阴影部分的点a，击中X，所以X被B膨胀的结果就是那个阴影部分。阴影部分包括X的所有范围，就象X膨胀了一圈似的。

结果分析：

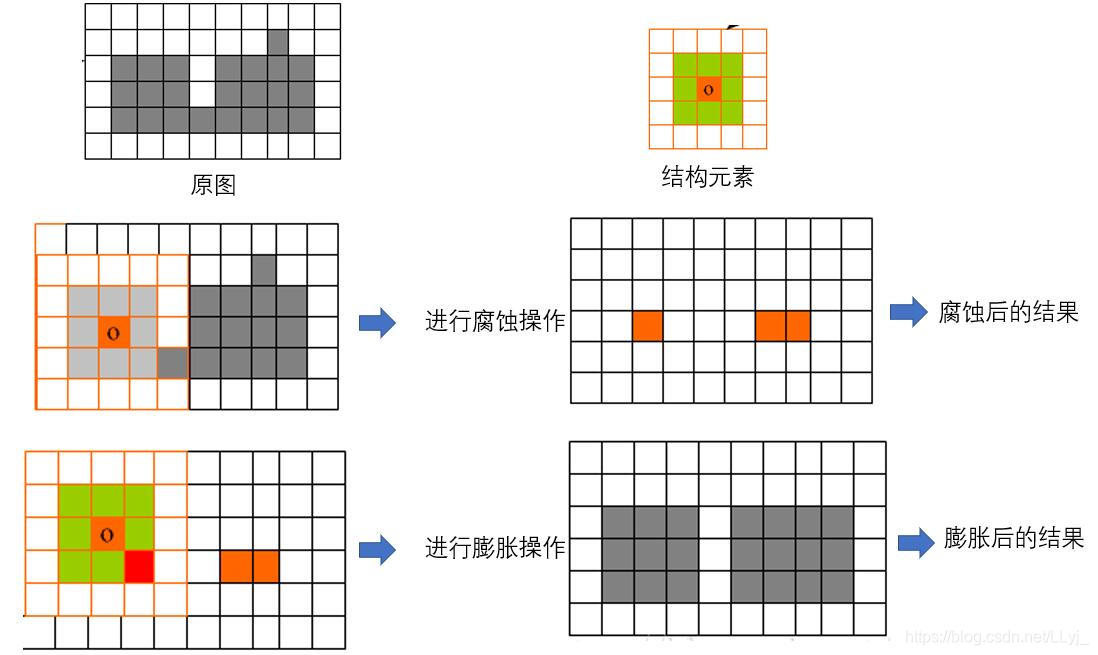
图像膨胀是腐蚀操作的”逆操作”，但严格意义上讲**不是逆操作**，因为对于一个非对称的结构元素，往往腐蚀之后的图像继续用相同的结构元素膨胀后，无法恢复原图像。而此时如果用原结构元素的反射元素进行腐蚀，则可以恢复原图像。所以严格意义上讲，**腐蚀和膨胀不是互逆的**。腐蚀和用原结构元素的反射元素膨胀，是互逆的。

膨胀的效果类似于“领域扩张”，将图像中的高亮区域或白色部分进行扩张，其运行结果图比原图的高亮区域更大，线条变粗了，主要用于**去噪**。以及**填充图像中的小孔和狭窄的缝隙**。

#### **5.1.2.3 开运算**

实现步骤：

* **先进行腐蚀操作，然后对结果进行膨胀**操作。
* 开运算可以去除小的噪点，**平滑对象的轮廓**，但总体上不会显著改变区域的面积。即**去除图像中的“凸”部分以及粘连**。然后还可以**去除小于结构元素的噪点**。同时不会改变图像的整体形状。

开运算操作如下：

**图3：开运算操作**

结果分析：

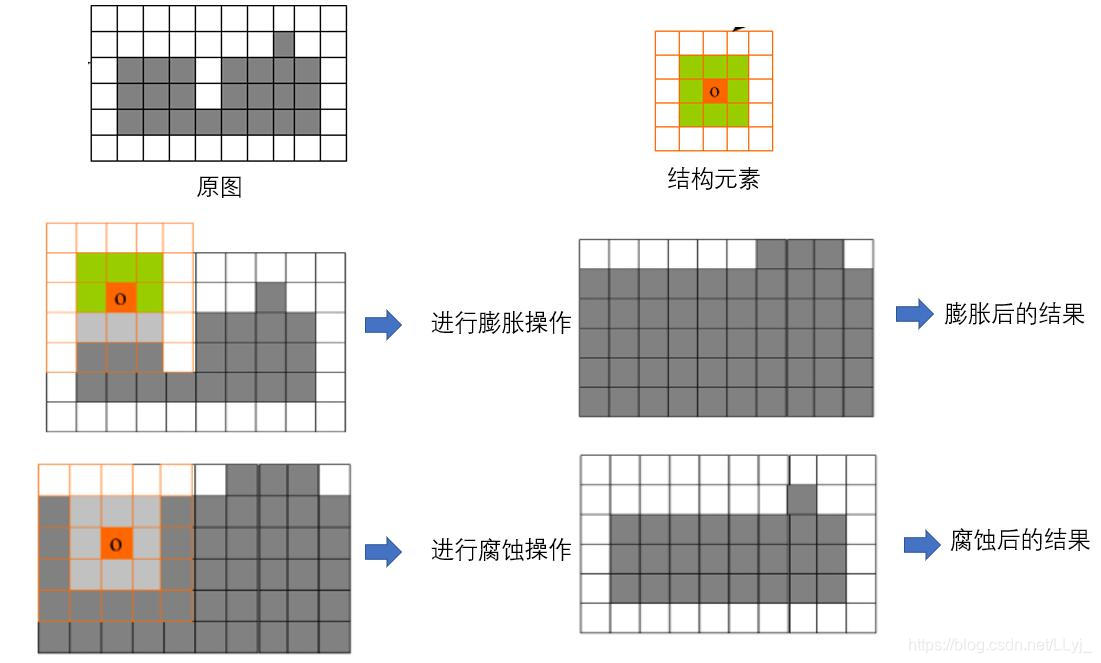
开运算操作之后，图像中的“凸”部分以及粘连被去除，且图像中的噪点也被去除。

#### **5.1.2.4 闭运算**

实现步骤：

* 先进行膨胀操作，然后对结果进行腐蚀操作。
* 闭运算可以填充小洞，**填充图像中的“凹”部分，连接近邻的对象**，平滑轮廓，同样不会显著改变区域的面积。

闭运算操作如下：



**图4：闭运算操作**

结果分析：

闭运算操作之后，图像中的“凹”部分被填充，且图像中的近邻对象被连接。

## **5.2 灰值形态学 - 膨胀、腐蚀、开/闭运算**

### **5.2.1 基本概念**

1. **结构元素**：与二值形态学类似，但可以包含不同的灰度值。
2. **膨胀（Dilation）**：对于灰度图像，膨胀操作结构元素覆盖区域内的最大值。
3. **腐蚀（Erosion）**：对于灰度图像，腐蚀操作取结构元素覆盖区域内的最小值。
4. **开运算（Opening）**：先腐蚀后膨胀的组合操作。
5. **闭运算（Closing）**：先膨胀后腐蚀的组合操作。

公式定义：

* **膨胀操作**：
* **腐蚀操作**：
* **开运算**：
* **闭运算**：
* **形态学梯度**：

其中，是输入图像，是结构元素。

根据公式可以简单将灰值形态学操作理解为：

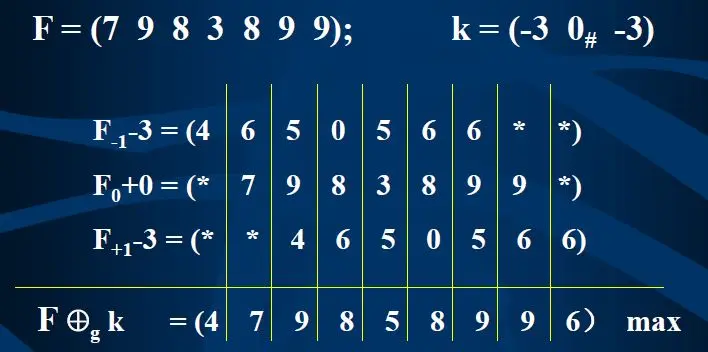
**对于原始图像中坐标处的像素灰度值，分别移动，得到，再加上结构元素中坐标处的灰度值，最后取所有结果的最大值。**

### **5.2.2 实现原理**

#### **5.2.2.1 灰值膨胀**

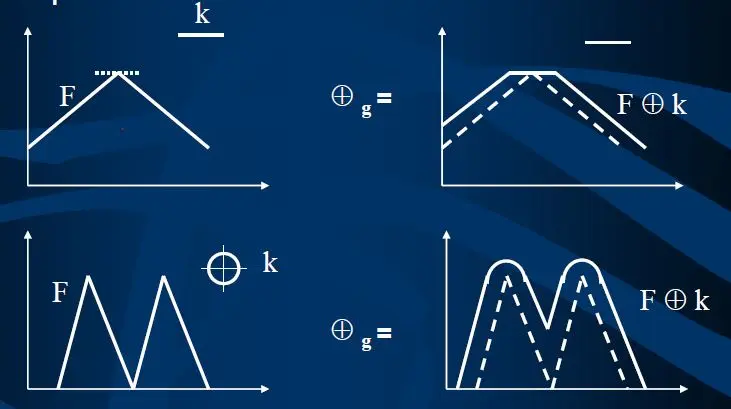
* 将结构元素的中心放置在图像的每个像素上。
* 计算结构元素**覆盖区域内的最大值**（注意要先移动结构元素，并将结构元素中对应的值加到原始图像中，再计算覆盖区域内的最大值），并将结果赋给当前像素。

计算示例：假设有一个**一维的灰度分布列表和一个一维的结构化要素**

**图5：灰值膨胀操作计算实例**

如上图所示，以点坐标，对于结构素中的第一个元素，**先将原始图像向左移动一个单位**，然后**每个灰度值再减3**，记作，空位用\*代替。 对于结构元素中的第二个元素，不用移动，同时每个灰度值加0，不变。 对于结构元素中的第三个元素，**将原始图像向右移动一个单位**，同样**每个灰度值减去3**，记作，余位用\*补齐。 最后，**每一列再求取最大值**，得到最后的结果，即是对原始图像以结构化要素进行灰度膨胀后的结果。可以看出，**原始图像不仅在灰度值上发生了变化，而且空间尺寸变得更大**。

对于二维的灰度分布列表和二维的结构化要素，其膨胀操作的计算过程与一维类似，只不过需要考虑二维的移动。

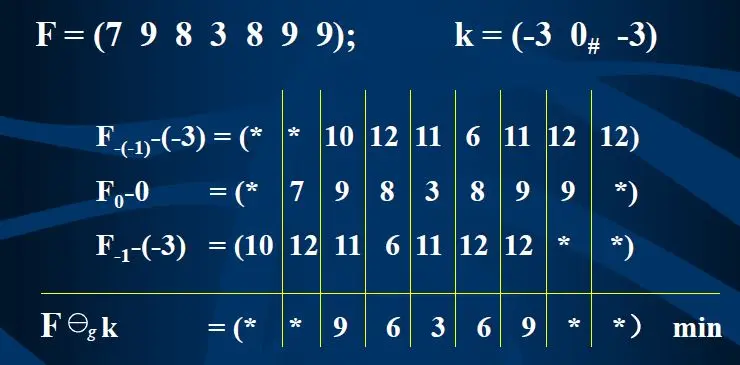
一些灰度膨胀运算的结果如下图所示：

**图6：灰值信号膨胀示例**

#### **5.2.2.2 灰值腐蚀**

* 将结构元素的中心放置在图像的每个像素上。
* 计算结构元素覆盖区域内的最小值（注意要先移动结构元素，并将结构元素中对应的值从原始图像中减去，再计算覆盖区域内的最小值），并将结果赋给当前像素。

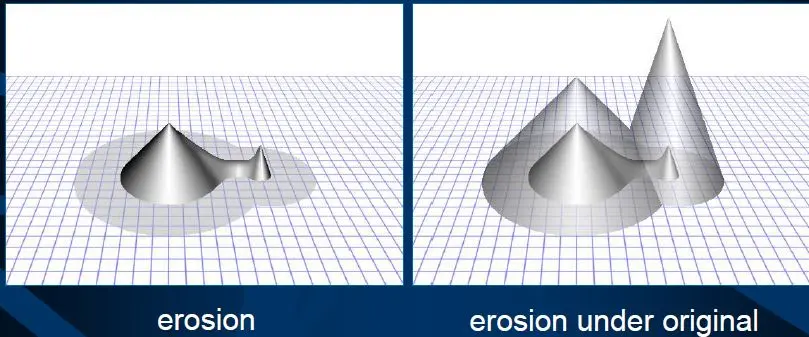
灰度腐蚀的显著效果是**不仅可以使得图像的尺寸变小，其灰度值也会变小**。可以将灰度腐蚀的过程想象成一个雪人受到阳光照射而逐渐溶化的过程。

以一个简单的一维数据为例，介绍灰度腐蚀的计算过程，如下图所示：

**图7：灰值腐蚀操作计算实例**

假定结构化元素中元素为原点，整个计算过程可以分为以下三个步骤： (1) 将原始数据**向右移动一个单位**，并给每个元素减去(-3)，余位用\*代替； (2) 原始数据移动0个单位，并给每个元素减去0，余位用\*代替 (3) 将原始数据**向左移动一个单位**，并将每个元素减去(-3)，余位用\*代替； (4) 依选取每一列中的最小值（默认\*最小），得到经腐蚀后的数据。

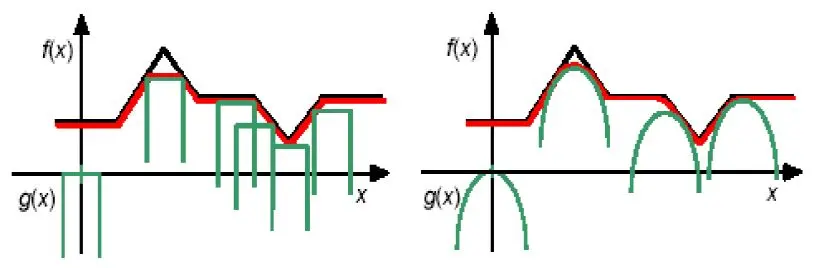
一个腐蚀示例效果图：

**图8：灰值腐蚀示例**

#### **5.2.2.3 灰值开运算**

* 先进行腐蚀操作，然后对结果进行膨胀操作。
* 可以去除小的亮区域，同时保持整体灰度分布。

灰度形态学的腐蚀运算跟二值形态学的开运算一样，也是利用结构化元素，先对原始灰度图像进行腐蚀后，再进行膨胀的运算。

灰度开运算能够**保留灰度图像中特定的强度**的同时，**削弱图像中的某种强度**，如下图所示，用一个一维数据为例，将灰度开运算的过程当作一个用结构化元素从下往上拟合原始数据的过程，图像中的一些尖峰无法拟合,将会被抹去，从而得到削弱图像强度的效果。

**图9：灰值开运算**

灰度开运算的过程，如下图所示，可以看出，对原始图像进行灰度开运算后，原始图像的尺寸并没有发生改变，只是一些灰度值变小了。

**图****10：灰值开运算实例**

灰度开运算的效果分析：

**图11：灰值开运算效果**

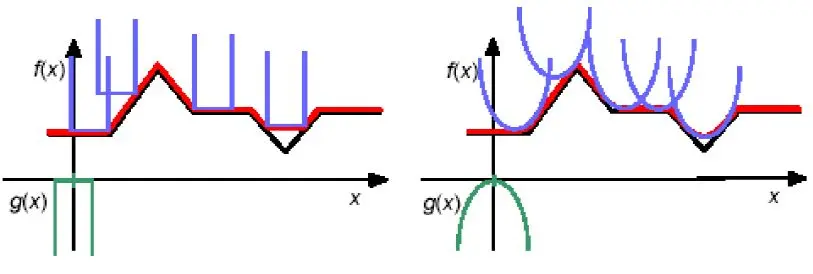
如图，通过灰度开运算，灰度图像中一些被暗色（黑）包围的较小的亮色（白）通过开运算被抹去，而一些较大的亮色区域却被保留了下来。这是因为开运算先进行腐蚀操作，腐蚀操作会使得图像中的亮色区域变小，而后再进行膨胀操作，膨胀操作会使得图像中的亮色区域变大，因此，开运算后的图像中，较小的亮色区域被抹去，而较大的亮色区域却被保留了下来。

同时也可以看出，结构化元素的尺寸越大，灰度图像中被抹去的亮色区域也越大。但是，**灰度开运算后的图像尺寸不会发生改变**。

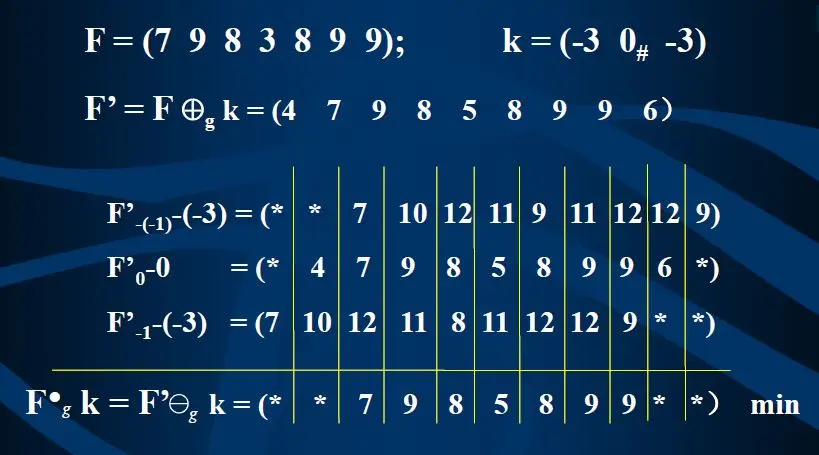
#### **5.2.2.4 灰值闭运算**

* 先进行膨胀操作，然后对结果进行腐蚀操作。
* 可以填充小的暗区域，同时保持整体灰度分布。

从数学公式的形式上，可以看出来，灰度闭运算与二值闭运算的计算过程非常相似，其基本的形态学运算灰度膨胀和腐蚀运算。与二值闭运算相比，灰度闭运算不改变图像本身的大小，而是会改变图像的灰度值。

为了更加形象的说明灰度闭运算的原理，以一个一维数据为例，如下图所示，可以将灰度闭运算看作是利用结构化元素对一副图像个上往下fit的过程，fit的过程中，会进入一些“死区”（尺寸小于结化元素的区域），则“死区”数据会被填埋。

**图12：灰值闭运算**

以数学计算的方式理解，灰度闭运算，其计算过程如下图所示：

**图13：灰值闭运算实例**

可以看到灰度闭运算，并不会改变图像本身的尺寸，而是会对增加图像的像素值，也就是，灰度闭运算会使得图像变得更亮。

灰度闭运算的效果分析：

**图14：灰值闭运算效果**

如图，可以看出灰度闭运算的效果，使得图片中**尺寸较小的亮色区域（像素值较大）的部分尺寸增加**，而**较大的亮色区域并未发生改变**。另外，当灰度闭运算的结构化元素尺寸越大，所造成原始灰度图像中被填埋的区域面积也越大。

## **5.3 灰值形态学-梯度求取（边缘）**

### **5.3.1 基本原理**

形态学梯度的基本定义是**膨胀操作与腐蚀操作的差值**：

其中， 是输入图像， 是结构元素， 表示膨胀操作， 表示腐蚀操作。

### **5.3.2 实现过程**

1. 对输入图像进行膨胀操作
2. 对输入图像进行腐蚀操作
3. 计算膨胀结果和腐蚀结果的差值

**边缘增强**：形态学梯度能有效地突出图像中的边缘，包括物体轮廓和纹理变化。 **对噪声的敏感性**：由于是基于局部极值操作，形态学梯度对噪声比较敏感。 **边缘宽度**：产生的边缘宽度与结构元素的大小有关。

### **5.3.3 结构元素大小的影响**

结构元素的大小对形态学梯度的结果有显著影响：

1. **小结构元素**：
   * 能够检测出更细微的边缘和纹理变化。
   * 对噪声更敏感，可能会产生更多的假边缘。
   * 得到的边缘更细、更精确。
2. **大结构元素**：
   * 能够检测出更显著的边缘，忽略细微变化。
   * 对噪声的抵抗力更强，但可能会丢失一些细节。
   * 得到的边缘更宽、更模糊。

### **5.3.4 思考：为什么结构元素变大，边缘会更亮，但是整体图像会变得更模糊**

1. **边缘变得更亮**：
   * 大尺寸结构元素在边缘处产生更大的灰度差异。膨胀操作会使亮区域扩大，腐蚀操作会使暗区域扩大。在边缘处，这两种效果的差异会更加明显。
2. **整体图像变得更模糊**：
   * 膨胀和腐蚀操作会分别取局部区域的最大值和最小值，这就抹去了区域内的细微变化。就是结构元素越大，越会造成图像的“平滑”。
   * 大尺寸结构元素会影响更大范围的像素，导致非边缘区域的灰度值趋于平均，减少了局部对比度。
3. **边缘宽度增加**：
   * 膨胀和腐蚀操作影响的范围变大，边缘的”过渡区”也随之扩大。

增大结构元素尺寸会导致一种”粗化”效果：主要的边缘特征被强化，而细节信息则被平滑掉。这就是为什么我们会看到边缘更加明显，但整体图像变得更加模糊的原因。

## **5.4 灰值形态学-快速细化（骨架提取）**

本实验实现了两种骨架提取方法：skeletonize 和 medial\_axis。这两种方法都是基于 skimage 库中的形态学操作实现的。

### **5.4.1 实现过程**

1. **图像预处理**：
   * 首先检查输入图像是否为二值图像。如果不是，使用 Otsu 阈值法将其转换为二值图像。
   * 这一步确保了骨架提取算法的输入是标准的二值图像。
2. **骨架提取方法选择**：
   * 根据 deal\_Type 参数选择使用 skeletonize 或 medial\_axis 方法。
   * skeletonize 方法（deal\_Type = 0）：
     + 使用 morphology.skeletonize 函数直接获取骨架。
   * medial\_axis 方法（deal\_Type = 1）：
     + 使用 morphology.medial\_axis 函数计算中轴线。
     + 这个函数还返回距离变换，但在当前实现中未使用。
3. **结果处理**：
   * 将骨架结果（布尔数组）转换为 8 位无符号整数类型的图像。
   * 骨架像素设置为 255（白色），背景像素为 0（黑色）。
4. **性能计时**：
   * 使用 time.time() 函数记录处理时间，以评估算法效率。

# **六、实验过程及代码**

## **6.1 二值形态学处理**

二值形态学处理包括腐蚀、膨胀、开运算和闭运算。这些操作都是基于滑动窗口的原理实现的。

### **6.1.1 二值腐蚀**

二值腐蚀的核心实现如下：

def erosion\_binary(img, np\_kernel):  
 k\_rows, k\_cols = np\_kernel.shape  
 pad\_height, pad\_width = k\_rows // 2, k\_cols // 2  
 padded\_img = np.pad(img, ((pad\_height, pad\_height), (pad\_width, pad\_width)), mode='constant', constant\_values=0)  
 window\_view = as\_strided(padded\_img,  
 shape=(img.shape[0], img.shape[1], k\_rows, k\_cols),  
 strides=padded\_img.strides \* 2)  
 mask = np\_kernel == 1  
 eroded = np.all(window\_view[..., mask] == 255, axis=-1)  
 return eroded.astype(np.uint8) \* 255

实现步骤：

1. 对输入图像进行填充，以处理边缘像素。

2. 使用 as\_strided 创建滑动窗口视图，避免显式循环。

3. 创建结构元素的掩码。

4. 对每个窗口应用腐蚀操作，检查是否所有对应的像素值都为255。

5. 将结果转换回二值图像。

### **6.1.2 二值膨胀**

二值膨胀的实现与腐蚀类似，主要区别在于：

def dilation\_binary(img, np\_kernel):  
 # ... [与腐蚀相同的前置代码] ...  
 np\_kernel\_reflected = np.flip(np\_kernel)  
 mask = np\_kernel\_reflected == 1  
 dilated = np.any(window\_view[..., mask] == 255, axis=-1)  
 return dilated.astype(np.uint8) \* 255

关键区别：

1. 使用结构元素的反射（np.flip(np\_kernel)）。

2. 使用 np.any 而不是 np.all，只要有一个像素为255就认为是前景。

### **6.1.3 开运算和闭运算**

开运算和闭运算是腐蚀和膨胀的组合：

def open\_binary(img, np\_kernel):  
 return dilation\_binary(erosion\_binary(img, np\_kernel), np\_kernel)  
   
def close\_binary(img, np\_kernel):  
 return erosion\_binary(dilation\_binary(img, np\_kernel), np\_kernel)

这两个操作分别用于去除小的噪点和填充小的孔洞。

优化：

1. 使用 as\_strided 的窗口函数可以显著提高处理速度。
2. 对于非对称的结构元素，在膨胀操作中使用结构元素的**反射**可以保持结果的位置和形状一致性。

## **6.2 灰值形态学处理**

灰值形态学处理包括腐蚀、膨胀、开运算和闭运算。这些操作的实现基于滑动窗口原理，使用 NumPy 的 as\_strided 函数来提高效率。

### **6.2.1 灰值腐蚀**

灰值腐蚀的核心实现如下：

def erosion\_gray(img, np\_kernel):  
 k\_rows, k\_cols = np\_kernel.shape  
 pad\_height, pad\_width = k\_rows // 2, k\_cols // 2  
 padded\_img = np.pad(img, ((pad\_height, pad\_height), (pad\_width, pad\_width)), mode='constant', constant\_values=255)  
 window\_view = as\_strided(padded\_img,  
 shape=(img.shape[0], img.shape[1], k\_rows, k\_cols),  
 strides=padded\_img.strides \* 2)  
 mask = np\_kernel == 1  
 eroded = np.min(window\_view[..., mask], axis=-1)  
 return eroded.astype(np.uint8)

实现步骤：

1. 对输入图像进行填充，使用255（最大灰度值）作为填充值。

2. 使用 as\_strided 创建滑动窗口视图。

3. 创建结构元素的掩码。

4. 对每个窗口应用腐蚀操作，取覆盖区域内的最小值。

5. 将结果转换回8位无符号整数类型。

### **6.2.2 灰值膨胀**

灰值膨胀的实现与腐蚀类似，主要区别在于：

def dilation\_gray(img, np\_kernel):  
 # ... [与腐蚀相同的前置代码，但填充值为0] ...  
 np\_kernel\_reflected = np.flip(np\_kernel)  
 mask = np\_kernel\_reflected == 1  
 dilated = np.max(window\_view[..., mask], axis=-1)  
 return dilated.astype(np.uint8)

关键区别：

1. 填充值使用0（最小灰度值）。

2. 使用结构元素的反射。

3. 使用 np.max 而不是 np.min，取覆盖区域内的最大值。

### **6.2.3 开运算和闭运算**

灰值开运算和闭运算是腐蚀和膨胀的组合：

def open\_gray(img, np\_kernel):  
 return dilation\_gray(erosion\_gray(img, np\_kernel), np\_kernel)  
   
def close\_gray(img, np\_kernel):  
 return erosion\_gray(dilation\_gray(img, np\_kernel), np\_kernel)

这两个操作分别用于去除小的亮区域和填充小的暗区域，同时保持整体灰度分布。

优化实现：

1. 使用 as\_strided 创建滑动窗口视图，避免了显式的嵌套循环，显著提高了处理速度。
2. 对于非对称的结构元素，在膨胀操作中使用结构元素的反射可以保持结果的位置和形状一致性。
3. 使用 NumPy 的向量化操作（如 np.min 和 np.max）进行计算，提高了效率。

注意：

1. 填充值的选择：腐蚀操作使用255（最大灰度值）作为填充值，而膨胀操作使用0（最小灰度值）。确保边界处理的正确性。
2. 结构元素的反射：在膨胀操作中使用结构元素的反射，这是为了保持与数学定义的一致性。

## **6.3 灰值形态学-梯度求取（边缘）**

ero\_img = erosion\_gray(img, np\_kernel) # 腐蚀  
 dil\_img = dilation\_gray(img, np\_kernel) # 膨胀  
 new\_img = cv.absdiff(dil\_img, ero\_img) # 取绝对差

直接在腐蚀和膨胀后，取绝对差，得到边缘。

## **6.4 灰值形态学-快速细化（骨架提取）**

本实验实现了两种骨架提取方法：skeletonize 和 medial\_axis。这两种方法都是基于 skimage 库中的形态学操作实现的。

### **6.4.1 实现代码**

if deal\_Type == 0:  
 # 使用skeletonize方法  
 skeleton = morphology.skeletonize(img)  
 new\_img = (skeleton \* 255).astype(np.uint8)  
 method = "skeletonize"  
 elif deal\_Type == 1:  
 # 使用medial\_axis方法  
 skeleton, distance = morphology.medial\_axis(img, return\_distance=True)  
 new\_img = (skeleton \* 255).astype(np.uint8)  
 method = "medial\_axis"

# **七、实验结果与分析**

## **7.1 二值形态学**

处理时间：

| 处理方法 | numpy时间（ms） | opencv时间（ms） |
| --- | --- | --- |
| 腐蚀 | 0.996 | 0.005以内 |
| 膨胀 | 0.996 | 0.005以内 |
| 开运算 | 1.993 | 0.005以内 |
| 闭运算 | 1.993 | 0.005以内 |

处理效果：

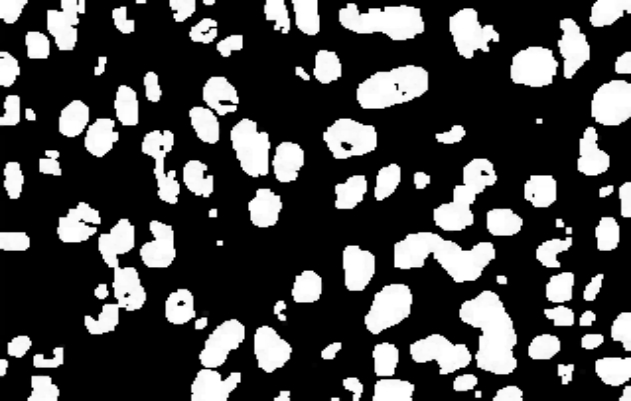
文本

描述已自动生成文本

描述已自动生成**形状

中度可信度描述已自动生成文本

描述已自动生成图15：二值腐蚀结果（3×3和5×5结构元素）**

**图16：二值膨胀结果（3×3和5×5结构元素）**

**图17：二值开运算结果（3×3和5×5结构元素）**

**图18：二值闭运算结果（3×3和5×5结构元素）**卡通人物

中度可信度描述已自动生成卡通人物

中度可信度描述已自动生成

## **7.2 灰值形态学**

处理时间：

| 处理方法 | numpy时间（ms） | opencv时间（ms） |
| --- | --- | --- |
| 腐蚀 | 1.004 | 0.005以内 |
| 膨胀 | 1.005 | 0.005以内 |
| 开运算 | 0.997 | 0.005以内 |
| 闭运算 | 0.997 | 0.005以内 |
| 边缘提取 | 1.598 | 0.005以内 |

黑白色的狗站在草地上

描述已自动生成狗叼着飞盘

描述已自动生成处理效果：

**图19：灰值腐蚀结果（3×3和5×5结构元素）**

女人戴着帽子

描述已自动生成女人戴着帽子

中度可信度描述已自动生成**图20：灰值膨胀结果（3×3和5×5结构元素）**

女人戴着帽子

描述已自动生成女人戴着帽子

中度可信度描述已自动生成**图21：灰值开运算结果（3×3和5×5结构元素）**狗站在地上的马

描述已自动生成狗叼着飞盘

描述已自动生成

**图22：灰值闭运算结果（3×3和5×5结构元素）**

**图23：灰值提取边缘结果（3×3和5×5结构元素）**图片包含 桌子, 食物, 关, 披萨

描述已自动生成男子的脸部特写黑白照

低可信度描述已自动生成

## **7.3 快速细化（提取骨架）**

处理时间：

| 处理方法 | 时间（ms） |
| --- | --- |
| skeletonize | 68.982 |
| medial\_axis | 152.058 |

黑板上的画

描述已自动生成文本

描述已自动生成处理效果：

**图24：细化提取骨架车牌-Skeleton和Medial\_axis（3×3和5×5结构元素）**

**图25：细化提取骨架-Skeleton和Medial\_axis（3×3和5×5结构元素）**男子的脸部特写与配字黑白照

中度可信度描述已自动生成男子的脸部特写与配字黑白照

低可信度描述已自动生成

# **八、实验总结及心得体会**

1. **numpy优化计算**： 使用 NumPy 的 as\_strided 函数来创建**滑动窗口视图**可以显著提高处理速度。比如在二值腐蚀操作中：

* window\_view = as\_strided(padded\_img,  
   shape=(img.shape[0], img.shape[1], k\_rows, k\_cols),  
   strides=padded\_img.strides \* 2)  
  eroded = np.all(window\_view[..., mask] == 255, axis=-1)

1. **形态学操作**： 在灰值形态学中：
   * 膨胀和腐蚀操作不仅影响图像的形状，还会**改变其灰度分布**。
   * 开运算和闭运算的组合可以有效地去除噪声同时保持主要特征。
2. **开闭运算**：
   * 开运算（先腐蚀后膨胀）能够**有效去除小的亮点噪声，同时保持大的亮区域不变**。所以它特别适用于去除图像中的小物体或噪点，同时保持主要特征。
   * 闭运算（先膨胀后腐蚀）则能**填充小的暗区域，连接近邻的亮区域**。所以它非常适合用于填补物体中的小孔或缝隙，以及连接断开的线条。
3. **形态学边缘提取中结构元素尺寸的影响**： 我观察到当增大结构元素的尺寸时，提取的**边缘变得更粗但整体图像更模糊**。
   * 边缘变粗：大尺寸的结构元素在膨胀操作时会**扩展边缘区域**，而在腐蚀操作时会**收缩非边缘区域**。导致边缘和非边缘区域之间的过渡带变宽，所以边缘看起来更粗，更明显。
   * 图像变模糊：大的结构元素平均化更大范围内的像素值。在非边缘区域，这种平均化效应会减少局部的灰度变化，导致细节丢失，使图像整体看起来更模糊。

* 所以在选择结构元素大小时需要在边缘检测的精确性和抗噪声能力之间做出权衡。**小的结构元素可以捕捉更细微的边缘，但也更容易受到噪声影响；大的结构元素能够检测出更显著的边缘并抑制噪声，但可能会丢失一些细节信息。**