目录

基本图像操作方法

孙淼

合肥工业大学计算机与信息学院

摘 要：本文介绍和实现了图像处理中的一些基本算法，并对算法所能达到的效果进行分析；具体内容有：读入一幅图像，然后输出其直方图、统计直方图；将 RGB 模式下的彩色图像转化成灰度图像、黑白图像，以及 HSV 等其他模式的图像，分析其区别；针对读入的图像，对其进行滤波操作（多于 5 种滤波方式），并对滤波后的结果进行分析与比较；对图像进行数学形态学操作，并对各种操作后的效果进行分析和比较。最后分析了上述图像的基本操作都有哪些应用，可以应用于哪些方面；除了上述操作外，还有哪些有效的图像预处理方法，并列出了 2-3 种，解释了其有效性。

关键词：图像处理；直方图；灰度图像；黑白图像；滤波；数学形态学；

1 算法原理

1.1 图形直方图

对于第一个问题“读入一幅图像，然后输出其直方图、统计直方图”，我们首先需要知道什么是图像的直方图、统计直方图。

在数字图像处理中，灰度直方图是最简单且最有用的工具，可以说，对图像的分析与观察，直到形成一个有效的处理方法，都离不开直方图，灰度直方图是灰度级的函数，是对图像中灰度级分布的统计，有两种表示形式。其一是图形表示形式：横坐标表示灰度级，纵坐标表示图像中对应灰度级所出现的像素个数。其二是数组表示形式：数组表示形式，数组的下标表示相应的灰度级，数组的元素表示该灰度下的像素个数。举个例子，如下图 1-1 所示的灰度图，其图形表示形式如图 1-2 所示，其数组表示形式如 1-3 所示。

1	2	3	4	5	6
6	4	3	2	2	1
1	6	6	4	6	6
3	4	5	6	6	6
1	4	6	6	2	3
1	3	6	4	6	6

图 1-1 灰度图

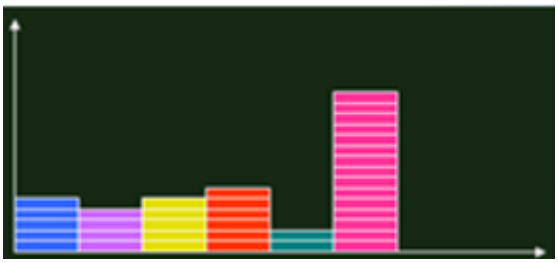

图 1-2 灰度图的图形表示形式

$$h=[5,4,5,6,2,14]$$

图 1-3 灰度图的数组表示形式

对于统计直方图，其是为利用图像的特征描述图像，可借助特征的统计直方图，图像特征的统计直方图实际上是一个一维的离散函数，即：

$$H(k) = nk / N$$

其中 nk 代表图像特征值为 k 的像素总数，N 为图像像素总数。

以上的灰度图直方图生成可以使用 Matlab 实现，在第二部分我将详细介绍算法的实现。

1.2 RGB 图转其他模式图像

要将 RGB 模式下的彩色图像转化成灰度图像、黑白图像，以及 HSV 等其他模式的图像，并分析其区别。我们首先需要知道什么是 RGB 彩色图像、灰度图像、黑白图像、HSV 图像等。颜色模型是用来精确标定和生成各种颜色的一套规则和定义。某种颜色模型所标定的所有颜色就构成了一个颜色空间。

RGB 模型是一种加色模型，用红、绿、蓝三种颜色按不同的比例相加，以产生多种多样的色光。每个分离的取值范围为 0~255，如图 1-4 所示。

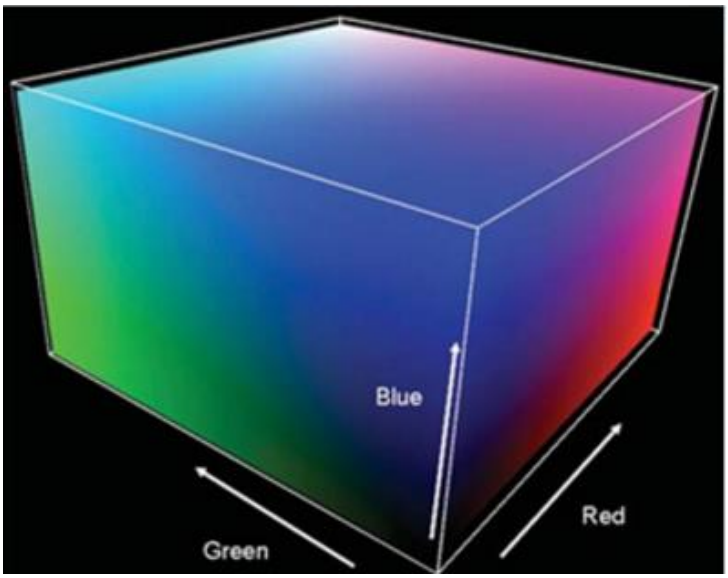

图 1-4 RGB 颜色模型

灰度图像是 R、G、B 三个分量相同的一种特殊的彩色图像；每个像素只需一个字节存放灰度值（又称为强度值、亮度值），范围为 0~255。

常见的灰度化方法有分量法、最大值法、平均值法和加权平均法（根据重要性及其他指标，将三个分量以不同的权值进行加权平均。由于人眼对绿色的敏感最高，对蓝色敏感最低，因此，按照人眼对各个颜色敏感度的权重进行加权平均就能得到较合理的灰度图像。）

对于黑白图像，一般是通过二值化实现，二值化即将图像上的像素点的灰度值设置为 0 或 255，呈现出明显的只有黑和白的视觉效果。二值化有多种方法，在给定阈值的情况下，可以采用全局二值化：对于全局阈值 T，白色 G>T；黑色 G<T；局部二值化：按一定的规则将图像分成 N 个窗口，针对 N 个窗口进行二值化；局部自适应二值化：通过对该窗口的像素的平均值 E,像素值之间的差平方 P,像素之间的均方根值 Q 等各种局部特征，设定一个参数方程进行阈值的计算，例如：T = a*E+b*P+c*Q，其中 a，b，c 是自由参数。这种方法更能表现出二值化图像中的细节。对于二值化的阈值，其选取方法也有很多，如双峰法、P 参数法、迭代法、OTSU 法。

HSV 颜色模型。特点是圆锥的顶面对应于 V=1；H 绕 V 轴的旋转角给定；饱和度 S 取值从 0-1（所以圆锥顶面的半径为 1），在圆锥顶点处，V=0，H、S 无定义，代表黑色；顶面中心处 S=0，V=1，H 无定义，代表白色。从 RGB 转换为 HSV 是有固定的公式的，如下图 1-5 所示。

RGB到HSV(HSB)的转换：

$$h = \begin{cases} \text{undefined} & \text{if } max = min \\ 60° \times \frac{g-b}{max-min} + 0°, & \text{if } max = r \text{ and } g \geq b \\ 60° \times \frac{g-b}{max-min} + 360°, & \text{if } max = r \text{ and } g < b \\ 60° \times \frac{b-r}{max-min} + 120°, & \text{if } max = g \\ 60° \times \frac{r-g}{max-min} + 240°, & \text{if } max = b \end{cases}$$

$$s = \begin{cases} 0, & \text{if } max = 0 \\ \frac{max-min}{max} = 1 - \frac{min}{max}, & \text{otherwise} \end{cases}$$

$$v = max$$

图 1-5 RGB 转换为 HSV 的公式

以上的各种模式图像的生成可以使用 Matlab 实现，在第二部分我将详细介绍算法的实现。

1.3 图像滤波

对于在 1.2 中输出的图像，要对其进行滤波操作（不少于 5 种滤波方式），并对滤波后的结果进行分析与比较。我们就首先要确定常见的滤波操作有哪些。

图像滤波涉及噪声、模板卷积和各种滤波算法。图像滤波的主要目的是为了给图像去噪，也称为图像平滑，是为了抑制图像噪声改善图像质量进行的处理。常见的噪声有椒盐噪声和高斯噪声。常见的滤波操作有：

（1）均值滤波：也称为线性滤波，其采用的主要方法为领域平均法，思想是假设图像由许多灰度恒定的小块组成，相邻的像素之间存在很高的空间相关性，而噪声则是统计独立的。所以可以用领域内各像素的灰度平均值代替像素原来的灰度值，实现图像的平滑。处理每个像素时，给定一个模板，包括该像素及其周围若干临近像素，将模板中全体像素的均值来代替原来的像素值，如 3*3 的平均算子如图 1-6 所示，加权滤波算子的一种如图 1-7 所示。

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

图 1-6 3*3 平均算子

$$H_1 = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

图 1-7 加权均值的一种滤波算子

（2）高斯滤波：一种线性平滑滤波，被认为是对图像平滑处理是最优的，适用于消除高斯噪声。高斯算子模板通过高斯关系式来设置：

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

在此基础上，我们的 5*5 高斯平均算子的模板如下图 1-8 所示：

0.002	0.013	0.220	0.013	0.002
0.013	0.060	0.098	0.060	0.013
0.220	0.098	0.162	0.098	0.220
0.013	0.060	0.098	0.060	0.013
0.002	0.013	0.220	0.013	0.002

图 1-8 5*5 高斯平均算子

（3）中值滤波：对于待处理的像素，让它与周围的像素一起组成一个模板，对模板中的像素值从小到大排列，取排列在最中间的灰度值作为待处理像素的灰度值，原理如下图 1-9 所示。

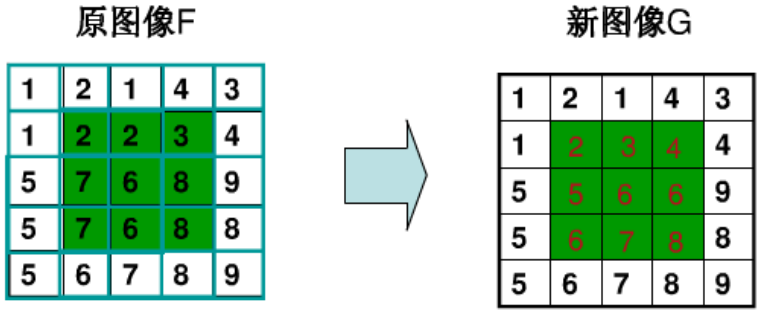

图 1-9 中值滤波原理

（4）KNN 平滑滤波：根据前面的处理结果可知，经过平滑（特别是均值）滤波处理之后，图像就会变得模糊。分析原因，在图像上的景物之所以可以辨认清楚，是因为目标物之间存在边界，而平滑处理则在一定程度上模糊了边界。所以设置边界保持滤波器时，我们的设计思想是首先判别当前像素是否为边界上的点，如何是，则不进行处理，如果不是，则进行平滑处理。

如此，我们可以提出 K 近邻（KNN）平滑滤波器，也就是其在进行平滑处理时，先判断当前像素是否为边界点，若是，则不进行平滑处理，若不是，则进行平滑处理。思想：在一个与待处理非边界像素邻近的范围内，寻找其像素值与之最接近的 K 个邻点，将该 K 个邻点的中值（均值）替换原像素值，使平滑运算对边界点的影响不大，这样就可以保持原有的灰度特效。如下图 1-10 所示，点是黄色区域的非边界点，点 2 是蓝色区域的边界点。

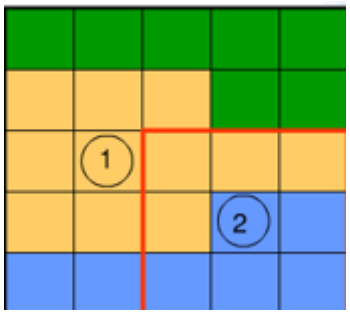

图 1-10 KNN 平滑滤波器

我们以待处理像素为中心，作一个 m*m 的作用模板，在模板中，选择 k 个与待处理像素灰度差为最小的像素（不包括当前像素本身），然后将这 k 个像素的灰度均值（中值）替换原来的像素值，由此可获得 KNN 均值滤波和 KNN 中值滤波的结果，如下图 1-11 所示。

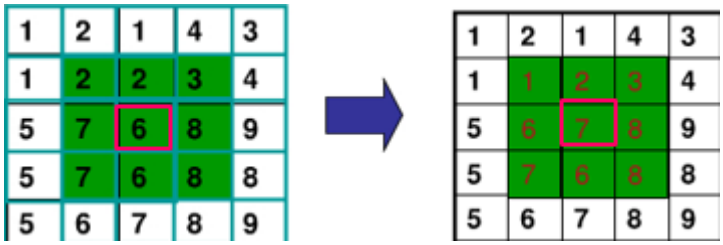

图 1-11 3*3 模板 k = 5 的中值滤波

KNN 滤波器因为有了边界保持的作用，所以在去除椒盐以及高斯噪声时，对图像景物的清晰度保持方面的效果非常明显，同时也付出了算法复杂度增加的代价。

（5）其他滤波：除此之外，还有许多滤波器如对称近邻平滑滤波器、最小方差平滑滤波器、Sigma 平滑滤波器，双边滤波器、维纳滤波器等。

对于上述滤波方法，简单易于实现的滤波操作，我们采用 C++实现，对于原理复杂的操作，我们将借助 Matlab 已有的函数进行实现。

1.4 数学形态学操作分析

要对图像进行数学形态学操作，并对各种操作后的效果进行分析和比较，就首先要明白什么是数学形态学操作。

数学形态学是根据形态学概念发展而来的具有严格数学理论基础的科学，并在图像处理和模式识别领域得到了成功应用。其基本思想是用具有一定形态的结构元素去度量和提取图像中的对应形状以达到对图像分析和识别的目的。一种特殊定义的邻域称之为“结构元素”，在每个像素位置上它与二值图像对应的区域进行特定的逻辑运算，逻辑运算的结果为输出图像的相应元素。形态学运算的效果取决于结构元素的大小，内容以及逻辑运算的性质。

常见的二值形态学基本运算是为了简化图像数据，保持它们的基本的形状特性，并除去不相干的结构，基本的运算包括：膨胀、腐蚀、开闭、击中与否变换、细化、粗化、骨架，在此我们对前三种进行实现。

（1）膨胀：使图像扩大；定义为：

$$A \oplus B = \left\{ x \mid (\hat{B})_x \cap A \neq \Phi \right\}$$

上式表示：B 的反射进行平移与 A 的交集不能为空。对 B 的反射进行移位，以便它能滑过集合（图像）A。含义是：每当结构元素在目标图像 A 上平移后，结构元素 与其覆盖的子图像中至少有一个元素相交时，就将目标图像中与结构元素的原点对应的那个位置的像素值置为“1”，否则置为 0。注意当结构元素中原点位置的值是 0 时，仍把它看

作是 0；而不再把它看作是 1。当结构元素在目标图像上平移时，允许结构元素中的非原点像素超出目标图像范围。

（2）腐蚀：使图像缩小，定义为：

$$A\Theta B=\{x\mid(B)_x\subseteq A\}$$

B 移动后完全包含在 A 中时，B 的原点位置的集合。含义是每当在目标图像 A 中找到一个与结构元素 B 相同的子图像时，就把该子图像中与 B 的原点位置对应的那个像素位置标注为 1，图像 A 上标注出的所有这样的像素组成的集合，即为腐蚀运算的结果。腐蚀运算的实质就是在目标图像中标出那些与结构元素相同的子图像的原点位置的像素。

（3）开&闭运算：

开运算：B 对 A 进行的开操作就是先用 B 对 A 腐蚀，然后用 B 对结果进行膨胀

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

闭运算：B 对 A 进行的闭操作就是先用 B 对 A 膨胀，然后用 B 对结果进行腐蚀

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

2 算法设计

2.1 直方图、统计直方图的算法

对于图像的直方图和统计直方图的生成，我们将借助 MATLAB 实现。

1）根据第一部分算法原理的分析，对于直方图，我们只需要将灰度图的所有像素点值读取，然后在一个横轴为 0-255 的直方图上将读取的结果统计上去即可，代码如下：

```
grayimage=imread('lena_Rgb2Gray_weighting.jpg');
subplot(2,2,1),imshow(grayimage),title('原始图像');
[m,n]=size(grayimage);
gp=zeros(1,256);
for i=1:256
    gp(i)=length(find(grayimage==(i-1)));
end
subplot(2,2,2),bar(0:255,gp);
```

2）对于统计直方图，只需要在后面除以(m*n)即可，代码如下：

```
grayimage=imread('lena_Rgb2Gray_weighting.jpg');
subplot(2,2,1),imshow(grayimage),title('原始图像');
[m,n]=size(grayimage);
```

```
gp=zeros(1,256);
for i=1:256
    gp(i)=length(find(grayimage==(i-1)))/(m*n);% /(m*n)使之成为统计直方图
end
subplot(2,2,2),bar(0:255,gp);
```

2.2 RGB 图转其他模式图像的算法

对于 RGB 图转其他模式图像的实现，我们将借助 MATLAB 实现。

常用的灰度化方法有平均值法、最大值法和加权平均法。根据第一部分算法原理的分析，对于平均值灰度化法，代码如下：

```
Img=imread('Lena.jpg');
subplot(2,2,1),imshow(Img),title('原始图像');
[n,m,a]=size(Img);%判断图像的大小
GrayImage=zeros(n,m);

for x=1:n%通过双循环对图像进行灰度化处理
    for y=1:m
    GrayImage(x,y)=(Img(x,y,1)+Img(x,y,2)+Img(x,y,3))/3; %均值实现灰度化
    end
end
subplot(2,2,2),imshow(uint8(GrayImage)),title('均值灰度化');
```

对于最大值灰度化法，代码如下：

```
Img=imread('Lena.jpg');
subplot(2,2,1),imshow(Img),title('原始图像');
[n,m,a]=size(Img);%判断图像的大小
GrayImage=zeros(n,m);

for x=1:n%通过双循环对图像进行灰度化处理
    for y=1:m
    GrayImage(x,y)=max(Img(x,y,1),max(Img(x,y,2),Img(x,y,3))); %最大值实现灰度化
    end
end
subplot(2,2,2),imshow(uint8(GrayImage)),title('最大值灰度化');
```

对于加权灰度值法，代码如下：

```
Img=imread('Lena.jpg');
subplot(2,2,1),imshow(Img),title('原始图像');
[n,m,a]=size(Img);%判断图像的大小
```

```
GrayImage=zeros(n,m);

for x=1:n%通过双循环对图像进行灰度化处理
    for y=1:m
    GrayImage(x,y)=0.3*Img(x,y,1)+0.59*Img(x,y,2)+0.11*Img(x,y,3);%加权实现灰度化
    end
end
subplot(2,2,2),imshow(uint8(GrayImage)),title('加权灰度化');
```

对于黑白图像，一般是通过二值化实现，二值化即将图像上的像素点的灰度值设置为 0 或 255，呈现出明显的只有黑和白的视觉效果。二值化有多种方法，在给定阈值的情况下，可以采用全局二值化：对于全局阈值 T，白色 G>T；黑色 G<T；对于二值化的阈值，其选取方法也有很多，如双峰法、P 参数法、迭代法、OTSU 法。在此我们使用 MATLAB 自带的方法找到最佳阈值，然后通过二值转化函数得到结果，代码如下：

```
Img=imread('Lena.jpg');
subplot(2,2,1),imshow(Img),title('原始图像');
[n,m]=size(Img);%判断图像的大小
thresh = graythresh(Img);      %自动确定二值化阈值
Img_imbinarize = im2bw(Img,thresh);        %对图像二值化
subplot(2,2,2),imshow(Img_imbinarize),title('调用系统函数实现二值化黑白图像');
```

从 RGB 转换为 HSV 是有固定的公式的，根据第一部分算法原理的分析，对于将 RGB 图像转化为 HSV 图像，代码如下：

```
Img=imread('Lena.jpg');
subplot(2,2,1),imshow(Img),title('原始图像');
Img_hsv = rgb2hsv(Img);        %rgb 转 HSV
subplot(2,2,2),imshow(Img_hsv),title('调用系统函数实现二值化黑白图像');
```

为了适当增加难度，我们知道 HSV 颜色平面是指的色相，饱和度和值，那么我们可以将 S 通道乘以比例因子来增加图像的饱和度。然后再将已处理的 HSV 图像转换回 RGB 颜色空间，显示新的 RGB 图像，处理后的图像中的颜色应该更加鲜艳，代码如下：

```
Img=imread('Lena.jpg');
subplot(2,2,1),imshow(Img),title('原始图像');
Img_hsv = rgb2hsv(Img);        %rgb 转 HSV
 [h,s,v] = imsplit(Img_hsv);
 saturationFactor = 2; %S 通道乘以比例因子 2
 s_temp = s * saturationFactor;
```

```
hsv_temp = cat(3,h,s_temp,v);
rgb_new = hsv2rgb(hsv_temp);
subplot(2,2,2),imshow(rgb_new),title('饱和度增强的 rgb 图');
```

2.3 图像滤波的算法

要完成这一部分，首先要对 Lena. jpg 添加噪声，主要有两种：椒盐噪声和高斯噪声。椒盐噪声，通常是由图像传感器，传输信道，解压处理等产生的黑白相间的亮暗点噪声（椒-黑，盐-白），通常出现在灰度图中；高斯噪声，顾名思义是指服从高斯分布（正态分布）的一类噪声，通常是因为不良照明和高温引起的传感器噪声。通常在 RGB 图像中，显现比较明显。

添加椒盐噪声代码如下：

```
Img=imread('Lena.jpg');
Img_gray = rgb2gray(Img);
subplot(2,2,1),imshow(Img_gray),title('原始图像');
Img_salt=imnoise(Img_gray ,'salt & pepper');
subplot(2,2,2),imshow(Img_salt),title('增加椒盐噪声的图像');
```

添加高斯噪声代码如下：

```
Img=imread('Lena.jpg');
Img_gray = rgb2gray(Img);
subplot(2,2,1),imshow(Img_gray),title('原始图像');
Img_guassian=imnoise(Img_gray ,'gaussian');
subplot(2,2,2),imshow(Img_guassian),title('增加高斯噪声的图像');
```

对于 Lena 的每张添加了上述两种噪声的图像，我们都进行滤波处理。

代码如下所示，我对添加了两种不同噪声的图像都进行高斯滤波：

```
Img=imread('Lena.jpg');
Img_gray = rgb2gray(Img);
Img_add_guassian=imnoise(Img_gray ,'gaussian');
f1= fspecial('gaussian',[5,5],1);
Img_f1 = imfilter(Img_add_guassian,f1,'replicate');
subplot(2,2,1),imshow(Img_f1),title('增加高斯噪声的图像高斯滤波处理后');
Img_add_salt=imnoise(Img_gray ,'salt & pepper');
Img_f2 = imfilter(Img_add_salt,f1,'replicate');
subplot(2,2,2),imshow(Img_f2),title('增加椒盐噪声的高斯滤波处理后');
```

我对添加了两种不同噪声的图像都进行中值滤波，代码如下所示：

```
Img=imread('Lena.jpg');
Img_gray = rgb2gray(Img);
Img_add_guassian=imnoise(Img_gray ,'gaussian');
Img_f1=medfilt2(Img_add_guassian,[3,3]);%进行 3*3 中值滤波;
subplot(2,2,1),imshow(Img_f1),title('增加高斯噪声的图像中值滤波处理后');
Img_add_salt=imnoise(Img_gray ,'salt & pepper');
Img_f2=medfilt2(Img_add_salt,[3,3]);%进行 3*3 中值滤波;
subplot(2,2,2),imshow(Img_f2),title('增加椒盐噪声的中值滤波处理后');
```

我对添加了两种不同噪声的图像都进行均值滤波，代码如下所示：

```
Img=imread('Lena.jpg');
Img_gray = rgb2gray(Img);
Img_add_guassian=imnoise(Img_gray ,'gaussian');
f_1=fspecial('average',[3,3]);%3*3 均值滤波
Img_f1 = imfilter(Img_add_guassian,f_1);
subplot(2,2,1),imshow(Img_f1),title('增加高斯噪声的图像均值滤波处理后');

Img_add_salt=imnoise(Img_gray ,'salt & pepper');
f_1=fspecial('average',[3,3]);%3*3 均值滤波
Img_f1 = imfilter(Img_add_salt,f_1);
subplot(2,2,2),imshow(Img_f2),title('增加椒盐噪声的均值滤波处理后');
```

我对添加了两种不同噪声的图像都进行维纳滤波（二维自适应除噪滤波器 wiener2 函数），代码如下所示：

```
Img=imread('Lena.jpg');
Img_gray = rgb2gray(Img);
Img_add_guassian=imnoise(Img_gray ,'gaussian');
Img_f1=wiener2(Img_add_guassian,[3 3]); %对加噪图像进行二维自适应维纳滤波
subplot(2,2,1),imshow(Img_f1),title('增加高斯噪声的图像均值滤波处理后');

Img_add_salt=imnoise(Img_gray ,'salt & pepper');
Img_f2 = wiener2(Img_add_salt,[3 3]); %对加噪图像进行二维自适应维纳滤波
subplot(2,2,2),imshow(Img_f2),title('增加椒盐噪声的均值滤波处理后');
```

我对添加了两种不同噪声的图像都进行对比度增强滤波，代码如下所示：

```
Img=imread('Lena.jpg');
Img_gray = rgb2gray(Img);
Img_add_guassian=imnoise(Img_gray ,'gaussian');
f_1=fspecial('unsharp',0.2);%对比度增强滤波器
Img_f1 = imfilter(Img_add_guassian,f_1);
```

```
6.  subplot(2,2,1),imshow(Img_f1),title('增加高斯噪声的图像对比度增强滤波处理后');
7.
8.  Img_add_salt=imnoise(Img_gray ,'salt & pepper');
9.  f_1=fspecial('unsharp',0.2);%对比度增强滤波器
10. Img_f1 = imfilter(Img_add_salt,f_1);
11. subplot(2,2,2),imshow(Img_f2),title('增加椒盐噪声的对比度增强滤波处理后');
```

2.4 数学形态学操作分析的算法

常见的二值形态学基本运算是为了简化图像数据，保持它们的基本的形状特性，并除去不相干的结构，基本的运算包括：膨胀、腐蚀、开闭、击中与否变换、细化、粗化、骨架，在此我们对前三种进行实现。

我对两种典型的需要膨胀的图像都进行了膨胀处理，其中第二个由于一次处理效果不明显，我对其进行了三次连续膨胀，代码如下所示：

```
1.  Img_1=imread('expand_1.jpg');
2.  B=[0 1 0
3.  1 1 1
4.  0 1 0];
5.  expand_1=imdilate(Img_1,B);%图像 1 被结构元素 B 膨胀
6.  subplot(2,2,1),imshow(Img_1),title('expand_1');
7.  subplot(2,2,2),imshow(expand_1),title('expand_1 膨胀一次的结果');
8.
9.  Img_2=imread('expand_2.jpg');
10. B=[0 1 0
11. 1 1 1
12. 0 1 0];
13. expand_1=imdilate(Img_2,B);%图像 1 被结构元素 B 膨胀
14. expand_2=imdilate(expand_1,B);%图像 1 被结构元素 B 膨胀
15. expand_3=imdilate(expand_2,B);%图像 1 被结构元素 B 膨胀
16. subplot(2,2,3),imshow(Img_2),title('expand_2');
17. subplot(2,2,4),imshow(expand_3),title('expand_2 膨胀三次的结果');
```

我对两种典型的需要腐蚀处理的图像都进行了膨胀处理，其中第二个由于一次处理效果不明显，我对其进行了三次连续膨胀，代码如下所示：

```
1.  Img_1=imread('expand_2.jpg');
2.  %strel 函数的功能是运用各种形状和大小构造结构元素
3.   se1=strel('disk',5);%这里是创建一个半径为 5 的平坦型圆盘结构元素
4.   %se=strel('square',5');%方型结构元素
5.  corrosion_1 = imerode(Img_1,se1);
6.  subplot(2,2,1),imshow(Img_1),title('expand_2');
```

```
subplot(2,2,2),imshow(corrosion_1),title('expand_2 腐蚀一次的结果');

Img_2=imread('corrosion.jpg');
se1=strel('disk',5);%这里是创建一个半径为 5 的平坦型圆盘结构元素
corrosion_1 = imerode(Img_2,se1);
corrosion_2 = imerode(corrosion_1,se1);
subplot(2,2,3),imshow(Img_2),title('corrosion_1');
subplot(2,2,4),imshow(corrosion_2),title('corrosion_1 腐蚀二次的结果');
```

我对两种典型的需要开处理的图像都进行了开处理，代码如下所示：

```
Img=imread('open.jpg');
%se=strel('square',5');%方型结构元素
se=strel('disk',5');%圆盘型结构元素
subplot(2,2,1),imshow(Img),title('开运算原始图像');
Img_open=imopen(Img,se);%直接开运算
subplot(2,2,2),imshow(Img_open),title('开运算后图像');

Img=imread('open_and_close.jpg');
%se=strel('square',5');%方型结构元素
se=strel('disk',5');%圆盘型结构元素
subplot(2,2,3),imshow(Img),title('开运算原始图像');
Img_open=imopen(Img,se);%直接开运算
subplot(2,2,4),imshow(Img_open),title('开运算后图像');
```

我对两种典型的需要闭处理的图像都进行了闭处理，代码如下所示：

```
Img=imread('close.jpg');
%se=strel('square',5');%方型结构元素
se=strel('disk',5');%圆盘型结构元素
subplot(2,2,1),imshow(Img),title('闭运算原始图像');
Img_close=imclose(Img,se);%直接开运算
subplot(2,2,2),imshow(Img_close),title('闭运算原始图像');

Img=imread('open_and_close.jpg');
%se=strel('square',5');%方型结构元素
se=strel('disk',5');%圆盘型结构元素
subplot(2,2,3),imshow(Img),title('闭运算原始图像');
Img_close=imclose(Img,se);%直接开运算
subplot(2,2,4),imshow(Img_close),title('闭运算原始图像');
```

3 实验结果

3.1 实验数据

对于得到直方图和统计直方图的部分，由于这两个图都是对灰度图得到的，所以我的实验数据是 lena_Rgb2Gray_weighting.jpg，也就是 Lena 女神的加权灰度图，如下：

图 3-1 Lena 加权灰度图

对于 rgb 图转 hsv 图，我的实验数据是 Lena 的彩图 Lena.jpg，如下：

图 3-2 Lena 彩图

对于第三部分滤波处理，用的也是 Lena 的灰度图。

对于第四部分数学形态学操作，膨胀、腐蚀、开、闭操作部分，我的实验数据是：

图 3-3 膨胀操作实验数据 1

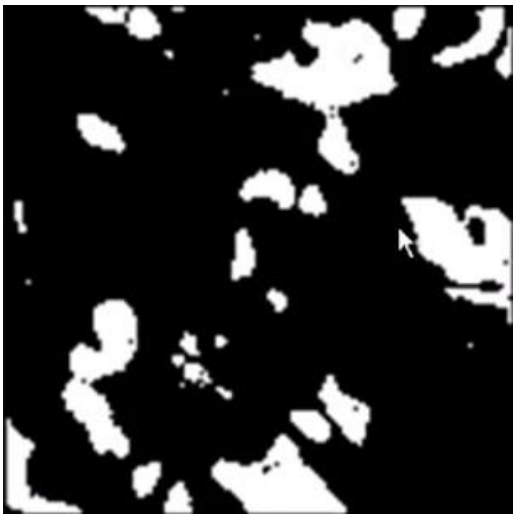

图 3-4 膨胀操作实验数据 2

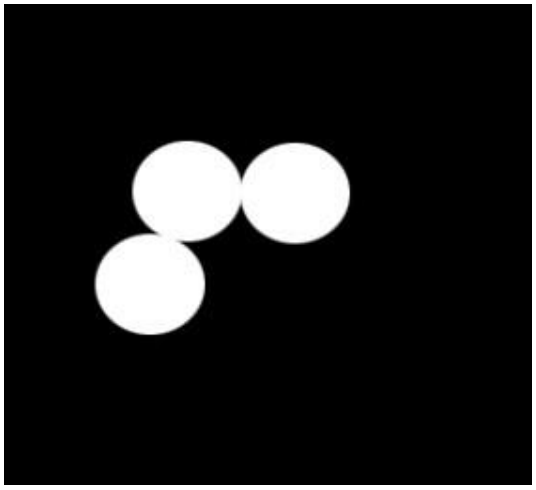

图 3-5 腐蚀操作实验数据 3

图 3-6 开操作实验数据

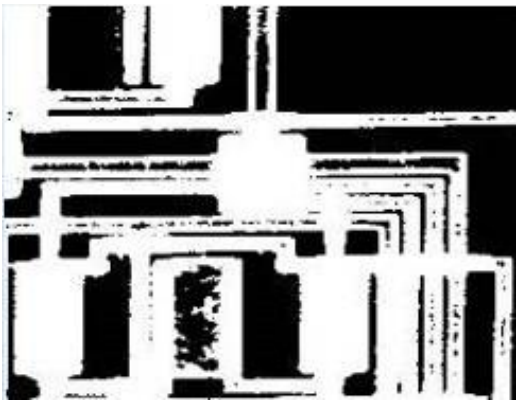

图 3-7 闭操作实验数据

图 3-8 闭合操作公共实验数据

3.2 实验结果

第二部分“算法设计”部分的所有代码，运行结果依次如下：

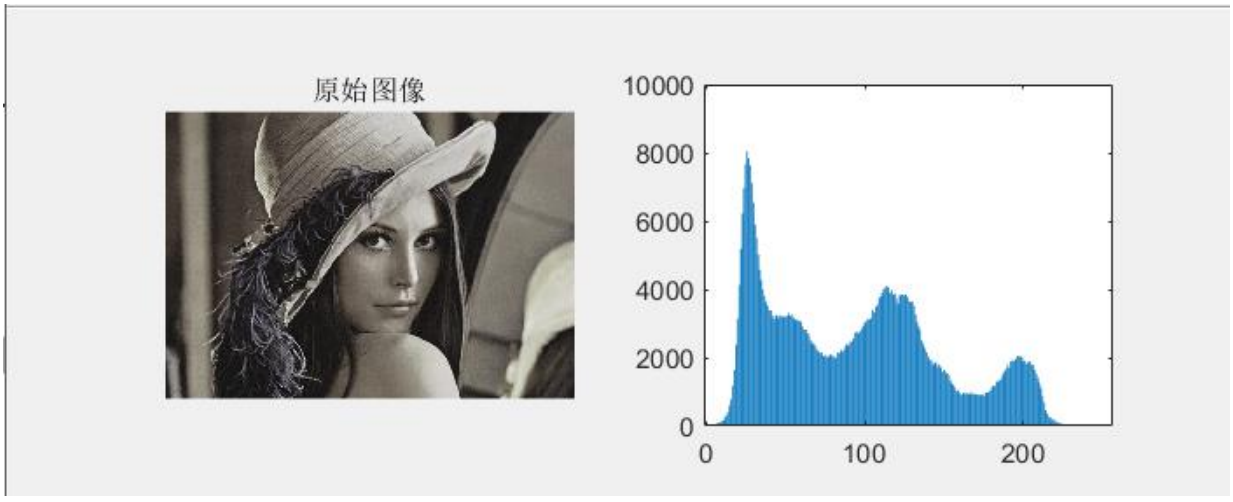

图 3-9 灰度直方图

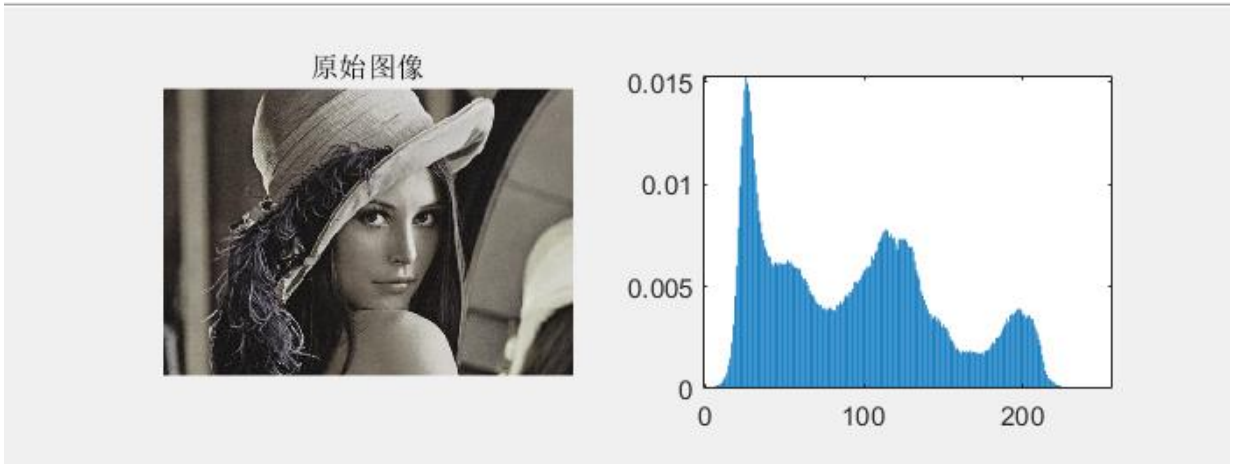

图 3-10 统计灰度直方图

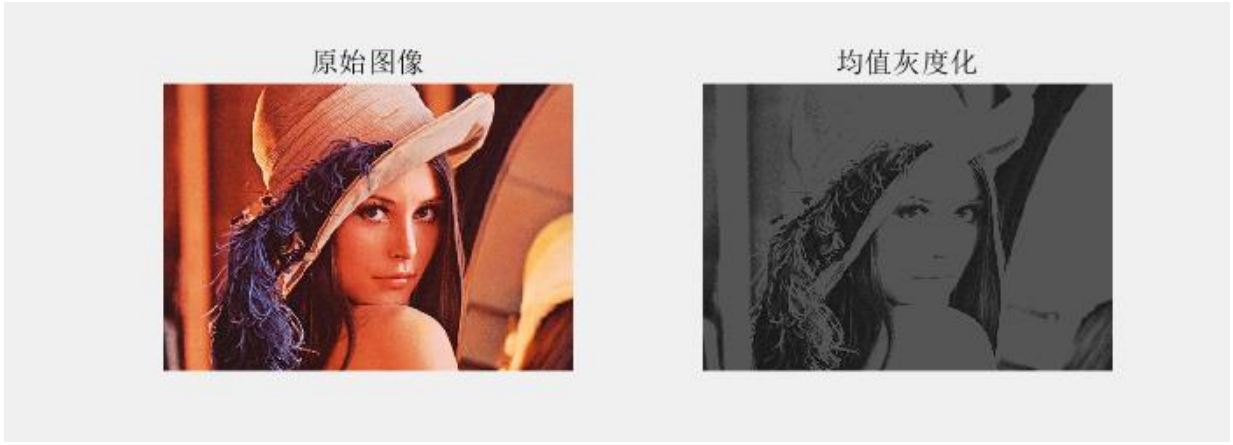

图 3-11 均值灰度化

图 3-12 最大值灰度化

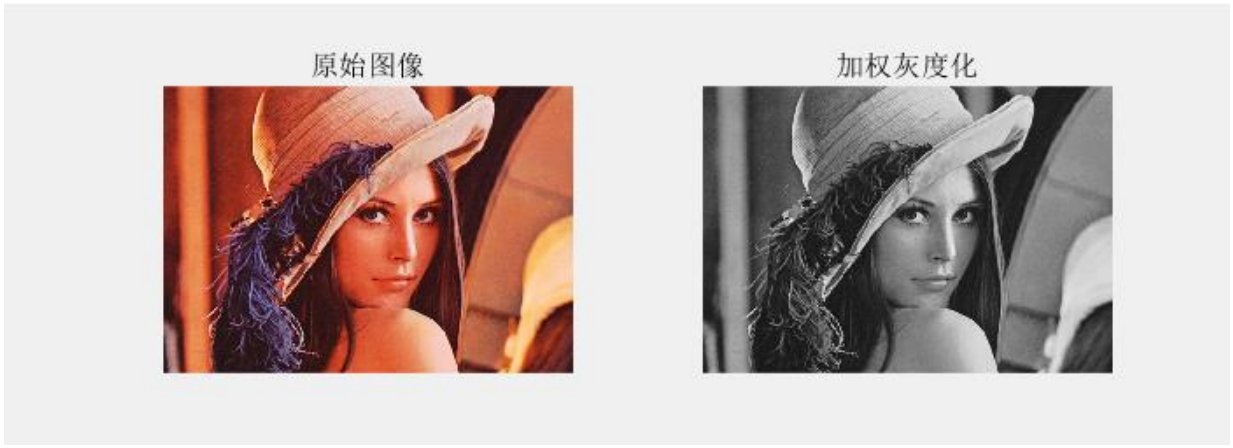

图 3-13 加权灰度化

图 3-14 二值化黑白图像

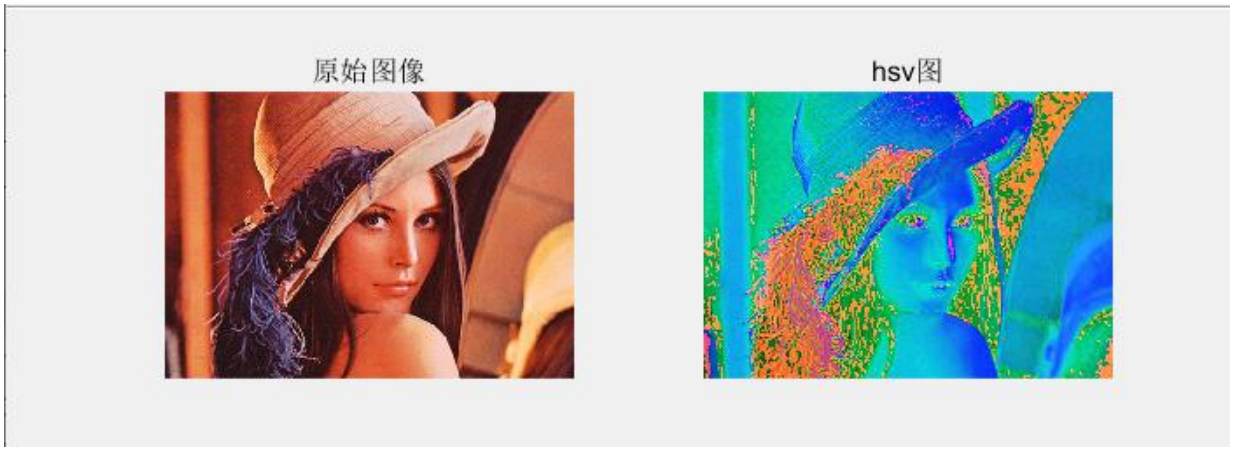

图 3-15 RGB 图转 HSV 图

图 3-16 RGB 转 HSC 图后，S（饱和度）值加倍后转回 RGB 图

图 3-17 添加椒盐噪声图

图 3-18 添加高斯噪声图

图 3-19 两种噪声图像进行高斯滤波处理的结果

图 3-20 两种噪声图像进行中值滤波处理的结果

图 3-21 两种噪声图像进行均值滤波处理的结果

图 3-22 两种噪声图像进行维纳滤波处理的结果

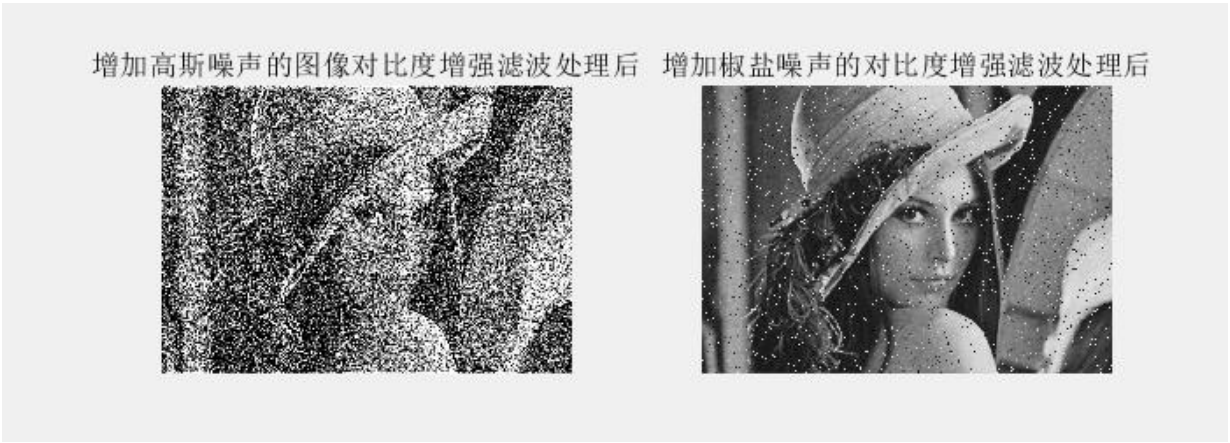

图 3-23 两种噪声图像进行对比度增强滤波处理后

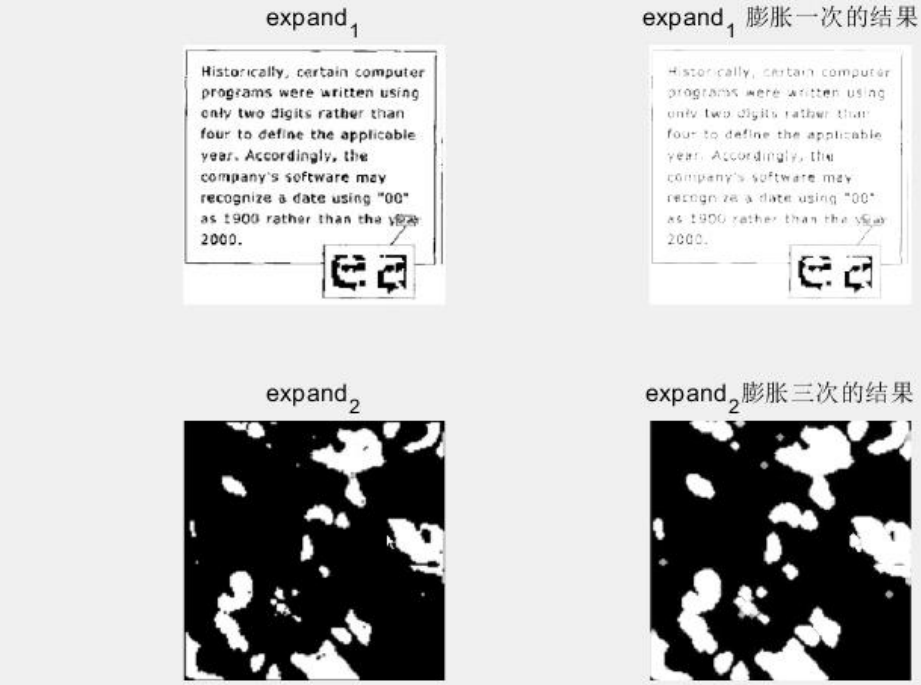

图 3-24 两种典型图像的膨胀处理结果

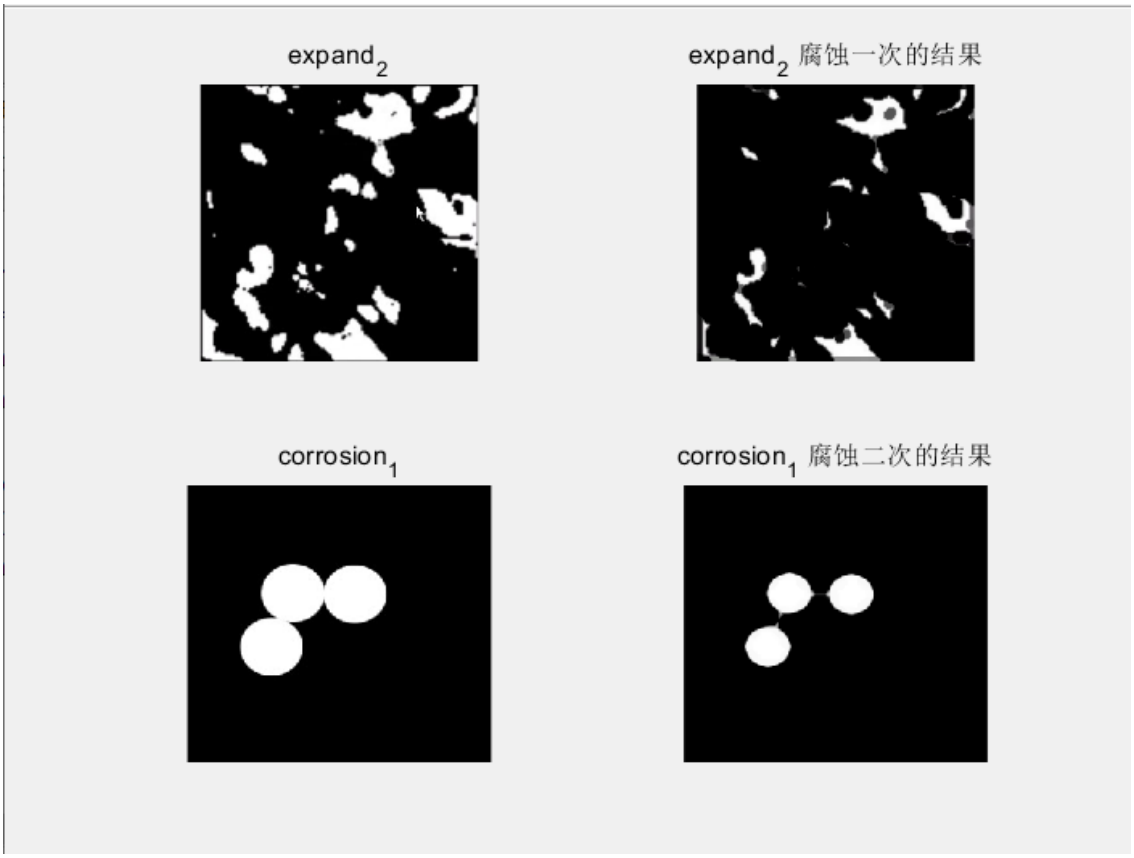

图 3-25 两种典型图像的腐蚀处理结果

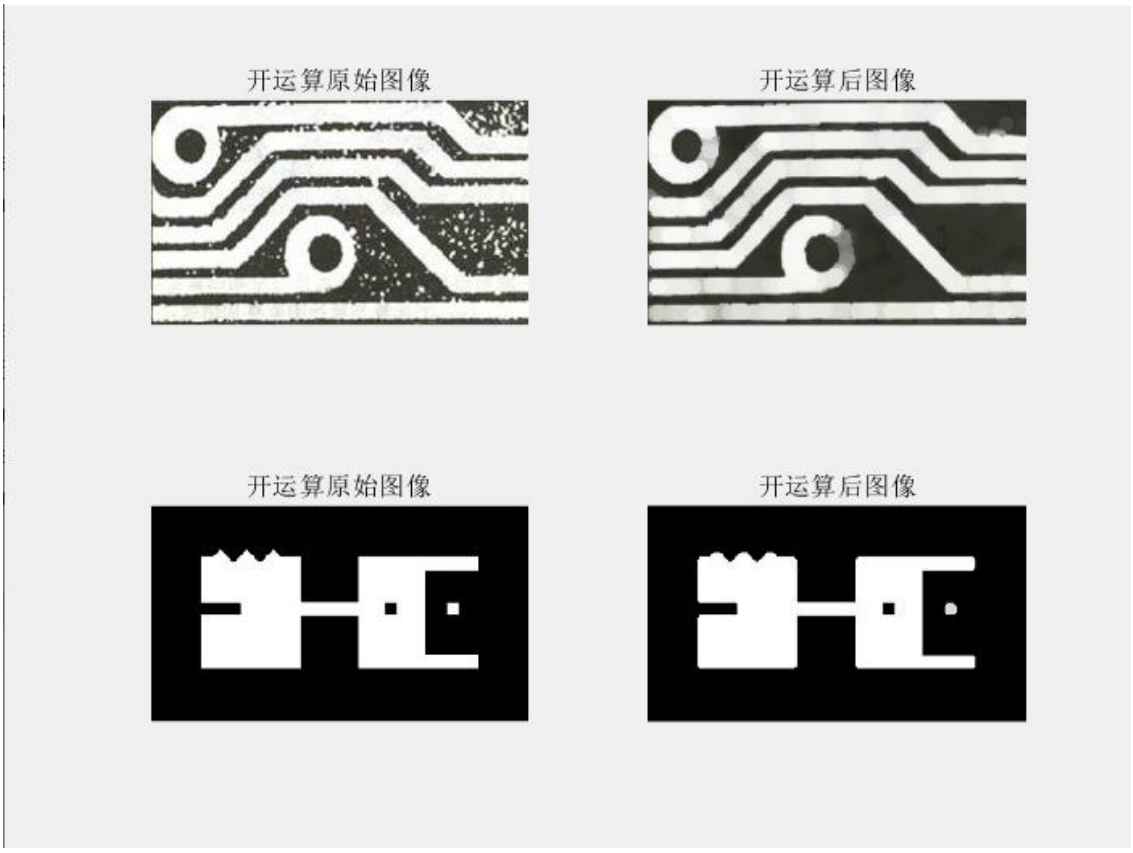

图 3-26 两种典型图像的开运算处理结果

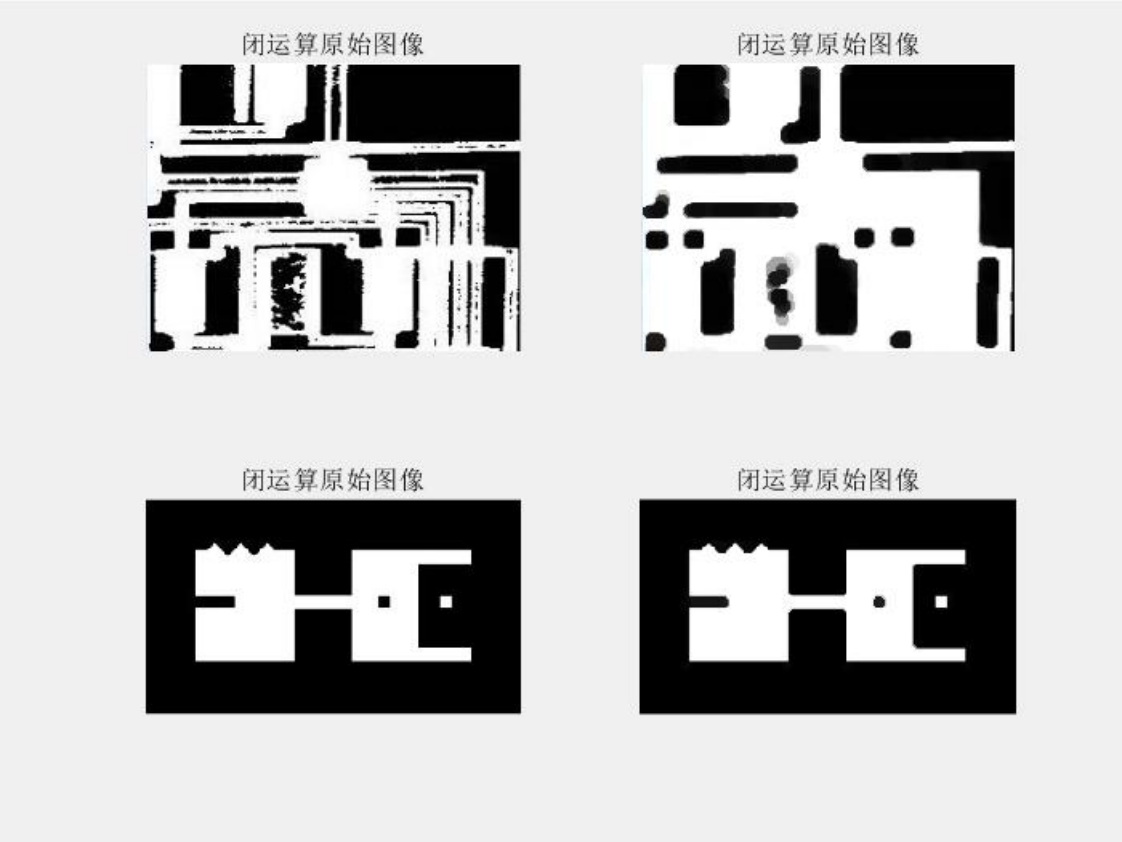

图 3-27 两种典型图像的闭运算处理结果

4 结论与思考

4.1 直方图、统计直方图的结论

灰度直方图是关于灰度级分布的函数，是对图像中灰度级分布的统计。灰度直方图是将数字图像中的所有像素，按照灰度值的大小，统计其出现的频率。灰度直方图是灰度级的函数，它表示图像中具有某种灰度级的像素的个数，反映了图像中某种灰度出现的频率。将纵坐标归一化到[0,1]区间内，也就是将灰度级出现的频率（像素个数）除以图像中的像素的总数，得到灰度统计直方图。

4.2 RGB 图转其他模式图像的结论

RGB 彩色图像中，一种彩色由 R（红色），G（绿色），B（蓝色）三原色按比例混合而成。图像的基本单元是一个像素，一个像素需要 3 块表示，分别代表 R，G，B，如果 8 为表示一个颜色，就由 0-255 区分不同亮度的某种原色。实际中数都是二进制形式的，并且未必按照 R，G，B 顺序，比如 OpenCV 是按照 B,G,R 顺序将三个色值保存在 3 个连续的字节里灰度图像是用不同饱和度的黑色来表示每个图像点，比如用 8 位 0-255 数字表示“灰色”程度，每个像素点只需要一个灰度值，8 位即可，这样一个 3X3 的灰度图，只需要 9 个 byte 就能保存。

4.3 图像滤波的的结论

对于滤波操作，我的分析如下：

- 均值滤波对椒盐噪声的滤波效果不太好，这是因为椒盐噪声的幅值为不变的常数，难以通过求平均值的方法得以完全消除，而噪声部分只是被弱化到周围的像素，使噪声幅值有所下降，对高斯噪声的滤波效果较好，这是因为高斯噪声的幅值有正有负，通过求平均值可以起到抵消作用。均值滤波的缺点也很明显，会引起图像模糊，减少图像中的细节。模板的尺寸可以是 3*3，5*5，7*7…尺寸越大，保留的细节越少。
- 高斯滤波与均值滤波相比，在去除噪声的同时可以保留更多的特征。尺寸越大的高斯滤波算子去除的细节越多，同时损失的特征也越多。
- 中值滤波对于噪声很大的像素，通过用相邻像素灰度替代该像素灰度的方式，达到消除噪声的目的。
- 维纳滤波对高斯噪声的抑制效果更好，但容易失去边缘信息；维纳滤波对椒盐噪声的去噪效果不如其他几种滤波方法。

对于对不同噪声的去噪效果，我的分析如下：

- 对于椒盐噪声，中值滤波效果比均值滤波效果好：椒盐噪声幅值近似相等且随机分布在不同位置上，图像有干净点也有污染点，中值滤波是选择相邻的干净点的值来代替污染点的值，所以处理效果较好。而因为椒盐噪声的均值不为 0，所以均值滤波不能很好的去除噪声点。
- 对于高斯噪声，均值滤波效果比中值滤波效果好，高斯噪声是幅值近似正态分布，但分布在每个像素上，因为图像中的每点都是污染点，所以中值滤波选不到合适的干净点。因为正态分布的均值为 0，因此根据统计数学知识，均值可以抑制噪声。

4.4 数学形态学操作分析的结论

- 可以看出，膨胀的作用是：

  可以将断裂开的目标物合并为一个整体；

  可以填补图像上物体中的细小空洞。
- 可以看出，腐蚀的作用是：

  可以将粘连在一起的不同目标物体分离；

  可以滤掉小的颗粒噪声；
- 可以看出，开操作的作用是：

使轮廓平滑，抑制物体边界的小离散点或尖峰；

用来消除小物体、在纤细点处分离物体、平滑较大物体的边界的同时并不明显改变其面积。

- 可以看出，闭操作的作用是：

用来填充物体内的细小空洞、连接邻近物体、平滑其边界的同时并不明显改变其面积。

4.5 思考题

- 你认为上述图像的基本操作都有哪些应用，可以应用于哪些方面？

1）灰度直方图：图像的灰度直方图只能反映图像的灰度分布情况，反映数字图像中每一灰度级与其出现频率间的关系，但它能描述该图像的概貌。其给出了一个简单可见的指示，用来判断一幅图像是否合理的利用了全部被允许的灰度级范围；

2）灰度化：灰度图像每个像素只需一个字节存放灰度值（又称强度值、亮度值），灰度范围为 0-255，灰度图像通常在单个电磁波频谱（如可见光）内测量每个像素的亮度得到的。用于显示的灰度图像通常用每个采样像素 8 位的非线性尺度来保存，这样可以有 256 级灰度。这种精度刚刚能够避免可见的条带失真，并且非常易于编程。

3）二值化：有利于做图像处理判别。举个实际应用的例子：二值化就是就是将一幅图像的所有像素点按照 256 灰阶分类，每个像素点表示一个灰阶，然后我们将高于某一灰阶像素全部显示成白色，低于某一灰阶的像素点显示成黑色。这样就完成了对一幅图像二值化处理。在实际应用中，例如说金属表面，良品在照明下显示为灰阶 125，而有瑕疵的产品会产生低于灰阶 125 的像素点，假设产生的是 70，然后你设定你的判别标准时 100（以上为白色，以下为黑色），这时瑕疵显示的是黑色，你测量你得到区域内图像的黑色面积，瑕疵产品的话产生的黑色面积就较大，以此你可以判别这个产品是否是次品还是良品。

4）HSV 颜色模型：HSV 在用于指定颜色分割时，有比较大的作用。H 和 S 分量代表了色彩信息。分割应用：用 H 和 S 分量来表示颜色距离，颜色距离指代表两种颜色之间的数值差异。有人通过实验对 HSV 颜色空间进行了大致划分，亮度大于 75%并且饱和度大于 20%为亮彩色区域，亮度小于 25%为黑色区域，亮度大于 75%并且饱和度小于 20%为白色区域，其他为彩色区域。对于不同的彩色区域，混合 H 与 S 变量，划定阈值，即可进行简单的分割。

5）均值滤波：去除图像中的不相关细节，其中“不相关”是指与滤波器模板尺寸相比较小的像素区域，从而对图像有一个整体的认知。即为了对感兴趣的物体得到一个大致的整体的描述而模糊一幅图像，忽略细小的细节。

6）高斯滤波：适用于消除高斯噪声，广泛应用于图像处理的减噪过程。相对于均值滤波平滑效果更柔和，而且边缘保留的也更好。

7）中值滤波：让周围的像素值接近真实的值从而消除孤立的噪声点。该方法在取出脉冲噪声、椒盐噪声的同时能保留图像的边缘细节。这些优良特性是线性滤波所不具备的。

8）维纳滤波：主要用于噪声抑制、系统辨识、预测和反演模拟。

9）膨胀：可以将断裂开的目标物合并为一个整体；可以填补图像上物体中的细小空洞。

10）腐蚀：可以将粘连在一起的不同目标物体分离；可以滤掉小的颗粒噪声；

11）开闭：开操作的作用是使轮廓平滑，抑制物体边界的小离散点或尖峰；用来消除小物体、在纤细点处分离物体、平滑较大物体的边界的同时并不明显改变其面积。闭操作的作用是用来填充物体内的细小空洞、连接邻近物体、平滑其边界的同时并不明显改变其面积。综合起来，先开后闭，开消除噪声，闭修复开运算造成的指纹断裂。

- 除了上述操作外，你还知道其他什么有效的图像预处理方法？请列出 2-3 种，并解释其有效性。

答：图像预处理的主要目的就是消除图像中无关的信息，恢复有用的真实信息，增强有关信息的可检测性和最大限度地简化数据，从而改进后续特征抽取、图像分割、匹配和识别的可靠性。其他常见的图像预处理方法还有：

1）边缘检测：根据灰度的不连续性，找出两种景物的分界线，主要思想就是抑制噪声（低通滤波、平滑、去噪、模糊），然后边缘特征增强（高通滤波、锐化），最后边缘定位。

2）角点检测：是一种有特征的点，包含了图像的局部特征，常见的基于边缘的角点检测算法首先对图像进行分割和边缘提取，角点为两条以上边缘的交点，该方法计算步骤复杂，现代检测算法应用较少；基于灰度的角点检测算法主要是计算梯度和曲率来计算角点，方法简单，在检测算法中应用广泛。

3）直线检测：对图像进行边缘检测，判断边缘中点形成的小线段是否具有相同的斜率，将具有相同斜率的归为一个集合，计算集合中点出现的频率，以此判断是否有直线的存在。

参考文献

[1]张德丰. 数字图像处理(MATLAB 版)[M].人民邮电出版社:, 201501.381.

[2]高飞. MATLAB 图像处理 375 例[M].人民邮电出版社:, 201510.504.

[3]张铮,倪红霞,苑春苗,杨立红. 精通 Matlab 数字图像处理与识别[M].人民邮电出版社:, 201304.412.

[4]张铮,徐超,任淑霞,韩海玲. 数字图像处理与机器视觉[M].人民邮电出版社:, 201405.596.

备注

关于源代码的使用环境等问题的说明：

MATLAB R2020a，无需配置环境，所有代码皆为.m 的脚本文件，直接运行即可得到.fig 的图像处理结果。