

**《数字图像处理实践》**

**实践报告**

专业：

年级：

姓名：

学号：

**实验一 图像基本变换**

**一、实验内容**

打开一幅彩色图像Image1，使用 Matlab 图像处理函数，对其进行下列变换：

（1）将 Image1 色彩通道互换，并显示效果；

（2）将 Image1 灰度化为 gray，并显示灰度化后图像；

（3）采用不同的插值方法实现 gray 的旋转、放大变换；

（4）打开另一幅彩色图像 Image2，和 Image1 进行代数运算，要求运用拼接、加减乘除等多种技术；

（5）实验要求中的拓展内容：将彩色图像采用不同的灰度化方法实现灰度化；将彩色图像变换到 YCbCr、HSV 空间，熟悉各分量数据并显示；不采用 Matlab 函数，自行设计基于双线性插值的图像放大程序。

**二、算法原理**

本实验主要使用的Matlab图像处理函数包括：imread、imshow、imwrite、

rgb2gray、imrotate、imresize、flip、im2double、imadd、imsubtract、immultiply、imdivide、rgb2hsv、rgb2ycbcr等。主要函数基本功能及算法原理说明如下。

（1）rgb2gray函数

功能：彩色图像灰度化。

调用格式：I = rgb2gray(RGB)：真彩色 RGB 图像变换为灰度图像 I。 其中，RGB 为真彩色图像（M×N×3），I 为对应的灰度图像（M×N）。此函数采用加权平均法，将 R、G、B 三个通道按照人眼对不同颜色敏感度的差异加权求和，一般形式为：I = 0.2989 R + 0.5870 G + 0.1140 B，从而得到视觉效果更符合人眼感受的灰度图像。

（2）imrotate函数

功能：实现图像旋转。

调用格式：B = imrotate(A,ANGLE,METHOD,BBOX)。其中，A 为待旋转图像；ANGLE 为旋转角度（单位为度），逆时针为正，顺时针为负；METHOD 为插值方式，可选 'nearest'、'bilinear'，默认 'nearest'；BBOX 用于指定输出图像的尺寸范围，可选 'crop' 或 'loose'：'crop'：输出图像 B 与输入图像 A 尺寸相同，旋转后超出范围的部分被裁剪；'loose'（默认）：输出图像 B 完整包含旋转后的全部内容，尺寸可能大于原图。

（3）imresize函数

功能：实现图像缩放。

调用格式：B = imresize(A, SCALE,METHOD))：返回原图 A 的 SCALE 倍大小图像 B；B = imresize(A, [NUMROWS NUMCOLS], METHOD))：对原图 A 进行比例缩放，返回图像 B 的行数 NUMROWS 和列数 NUMCOLS，如果二者为 NaN，表明 Matlab 自动调整了图像的缩放比例；[Y, NEWMAP] = imresize(X, MAP, SCALE, METHOD))：对索引图像进行成比例缩放。

（4）flip函数

功能：按指定维度翻转矩阵或图像。

调用格式：B=flip(X,DIM)：使矩阵 X 按特定轴翻转，dim 指定翻转方式：为 1 表示按行翻转；为 2 表示按列翻转。

（5）imadd函数

C=imadd(A,B) ：实现两幅图像相加。

A、B 均为图像，则要求 B 和 A 的尺寸相等；若 B 是一个标量，则 C 表示对图像 A 整体加上某个值（对小数部分取整）。 假如 A 和 B 对应运算和大于 255，C 仍取 255，即截断处理；为避免截断，可以将 C 存储为uint16，即 C=imadd(A,B,’uint16’)。

（6）imsubtract函数

功能：实现两幅图像相减。

调用格式：C=imsubtract(A,B)：差值结果小于 0 的赋值为 0，对 A、B 的要求同 imadd 相同。C=imabsdiff(A,B)：差值结果取绝对值。

（7）im2double函数

功能：将图像数据转换为 double 类型，并将像素值归一化至 0～1 范围。

调用格式：I2 = im2double(I)。I 可为 uint8、uint16、logical 或 single 型图

像（二维灰度或三维彩色）；I2 为与 I 同尺寸的 double 型图像。该函数常作为滤波、插值和色彩空间变换等浮点运算的预处理步骤，有助于避免数据溢出并提高运算精度。

（8）rgb2hsv函数

功能：实现 RGB 数据图像向 HSV 数据图像的转换。

调用格式： HSV = rgb2hsv(RGB)。其中，RGB 为三通道真彩色图像，HSV 为对应的三通道图像，其三个分量分别为：H（Hue，色调）：表示颜色类型，反映颜色在色相环中的位置；S（Saturation，饱和度）：表示颜色的纯度；V（Value，亮度）：表示颜色的明暗程度。HSV 的各分量值均被归一化到 [0,1] 范围内，便于后续分析与处理。

（9）rgb2ycbcr函数

功能：将 RGB 彩色图像转换到 YCbCr 颜色空间。

调用格式：YCBCR = rgb2ycbcr(RGB) 。其中，RGB 为输入的真彩色图像，YCBCR 为输出的三通道图像，其三个分量含义为：Y：亮度分量；Cb：蓝色色度差分量；Cr：红色色度差分量。YCbCr 颜色空间常用于视频编码与压缩，其核心思想是将亮度与色度信息分离，以便对不同分量采用不同的压缩策略。

（10）最近邻插值算法

最近邻插值是最简单的图像几何变换插值方法。当图像旋转或缩放后，新图

像中的像素通常对应到原图中的一个非整数位置”（即浮点坐标）。最近邻插值的思想是：直接找到离该位置最近的原图像素点，把它的值作为新像素的值。即：



其中 (u, v) 是新图像像素映射到原图的坐标。

算法优缺点：实现简单，速度非常快；不需要加权计算，也不需要邻域查找；容易出现“块状”现象；图像看起来比较“硬”或“颗粒感强”。适用于速度要求极高、图像内容简单、不在乎平滑效果的情况。

（12）双线性插值算法

相比于最近邻插值方法，双线性插值更精确，它利用该点周围的 4 个邻近像素进行加权平均，根据距离的线性关系计算新的像素值。当新像素映射到原图坐标 (u, v) 时，找到它邻域的四个整数坐标点：(i, j)、(i, j+1)、(i+1, j)、(i+1, j+1)。设a = u - i（x方向的小数部分），b = v - j（y方向的小数部分），双线性插值公式为：



即先在 x 方向做一次线性插值，再在 y 方向做一次插值。因此称为双线性。

算法优缺点：平滑效果比最近邻好得多；图像缩放或旋转后边缘更自然；不适合过度放大（仍可能模糊）；运算量适中。

**三、程序流程**

设计的基于双线性插值放大程序流程图1-1所示。



图1-1 自设计双线性插值流程图

**四、主要代码**

Image1 = imread("Images\peppers.jpg","jpg");

% 红绿通道互换（R/G/B->1/2/3）

Image2 = Image1;

Image2(:,:,1) = Image1(:,:,2);

Image2(:,:,2) = Image1(:,:,1);

imshow(Image2);

imwrite(Image2,'red-green.jpg');

% 红蓝通道互换

Image3 = Image1;

Image3(:,:,1) = Image1(:,:,3);

Image3(:,:,3) = Image1(:,:,1);

imshow(Image3);

imwrite(Image3,'red-blue.jpg');

% 蓝绿通道互换

Image4 = Image1;

Image4(:,:,2) = Image1(:,:,3);

Image4(:,:,3) = Image1(:,:,2);

imshow(Image4);

imwrite(Image4,'green-blue.jpg');

% 灰度化

gray = rgb2gray(Image1);

figure;

subplot(1,2,1),imshow(Image1),title('Original Image');

subplot(1,2,2),imshow(gray),title('Gray Image');

imwrite(gray,'gray\_image.jpg');

% 图像旋转

Newgray1 = imrotate(gray,15,"nearest");

Newgray2 = imrotate(gray,15,"bilinear");

figure;

subplot(1,2,1),imshow(Newgray1),title('旋转15°（最邻近插值）');

subplot(1,2,2),imshow(Newgray2),title('旋转15°（双线性插值）');

imwrite(Newgray1,'rorate1.jpg');

imwrite(Newgray2,'rorate2.jpg');

% 图像缩放

Newgray3 = imresize(gray,2.5,"nearest");

Newgray4 = imresize(gray,2.5,"bilinear");

figure;

subplot(1,2,1),imshow(Newgray3),title('放大2.5倍（最近邻插值）');

subplot(1,2,2),imshow(Newgray4),title('放大2.5倍（双线性插值）');

imwrite(Newgray3,'scale1.jpg');

imwrite(Newgray4,'scale2.jpg');

% 图像镜像与拼接

Image2 = imread("Images\lotus.bmp");

HImage = flip(Image2,2);

VImage = flip(Image2,1);

CImage = flip(HImage,1);

[h,w] = size(Image2);

NewImage = zeros(h\*2,w\*2,3);

NewImage = [Image2 HImage;VImage CImage];

figure;

imshow(NewImage);

imwrite(NewImage,'new\_lotus.jpg');

% 图像间运算

C\_add = imadd(Image2,VImage);

C\_subtract = imsubtract(Image2,VImage);

C\_subtract\_abs = imabsdiff(Image2,VImage);

C\_multiply = immultiply(Image2,VImage);

C\_divide = imdivide(Image2,VImage);

figure;

subplot(2,3,1),imshow(Image2),title('Original Image');

subplot(2,3,2),imshow(C\_add),title('Add');

subplot(2,3,3),imshow(C\_subtract),title('Subtract');

subplot(2,3,4),imshow(C\_subtract\_abs),title('Abs\_subtract');

subplot(2,3,5),imshow(C\_multiply),title('Multiply');

subplot(2,3,6),imshow(C\_divide),title('Divide');

imwrite(C\_add,'Add\_image.jpg');

imwrite(C\_subtract,'Subtract\_image.jpg');

imwrite(C\_subtract\_abs,'Abs\_subtract\_image.jpg');

imwrite(C\_multiply,'Multiply\_image.jpg');

imwrite(C\_divide,'Divide\_image.jpg');

% 不同方法灰度化

Image = im2double(Image1);

R = Image(:,:,1);

G = Image(:,:,2);

B = Image(:,:,3);

gray\_avg = (R+G+B)/3; % 均值化方法

figure;imshow(gray\_avg);title('Gray（Average）');

imwrite(gray\_avg,'Average\_gray.jpg');

gray\_max = max(max(R,G),B); % 最大值方法

figure;imshow(gray\_max);title('Gray（Max）');

imwrite(gray\_max,'Max\_gray.jpg');

gray\_min = min(min(R,G),B); % 最小值方法

figure;imshow(gray\_min);title('Gray（Min）');

imwrite(gray\_min,'Min\_gray.jpg');

% 彩色图像变换到HSV空间，熟悉各分量数据并显示

HSV = rgb2hsv(Image1);

figure;

subplot(1,3,1);imshow(HSV(:,:,1));title('HSV - H');

subplot(1,3,2); imshow(HSV(:,:,2)); title('HSV - S');

subplot(1,3,3); imshow(HSV(:,:,3)); title('HSV - V');

% 彩色图像变换到YCbCr空间，熟悉各分量数据并显示

YCBCR = rgb2ycbcr(Image1);

figure;

subplot(1,3,1); imshow(YCBCR(:,:,1)); title('YCbCr - Y');

subplot(1,3,2); imshow(YCBCR(:,:,2)); title('YCbCr - Cb');

subplot(1,3,3); imshow(YCBCR(:,:,3)); title('YCbCr - Cr');

% 设计基于双线性插值的图像放大程序

function enlarged = bilinear\_zoom(img, scale)

% BILINEAR\_ZOOM 基于双线性插值的图像放大程序（自写）

% img : 灰度图 (H×W) 或彩色图 (H×W×3)，建议先用 im2double 归一化

% scale : 放大倍数（>1 的实数）

% enlarged : 放大后的图像

% --- 判断是否为彩色图像 ---

if ndims(img) == 3

[H, W, C] = size(img);

else

[H, W] = size(img);

C = 1;

img = reshape(img, [H, W, 1]); % 统一成三维，便于写循环

end

% --- 目标图像尺寸 ---

newH = round(H \* scale);

newW = round(W \* scale);

enlarged = zeros(newH, newW, C);

% --- 对每个通道分别做双线性插值 ---

for c = 1:C

for i = 1:newH % i: 目标图像中的行索引

for j = 1:newW % j: 目标图像中的列索引

% 1）映射回原图中的“浮点坐标”（行 x，对应 y）

x = (i - 0.5) / scale + 0.5;

y = (j - 0.5) / scale + 0.5;

% 2）边界处理，防止访问越界

if x < 1, x = 1; end

if x > H - 1, x = H - 1; end

if y < 1, y = 1; end

if y > W - 1, y = W - 1; end

% 3）找到四个邻近整数格点

x1 = floor(x); x2 = x1 + 1;

y1 = floor(y); y2 = y1 + 1;

% 4）计算距离权重

dx = x - x1;

dy = y - y1;

% 5）取四个邻点的像素值

I11 = img(x1, y1, c);

I12 = img(x1, y2, c);

I21 = img(x2, y1, c);

I22 = img(x2, y2, c);

% 6）双线性插值公式

enlarged(i, j, c) = ...

(1-dx)\*(1-dy)\*I11 + ...

(1-dx)\*dy \*I12 + ...

dx \*(1-dy)\*I21 + ...

dx \*dy \*I22;

end

end

end

% --- 如果是灰度图，去掉多余的第三维 ---

if C == 1

enlarged = enlarged(:,:,1);

end

end

I = im2double(Image1); % 转成 0~1 双精度

big = bilinear\_zoom(I, 2); % 放大 2 倍

figure;imshow(big); title('自写双线性插值放大效果图1');

imwrite(big,'bilinear\_zoom\_2x.jpg');

gray\_d = im2double(gray);

big\_gray = bilinear\_zoom(gray\_d, 2);

figure;imshow(big\_gray); title('自写双线性插值放大效果图2');

imwrite(big\_gray,'bilinear\_zoom\_2x\_gray.jpg');

**五、运行结果**

图1-2为不同通道色彩互换结果图。由图可知，当红绿色彩通道互换时，原始图像中红椒变成绿椒，绿椒变成红椒；当红蓝色彩通道互换时，原始图像中红椒变成蓝色，绿椒颜色基本不变；当绿蓝色通道互换时，原始图像中绿椒变成偏蓝色，红椒颜色基本不变。

|  |  |  |  |
| --- | --- | --- | --- |
| peppers | red-green | red-blue | green-blue |
| （a）原始图像 | （b）红绿色彩通道互换 | （c）红蓝色彩通道互换 | （d）绿蓝色彩通道互换 |

图1-2 不同色彩通道互换图

图1-3为对原始图像进行一系列变换得到的实验结果。其中图像（b）为对原始图像进行灰度变换得到的图像；图像（c）为对变换后的灰度图像基于最邻近插值逆时针旋转15°后所得结果；图像（d）为对变换后的灰度图像基于双线性逆时针旋转15°后所得结果；图像（e）为基于最邻近插值对灰度图像放大2.5倍所得图像；图像（f）为基于双线性插值对灰度图像放大2.5倍所得图像。

|  |  |  |  |
| --- | --- | --- | --- |
| peppers | gray_image | rorate1 | rorate2 |
| （a）原始图像 | （b）灰度化图像 | （c）旋转(最邻近插值) | （d）旋转(双线性插值) |
| scale1 | | scale2 | |
| （e）放大(最邻近插值) | | （f）放大(双线性插值) | |

图1-3 一系列变换结果图



图1-4 图像镜像与拼接结果图

图1-4为图像镜像与拼接实验结果图。其中左上角图像为原始图像，右上角图像为原图按列翻转，即左右颠倒；左下角图像为原图按行翻转，即上下颠倒；右下角图像为右上角图像按行翻转得到。最终拼接得到图1-4。

|  |  |  |
| --- | --- | --- |
| Add_image | Subtract_image | Abs_subtract_image |
| （a）图像加法 | （b）图像减法 | （c）图像减法（绝对值） |
| Multiply_image | Divide_image |  |
| （e）图像乘法 | （f）图像除法 |  |

图1-5 图像运算实验结果图

图1-5为图像运算实验结果图。计图1-4中左上角图像为A，左下角图像为B：图（a）为A加B实验结果图；图（b）为A减B实验结果图；图（c）为A减B取绝对值的实验结果图，可以看见相比于图（b）中一些值小于0被截断为0，在图像中显示为黑色，图（c）中取绝对值之后黑色区域明显减少；图（d）为A乘B实验结果图，可以看见乘法运算之后大于255值被截断为255，呈现白色；图（f）为A除以B实验结果图，除法结果矩阵里存在大量大于 1 的浮点值，imshow 把这些值按默认伪彩色映射表直接标成了蓝色。

图1-6为不同灰度化方法对图像进行灰度变换实验结果。图（a）为Matlab内置rgb2gray函数按NTSC标准（0.299R+0.587G+0.114B）加权灰度化的结果；图（b）为平均灰度化实验结果，具体为三通道算术平均（（R+G+B）/3）；图（c）为最大值灰度化实验结果，具体为取R、G、B最大值 max(R,G,B) 构成的最大值灰度化结果；图（d）为最小值灰度化实验结果，具体为取R、G、B最小值 min(R,G,B) 构成的最小值灰度化结果。

|  |  |  |  |
| --- | --- | --- | --- |
| gray_image | Average_gray | Max_gray | Min_gray |
| （a）权重灰度化 | （b）均值灰度 | （c）最大值灰度化 | （d）最小值灰度化 |

图1-6 不同灰度化方法变换实验结果

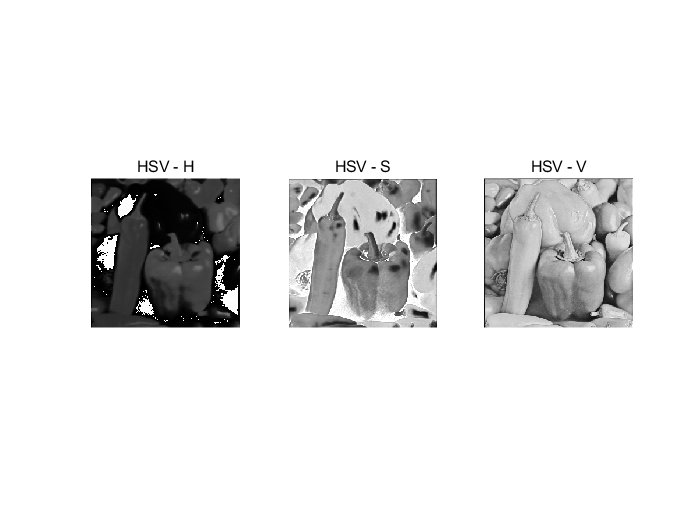


图1-7 变换至HSV空间



图1-8 变换至YCbCr空间

图1-7给出了原始图像转换到 HSV 颜色空间后的三个分量图像，从左到右依次为 H（色调）、S（饱和度）和 V（明度）通道。由图可见，H 通道反映的是颜色在 0～360°色相环上的分布，画面中不同颜色的大致轮廓仍然可以区分；S 通道主要刻画颜色的纯度，饱和度较高的区域呈现为较亮的灰度，而低饱和度或接近灰度的区域则较暗；V 通道则对应亮度信息，明亮区域灰度值较高，暗部灰度值较低，整体结构与原图的光照分布基本一致。通过对比可以看出，HSV 空间能够将色调信息与亮度信息有效分离，有利于后续基于颜色或亮度的图像分析与处理。

图1-8显示的是图像转换到 YCbCr 颜色空间后的三个分量图像，分别为 Y、Cb 和 Cr 通道。其中，Y 通道表示亮度分量，与人眼观察到的灰度图像最为接近，保留了图像的大部分细节和边缘信息；Cb 通道为蓝色色度差分量，数值越大表示蓝色成分越强，由于原图中无明显蓝色，因此均表现较暗灰度；Cr 通道为红色色度差分量，红色成分丰富的区域（如红椒）灰度值较高，而绿椒的灰度值较低。Y 与 Cb、Cr 的分离，使得在保持亮度细节的同时，可以对色度分量进行单独处理。除此之外，由于这种亮度和色差信号的表示方式便于压缩和传输，YCbCr 颜色空间被广泛应用于 JPEG、MPEG 等图像与视频编码标准中。

|  |  |
| --- | --- |
| bilinear_zoom_2x | bilinear_zoom_2x_gray |
| （a）设计程序放大2倍图（RGB） | （b）设计程序放大2倍图（Gray） |

图1-9 基于双线性插值自行设计程序结果图

图1-9为基于双线性插值自行设计程序放大原始图像为原来2倍结果图。其中图 (a) 为彩色图像（RGB）的放大结果，图 (b) 为灰度图像（Gray）的放大结果。可以看到，自编双线性插值程序在放大后能够较好地保持原图像的整体结构与细节信息，没有明显的块状伪影或锯齿状边缘，表明该算法在图像几何变换中具有较好的视觉效果。

**六、实验心得**

实验一让我对数字图像在空间域中的各种基本操作有了系统、直观的认识。从最基础的通道互换、灰度化开始，到旋转、缩放、镜像拼接以及图像间的加减乘除运算，都让我更清楚地理解了图像其实就是一个矩阵，不同函数本质上是在做矩阵重排、插值或逐元素运算。特别是在最近邻插值与双线性插值的对比中，可以明显看到前者虽然实现简单、速度快，但放大后锯齿感明显；而双线性插值在边缘和平滑性方面表现更好，这也让我体会到“算法复杂度”和“视觉效果”之间的权衡关系。

在不同灰度化方法以及颜色空间变换（RGB→HSV、YCbCr）的实验中，我进一步认识到：图像的表现形式可以很多样，但本质信息是相通的。比如 HSV 中的 H、S、V 分别更适合描述色调、饱和度和亮度；YCbCr 则把亮度与色差信息分离，便于压缩与编码。最后，自行编写双线性插值放大程序的过程，加深了我对几何变换坐标映射和四点加权公式的理解，也体验到了从“调用库函数”到“亲自实现算法”的差别，整体上提升了我对图像基本变换原理和实现细节的掌握程度。

**实验二：图像增强**

**一、实验内容**

打开一幅彩色图像 Image1，使用 Matlab 图像处理函数，对其进行下列变换：

（1）将 Image1 灰度化为 gray，统计并显示其灰度直方图；

（2）对 gray 进行分段线性变换；

（3）对 gray 进行直方图均衡化；

（4）对 gray 进行伪彩色增强；

（5）对 gray 添加噪声并平滑；

（6）对 gray 利用 Sobel 算子锐化；

（7）实验要求中的拓展内容：对以上处理变换参数，查看处理效果；更改伪彩色增强方法为热金属编码或彩虹编码；设计不同的平滑滤波、锐化滤波方法，查看处理效果；自行设计方法，实现对彩色图像增强处理。

**二、算法原理**

本实验主要使用的 Matlab 图像处理函数包括：imhist 函数、histeq 函数、imnoise 函数等。主要函数基本功能及自己实现算法原理说明如下。

（1）imhist函数

功能：统计变显示图像的直方图。

调用格式：imhist(I)：显示图像 I 的直方图。 imhist(I, n)：显示图像 I 的直方图，n 指定直方图中的列数。[COUNTS,X] = imhist(...)：返回直方图数据向量 COUNTS 和相应的色彩值向量 X。

（2）histeq函数

功能：直方图均衡化

调用格式：J = histeq(I,hgram)：将图像 I 的直方图变成用户指定的向量 hgram，hgram 中的各元素值域为[0,1]；J = histeq(I,N)：对原始图像 I 进行直方图均衡化，N 为输出图像的灰度技术，默认 N 为 64。

（3）线性变换算法

本实验中线性变换算法吧灰度值分为3段，各段做线性拉升或压缩。原图灰度 gray∈[0,1]，定义了四个关键点：a=80/256，b=180/256；c=30/256；d=220/256。三个映射为：[0,a]->[0,c]；[a,b]->[c,d]；[b,1]->[d,1]。以中间段为例，计算公式为：

可达到暗部压缩，中段大幅增强，高亮略提升，使图像整体更清晰、对比度集中在主要灰度区域的效果。

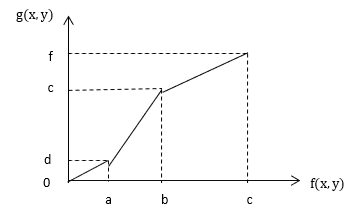


图2-1 线性变换算法示例图

（4）伪彩色增强算法（彩虹编码）

实现效果：随着灰度从0到1，RGB颜色从蓝->青->绿->黄->红。可增强对比度；让不同灰度区域在视觉上更容易区分。具体分为四段，具体说明如下：

第一段：gray < 64/256：R = 0；G = 4\*gray；B = 1；蓝色向青色渐变

第二段：64/256≤gray < 128/256：R = 0；G = 4\*gray；B = 1；青色向绿色渐变；

第三段：128/256≤gray < 192/256：R = 4\*gray - 2；G = 1；B = 0；绿色向黄色渐变；

第四段：gray≥192/256：R = 1；G = 4 - 4\*gray；B = 0；黄色向红色渐变。

（5）模拟噪声算法原理

1）高斯噪声

特点：高斯噪声（Gaussian noise）又称加性白噪声，假定噪声值服从正态分布，取值连续，几乎覆盖整幅图像。由于其在频域中包含较多高频成分，会使图像出现整体的“颗粒感”与轻微模糊。

模型：设噪声为，则，即均值为 0、方差为的高斯随机变量。含噪图像可写为：。MatLab可用：noiseG = imnoise(gray, 'gaussian', 0, 0.01)；其中均值设为0.方差设为0.01。

2）椒盐噪声

特点：椒盐噪声（Salt & Pepper noise）属于典型的脉冲噪声，表现为图像中随机出现的黑点（0）和白点（1），像在图像上撒了“椒盐”一样。它主要集中在高频部分，对人眼视觉影响非常明显。

模型：设噪声密度为，则像素被污染后的概率分布为：



在Matlab可用：noiseSP = imnoise(gray, 'salt & pepper', 0.1)；其中 0.1 表示有 10% 的像素被椒盐噪声污染。

3）乘法噪声

特点：法噪声（Speckle noise）常见于雷达、超声、SAR 等相干成像系统中。该噪声的强度与原始图像亮度相关，表现为细小的“斑点状”纹理，同样含有较多高频成分，会严重影响纹理分析和边缘检测。

常用模型为：。其中为原图，为均值为 0 的随机噪声。可以理解为在原图上叠加一个与亮度成比例的扰动。MATLAB中生成方式为：noiseSpk = imnoise(gray, 'speckle', 0.04)；其中 0.04 为噪声的方差参数。

（6）平滑滤波常用方法

目的：平滑滤波的主要作用是抑制图像中的高频噪声成分，使图像更加平滑、噪声减弱，同时尽量保留有用的结构信息。下面介绍三种常用的空间域平滑滤波方法。

1）均值滤波

算法思想：对每个像素，以其为中心取一个  的邻域窗口，用该窗口内所有像素的算术平均值替代中心像素值，从而减弱随机噪声。数学公式为：



其中为以为中心的领域，为原图像，为滤波后图像。

特点为：实现简单、计算量小，对随机噪声有一定抑制作用，但会明显模糊边缘与细节，对椒盐噪声抑制能力较弱。Matlab中常用 3×3 均值模板：h\_avg = fspecial('average', [3 3]);out\_avg = imfilter(gray, h\_avg, 'replicate')。

2）高斯滤波

算法思想：使用符合高斯分布的权值对邻域像素进行加权平均，距离中心越近权重越大，距离越远权重越小。二维高斯核可表示为：



滤波过程即为原图与该高斯核的卷积。

特点为：平滑效果比简单均值滤波更加自然；对加性高斯噪声的抑制效果较好；通过调节和模板大小，可以控制模糊程度与平滑强度之间的折中。Matlab使用模板为：h\_gaus = fspecial('gaussian', [3 3], 0.5);out\_gaus = imfilter(gray, h\_gaus, 'replicate')。

3）中值滤波

算法思想：中值滤波属于非线性滤波。在以当前像素为中心的邻域内，将所有像素值排序，选取中间的那个值（中值）来替代中心像素，而不是取平均值。数学形式可写为：



特点为：对椒盐噪声、脉冲噪声的抑制能力非常强，因为孤立的 0 或 1 在排序后通常不会处于中间位置；在去除噪声的同时，能较好地保持边缘和细节结构，不会像均值滤波那样严重模糊轮廓；由于是非线性运算，不适合用频域分析，但在空间域中应用十分广泛。MatLab中可直接调用：out\_med = medfilt2(gray, [3 3])。

（7）锐化滤波常用方法

目的：图像锐化的目的在于增强图像中的边缘和细节，使图像看起来更加清晰。从频域角度看，边缘和细节对应图像的高频成分，因此锐化的本质是增强高频分量。一般流程为：首先利用一阶或二阶差分算子（如 Sobel、Laplacian）检测出图像的边缘信息，然后将一定比例的边缘信息叠加到原图中，即：



其中为原图，为边缘图，为锐化系数，这样可以在保持原有灰度结构基础上，加强轮廓与细节。

1）Sobel梯度锐化

算法思想：先用Sobel 算子把“哪里有明显变化（边缘）”找出来；再把这些“边缘亮一点”加回原图使边缘变清晰。

Sobel 算子是一种一阶梯度算子，用于近似计算图像的水平和垂直方向灰度变化率，也就是“梯度”。H1=[-1 -2 -1;0 0 0;1 2 1]，H2=[-1 0 1;-2 0 2;-1 0 1]，H1：检测水平方向的边缘（上下方向的变化），H2：检测垂直方向的边缘（左右方向的变化）。用公式写出来，对灰度图 ，Sobel 近似计算：



其中\*表示卷积，和分别是水平方向和垂直方向的梯度。梯度值越大，说明该位置灰度变化越剧烈，也就越可能是边缘。梯度的幅值通常可以这样近似计算：，代码中为：edgeImage = abs(R1) + abs(R2)。得到的 edgeImage 实际上就是一幅边缘强度图：边缘位置亮、平坦区域暗。

接下来进行锐化，对应前面讲的通用模型：



含义为在在原图基础上，把边缘区域的亮度再抬高一点，从而增强轮廓、纹理，使图像看起来更清晰。

该算法特点为：实现简单，对边缘有明显增强效果，可以直观展示梯度到锐化的过程；对噪声也比较敏感（噪声也是高频信号），若图像含噪较多，建议先平滑再用 Sobel 锐化；

2）Laplacian 锐化（拉普拉斯锐化）

Laplacian 锐化使用的是二阶导数算子，着重刻画灰度的快速变化区域。连续情形下，二维拉普拉斯算子定义为：



它反映了图像灰度的弯曲程度：平坦区域的二阶导数接近0；灰度突变处（边缘）二阶导数显著非零，且可能出现正负变化。离散情况下，使用3×3 模板近似，例如：[0 -1 0;-1 4 -1;0 -1 0]。

拉普拉斯锐化公式：在边缘处幅值大，在平坦区域接近 0；将其乘以一个系数再从原图中减去，相当于对边缘附近的灰度变化进行加强。

该算法特点为：对边缘增强非常明显，锐化效果强；但对噪声极其敏感，因为噪声中包含大量高频分量，容易被二阶导数放大；

3）LoG（Laplacian of Gaussian）

LoG 可以理解为“先平滑、再拉普拉斯”的组合：



其中是高斯核。相比直接用拉普拉斯，LoG 的第一步是：用高斯滤波先对图像做低通平滑，抑制一部分随机噪声；再对平滑后的图像施加拉普拉斯算子。在增强边缘的同时，相比纯拉普拉斯略微减小了对噪声的放大。

LoG锐化公式：。其中是对图像先高斯平滑再二阶求导的结果。

4）Unsharp Mask（反遮罩锐化）

Unsharp Mask 是实际应用中最常见、效果最自然的一种锐化方法，很多图像处理软件中的锐化和清晰度都基于这一思想。核心步骤如下：

· 对原图做平滑滤波，得到一幅“模糊图”；

· 把原图减去模糊图，得到一幅只包含细节或边缘的图像：



· 把这部分高频信息放大后加回原图，得到锐化结果：



其中控制锐化强度。可以把模糊图看成图像的大致轮廓，减掉它后剩下就是细节和边缘。Unsharp 就是在原图上叠加一部分细节，让轮廓更清楚，但整体亮度关系变化不大，所以观感非常自然。该算法锐化效果自然、不容易出现严重伪影，是实际应用中非常常用的一种锐化方法。

（8）热金属编码

算法思想：把灰度值 g∈[0,1] 分成 3 段：

· 0 ~ 1/3：黑 → 红（R 从 0 到 1，G=0，B=0）

· 1/3 ~ 2/3：红 → 黄（R=1，G 从 0 到 1，B=0）

· 2/3 ~ 1：黄 → 白（R=1，G=1，B 从 0 到 1）

（9）自设计彩色图像增强处理算法

彩色图像在 RGB 空间里三个通道都是“混着亮度和颜色”的，直接在 R/G/B 上做直方图均衡或锐化很容易产生颜色失真。因此，我们设计的增强流程是：先把彩色图像从 RGB转为HSV，在 HSV 里：对 V 通道（亮度） 做对比度拉伸 和Unsharp 锐化；对 S 通道（饱和度） 做适度放大，让颜色更鲜艳；最后再转回 RGB。

假设输入彩色图像为，像素值已归一化到：

1）颜色空间变换

将 RGB 图像转换到 HSV 空间：。其中。其中为色调，决定是什么颜色，为饱和度，决定颜色纯不纯，为亮度，决定明暗。

2）亮度通道V对比度拉伸

仅对 V 通道进行动态范围增强。使用类似“自动线性拉伸”的思想，将 V 的低、高手尾部分略微截断，然后在剩余区间内线性拉伸到：



其中可由stretchlim自动估计，本实验中设置保留原来1%-99%像素。

3）亮度通道 V 的锐化（Unsharp 形式）、

在拉伸后的亮度上做 Unsharp Mask：首先用高斯滤波（）得到模糊版本，接着提取高频细节；将高频按系数叠加到亮度上：



最后截断到。即可在不明显改变整体亮度的前提下，增强花瓣边缘和纹理等细节。

4）饱和度 S 的增强

为了让颜色更鲜艳，对饱和度 S 做线性放大：，其中。过低的饱和度区域被适度提升，灰蒙蒙的区域会变得“有颜色”。

5）重构增强后的 HSV 并转回 RGB

保持不变，将增强后的饱和度和亮度 组合：



再转回RGB空间得到最终增强结果：。

本算法将亮度增强与颜色分开控制，避免拉亮的同时把颜色弄“偏”；可以针对亮度通道单独做锐化，只增强细节轮廓；适当提高饱和度，让颜色看起来更通透、鲜艳”。

**三、程序流程**

程序模块图如图2-2所示，具体算法细节见算法原理和主要代码。

****

图2-2 程序模块图

**四、主要代码**

Image1 = im2double(imread("Images\lotus.bmp"));

%% 图像灰度化+统计显示灰度直方图

gray = rgb2gray(Image1);

imwrite(gray,'灰度图像.png');

imhist(gray);

%% 对gray进行分段线性变换

[h,w] = size(gray);

NewImage1 = zeros(h,w); % 新建矩阵

a=80/256;b=180/256;c=30/256;d=220/256;

for x=1:w

for y=1:h

if gray(y,x)<a

NewImage1(y,x) = gray(y,x)\*c/a; % [0,80]->[0,c]

elseif gray(y,x)<b

NewImage1(y,x) = (gray(y,x)-a)\*(d-c)/(b-a)+c; % [30,180]->[30,220]

else

NewImage1(y,x) = (gray(y,x)-b)\*(255-d)/(255-b)+d; % [180-255]->[220,255]

end

end

end

%figure,imshow(NewImage1),title('分段线性变换');

%imwrite(NewImage1,'线性变换.png');

%% 对gray进行直方图均衡化

NewImage2 = histeq(gray);

%figure,imshow(NewImage2),title('直方图均衡化');

%imwrite(NewImage2,'直方图均衡化.png');

%% 对gray进行伪彩色增强（彩虹编码）

NewImage3 = zeros(h,w,3);

for x=1:w

for y=1:h

if gray(y,x)<64/256

NewImage3(y,x,1)=0;

NewImage3(y,x,2)=4\*gray(y,x);

NewImage3(y,x,3)=1;

elseif gray(y,x)<128/256

NewImage3(y,x,1)=0;

NewImage3(y,x,2)=1;

NewImage3(y,x,3)=2-4\*gray(y,x);

elseif gray(y,x)<192/256

NewImage3(y,x,1)=4\*gray(y,x)-2;

NewImage3(y,x,2)=1;

NewImage3(y,x,3)=0;

else

NewImage3(y,x,1)=1;

NewImage3(y,x,2)=4-4\*gray(y,x);

NewImage3(y,x,3)=0;

end

end

end

%figure,imshow(NewImage3),title('伪彩色增强');

%imwrite(NewImage3,'伪彩色增强.png');

%% 对gray添加噪声并平滑

noiseG = imnoise(gray, 'gaussian', 0, 0.01); % 高斯噪声

noiseSP = imnoise(gray, 'salt & pepper', 0.1); % 椒盐噪声

noiseSpk = imnoise(gray, 'speckle', 0.04); % 乘法噪声

imwrite(noiseG,'高斯噪声.png');imwrite(noiseSP,'椒盐噪声.png');imwrite(noiseSpk,'乘法噪声.png');

h\_avg = fspecial('average', [3 3]); % 3x3 均值滤波器

h\_gau = fspecial('gaussian', [3 3], 0.5);% 3x3 高斯滤波器，sigma=0.5

win\_med = [3 3]; % 中值滤波窗口

% 对高斯噪声图滤波

g\_avg = imfilter(noiseG, h\_avg, 'replicate');

g\_gau = imfilter(noiseG, h\_gau, 'replicate');

g\_med = medfilt2(noiseG, win\_med);

imwrite(g\_avg, '高斯\_均值滤波.png');imwrite(g\_gau, '高斯\_高斯滤波.png');imwrite(g\_med, '高斯\_中值滤波.png');

% 对椒盐噪声图滤波

sp\_avg = imfilter(noiseSP, h\_avg, 'replicate');

sp\_gau = imfilter(noiseSP, h\_gau, 'replicate');

sp\_med = medfilt2(noiseSP, win\_med);

imwrite(sp\_avg, '椒盐\_均值滤波.png');imwrite(sp\_gau, '椒盐\_高斯滤波.png');imwrite(sp\_med, '椒盐\_中值滤波.png');

% 对乘法噪声图滤波

sk\_avg = imfilter(noiseSpk, h\_avg, 'replicate');

sk\_gau = imfilter(noiseSpk, h\_gau, 'replicate');

sk\_med = medfilt2(noiseSpk, win\_med);

imwrite(sk\_avg, '乘法\_均值滤波.png');imwrite(sk\_gau, '乘法\_高斯滤波.png');imwrite(sk\_med, '乘法\_中值滤波.png');

%% 对gray利用Sobel算子锐化以及其他算子锐化

% Sobel梯度锐化

H1 = [-1 -2 -1;0 0 0;1 2 1]; % 垂直方向

H2 = [-1 0 1;-2 0 2;-1 0 1]; % 水平方向

R1 = imfilter(gray, H1, 'replicate'); % 垂直梯度

R2 = imfilter(gray, H2, 'replicate'); % 水平梯度

edgeImage = abs(R1)+abs(R2);

k\_sobel = 0.5; % 锐度强化系数

sharp\_sobel = gray + k\_sobel\*edgeImage;

sharp\_sobel = max(min(sharp\_sobel,1),0); % 截断到[0,1]

imwrite(edgeImage,'Sobel梯度图像.png');imwrite(sharp\_sobel,'Sobel锐化图像.png');

% Laplacian锐化

h\_lap = fspecial('laplacian', 0.2); % alpha=0.2，控制中心权重

lap\_resp = imfilter(gray, h\_lap, 'replicate');

k\_lap = 1.0;

sharp\_lap = gray - k\_lap \* lap\_resp; % g = f - laplacian(f)

sharp\_lap = max(min(sharp\_lap, 1), 0);

imwrite(mat2gray(lap\_resp),'Laplacian响应图像.png');

imwrite(sharp\_lap,'Laplacian锐化图像.png');

% LoG锐化

h\_log = fspecial('log', [5 5], 0.5); % 5x5，高斯σ=0.5

log\_resp = imfilter(gray, h\_log, 'replicate');

k\_log = 1.0;

sharp\_log = gray - k\_log \* log\_resp;

sharp\_log = max(min(sharp\_log, 1), 0);

imwrite(mat2gray(log\_resp),'LoG响应图像.png');

imwrite(sharp\_log,'LoG锐化图像.png');

% Unsharp Mask反遮罩锐化

sharp\_unsharp = imsharpen(gray, 'Radius', 1, 'Amount', 1.2); % Radius 控制模糊范围，Amount 控制锐化强度

imwrite(sharp\_unsharp, 'Unsharp锐化图像.png');

%% 热金属编码

NewImage\_hot = zeros(h,w,3);

for y=1:h

for x=1:w

g=gray(y,x);

if g<1/3 % 黑 -> 红

R=3\*g;

G=0;

B=0;

elseif g<2/3 % 红 -> 黄

R=1;

G=3\*g-1;

B=0;

else % 黄 -> 白

R=1;

G=1;

B=3\*g-2;

end

NewImage\_hot(y,x,1)=R;

NewImage\_hot(y,x,2)=G;

NewImage\_hot(y,x,3)=B;

end

end

imwrite(NewImage\_hot,'热金属编码.png');

%% 自设计方法进行彩色图像增强

hsvImage = rgb2hsv(Image1);

H = hsvImage(:,:,1); % 色调

S = hsvImage(:,:,2); % 饱和度

V = hsvImage(:,:,3); % 亮度

lowHigh = stretchlim(V,[0.01,0.99]); % 去除前1%和后1%

V\_cs = imadjust(V,lowHigh,[0,1]); % 线性拉伸

h\_guas = fspecial('gaussian',[5,5],1); % 高斯滤波

V\_blur = imfilter(V\_cs,h\_guas,'replicate'); % 模糊图

k = 0.3;

V\_sharp = V\_cs+k\*(V\_cs-V\_blur); % 锐化

V\_sharp = max(min(V\_sharp,1),0); % 截断

alpha = 1.3;

S\_enh = S\*alpha; % 饱和度增强

S\_enh = max(min(S\_enh,1),0); % 截断

hsv\_enh = cat(3,H,S\_enh,V\_sharp);

rgb\_enh = hsv2rgb(hsv\_enh);

imwrite(rgb\_enh,'彩色增强图.png');

**五、运行结果**

|  |  |
| --- | --- |
| 灰度图像 | **直方图统计图** |
| （a）原始灰度图像 | （b）灰度直方图 |
| 直方图均衡化 | 线性变换 |
| （c）直方图均衡化 | （d）分段线性变换 |

图2-3 实验二参考代码运行结果

实验二参考代码运行结果如图2-3所示。（a）为原始灰度图像，（b）为其灰度直方图，可见直方图整体集中在低灰度区域（约 0～0.6），中高灰度部分像素较少，说明图像整体偏暗，亮区域较少，对比度分布不均。

（c）为直方图均衡化实验结果图，直方图均衡化试图将灰度值在 [0,1] 区间上尽可能拉伸并均匀分布：从图像效果看，整体亮度显著提高，暗部细节被“提亮”，花瓣纹理更加清晰；背景与前景之间的对比度增大，花朵轮廓更清楚；但同时也可以观察到亮部略有偏白。可见直方图均衡化在整体提升对比度方面效果显著，但容易造成局部过增强。

（d）为分段线性变换实验结果图，通过设置三段线性变换区间，分别改变不同灰度范围的对比度，可见分段线性变换后，背景明显变暗，主体莲花从背景中“凸显出来”，整体层次感增强。

|  |  |
| --- | --- |
| 椒盐噪声 | 椒盐_均值滤波 |
| （a）椒盐噪声 | （b）均值滤波 |
| 椒盐_高斯滤波 | 椒盐_中值滤波 |
| （c）高斯滤波 | （d）中值滤波 |

图2-4 椒盐噪声及滤波结果图

盐噪声模拟和各种滤波方法实验结果如图2-4所示。图（a）中随机出现大量黑白“噪点”，对莲花主体和背景均有明显干扰。均值滤波实验结果如图（b）所示，此滤波方法对椒盐噪声有一定抑制，噪点数量明显减少；但由于简单求平均值，噪声点会拉低（或拉高）邻域的平均灰度，导致图像出现明显的模糊现象。高斯滤波实验结果如图（c）所示，与均值滤波类似，高斯滤波也是对邻域进行加权平均，只是靠近中心的权值更大，可见此滤波方法对椒盐噪声的抑制能力有限，仍然可以看到较多黑白噪点，图像同样存在一定程度的模糊。中值滤波实验结果如图（d）所示，可见对椒盐噪声的抑制效果最为明显：大部分孤立的白点和黑点被成功去除；花瓣轮廓和主体结构基本得以恢复，模糊程度远小于均值、高斯滤波，边缘过渡较为自然。因此中值滤波是处理椒盐噪声的最优选择，既能有效去除脉冲噪声，又能较好地保持边缘信息。

|  |  |
| --- | --- |
| 高斯噪声 | 高斯_均值滤波 |
| （a）高斯噪声 | （b）均值滤波 |
| 高斯_高斯滤波 | 高斯_中值滤波 |
| （c）高斯滤波 | （d）中值滤波 |

图2-5 高斯噪声及滤波结果图

高斯噪声模拟和各种滤波方法效果如图2-5所示。图（a）整幅图像上覆盖均匀的随机“颗粒”。均值滤波实验结果如图（b）所示，此滤波方法对高斯噪声有一定抑制作用，颗粒感明显下降，但边缘与细节同样被平均化，出现明显的“涂抹感”，花瓣纹理变得模糊。高斯滤波实验结果如图（c）所示，高斯滤波的噪声模型与高斯噪声高度匹配，因此去噪效果最好，可见噪声被有效抑制，图像整体仍保持相对清晰的结构。中值滤波实验结果如图（d）所示，结果图中高斯噪声仍然存在，但部分极端噪声被削弱；花瓣结构保留较好，边缘较为清晰。因此对高斯噪声而言，高斯滤波器在噪声抑制与细节保留之间取得了较好的平衡，是本实验中表现最好的一种平滑滤波方法；均值滤波去噪但严重模糊，中值滤波对该类噪声的针对性较弱。

乘法噪声模拟和各种滤波方法效果如图2-6所示。图（a）出现类似“颗粒状”的斑点结构，尤其在中高灰度区域更加明显；且噪声的强度与原始亮度相关，亮的地方颗粒更重，暗部相对较轻。均值滤波实验结果如图（b）所示，此滤波方法对对整体颗粒感有一定缓解，背景略显平滑，但由于噪声是乘法形式，与图像本身耦合较紧，线性平均难以完全消除。高斯滤波实验结果如图（c）所示，相比均值滤波，高斯滤波在减弱颗粒噪声的同时，保留了一部分结构信息，但画面仍然略显粗糙。中值滤波实验结果如图（d）所示，中值滤波对斑点噪声同样具有一定抑制作用，尤其对孤立的强噪点效果较好；但由于乘法噪声并非完全孤立的单个像素，而是成片分布，因此中值滤波无法像处理椒盐噪声那样实现“几乎完全消除”。因此对乘法噪声而言，单一的空间域平滑滤波难以完全恢复原图，均值、高斯、中值各有优缺点：高斯滤波平滑自然，中值滤波更能保留边缘，而均值滤波最简单但模糊最明显。

|  |  |
| --- | --- |
| **乘法噪声** | **乘法_均值滤波** |
| （a）乘法噪声 | （b）均值滤波 |
| 乘法_高斯滤波 | 乘法_中值滤波 |
| （c）高斯滤波 | （d）中值滤波 |

图2-6 乘法噪声及滤波结果图

锐化滤波实验结果如图2-7所示。（a）为原始灰度图像。（b）为Unsharp Mask 锐化图像，可见图像整体清晰度明显提升，花瓣边缘与纹理细节更加突出；背景保持平滑，没有出现明显的“光晕”或伪影。

（c）为Sobel梯度图像，图像几乎只保留了边缘结构，内部区域接近全黑；轮廓线条清晰，能够直观看出各个物体的边界；反映了 Sobel 算子对一阶梯度敏感的特点，是典型的边缘检测结果。（d）为Sobel锐化图像，在原图上叠加 Sobel 梯度信息后，花瓣轮廓和表面纹理变得更加清晰；但与 Unsharp 相比，Sobel 锐化对边缘的增强更“硬”，对高频和噪声较敏感。

（e）为拉普拉斯响应图像，拉普拉斯响应图强调的是灰度的二阶变化，也就是“弯曲度”；图中只有边缘附近有明显灰度变化，且出现正负变化，说明拉普拉斯算子对灰度突变极其敏感，是一种典型的高通响应图。（f）为拉普拉斯锐化图像，将拉普拉斯响应按一定比例从原图中减去后，边缘锐度显著提高；花瓣轮廓非常清晰，但同时背景中的噪声和细小纹理也被放大，画面略显“生硬”；体现了拉普拉斯锐化增强强度大、对噪声非常敏感的特点。

（g）为LoG 响应图像，相比纯拉普拉斯响应，LoG 响应整体更加平滑，噪声点较少，这是因为 LoG 在计算二阶导数前先做了高斯平滑，削弱了小的随机噪声。（h）为LoG 锐化图像，锐化效果介于 Unsharp 和拉普拉斯之间：边缘清晰度明显提升；噪声放大程度小于纯拉普拉斯锐化，整体视觉效果较为稳定。

|  |  |  |  |
| --- | --- | --- | --- |
| 灰度图像 | Unsharp锐化图像 | Sobel梯度图像 | Sobel锐化图像 |
| （a）原始灰度图像 | （b）Unsharp Mask锐化 | （c）Sobel梯度图像 | （d）Sobel锐化图像 |
| Laplacian响应图像 | Laplacian锐化图像 | LoG响应图像 | LoG锐化图像 |
| （e）拉普拉斯响应图像 | （f）拉普拉斯锐化图像 | （g）LoG响应图像 | （h）LoG锐化图像 |

图2-7 不同锐化方法实验结果图

图2-8为彩虹编码和热金属编码实验结果图。图（a）彩虹编码将灰度值从低到高依次映射为蓝→青→绿→黄→红五段颜色，因此在图像中形成了强烈且富有层次的色彩变化。从图中可用看出：暗部区域呈现深蓝或青蓝色；中等亮度区域呈现绿色或黄绿色；结合图2-3（b）的灰度直方图，无黄色到红色的渐变。整体来看，彩虹编码对比强烈、层次清晰，非常适用于观察亮度变化范围较宽、细节丰富的灰度图像。图（b）热金属编码从黑→红→黄→白的色阶变化模拟了“温度从低到高”的视觉感受。从图中可见暗部区域呈现深红甚至接近黑色；中等亮度区域呈现橙红或亮红色；亮部区域呈现明亮的黄色甚至白色；色彩过渡相较彩虹编码更柔和。整体来看，热金属编码具有暖色调突出、中心高亮明显、背景沉稳、视觉集中的特点，更适合表现亮区与暗区差别明显的场景，相比彩虹编码，它的视觉冲击力稍弱，但图像的立体感更强。

|  |  |
| --- | --- |
| 伪彩色增强 | 热金属编码 |
| （a）彩虹编码 | （b）热金属编码 |

图2-8 伪彩色增强实验结果图

|  |  |
| --- | --- |
| lotus | 彩色增强图 |
| （a）莲花rgb原图 | （b）彩色增强图 |

图2-9 自设计彩色图像增强方法结果图

图2-9为自设计彩色图像增强方法结果图。由图（b）可见，经本实验设计的基于 HSV 空间的彩色图像增强方法处理后，莲花图像整体亮度有所提升，暗部细节得到有效恢复，花瓣纹理和边缘轮廓更加清晰。同时，饱和度适度增强，使花瓣与背景的色彩对比更加鲜明，视觉感受更为通透。

**六、实验心得**

实验二主要围绕如何让图像看起来更清晰、更易于观察展开，让我对直方图均衡化、分段线性变换、伪彩色增强、噪声与滤波以及锐化算法有了较为全面的实践体会。首先，通过灰度直方图分析，我学会了用统计分布来判断图像的亮度与对比度情况：例如直方图集中在低灰度端说明整体偏暗；直方图均衡化虽然能显著拉伸对比度，但也可能出现局部过度增强，这提醒我在实际应用中需要结合任务和视觉效果选择合适的增强强度。分段线性变换则提供了一种“按灰度分段精细调节”的思路，可以针对暗部、中灰和亮部分别设计增益，更灵活地突出目标区域。

在噪声和滤波实验部分，我对不同噪声模型和滤波器的适用场景有了更直观的认识：中值滤波在去除椒盐噪声方面效果最好，高斯滤波在抑制高斯噪声时表现更均衡，而简单均值滤波虽然实现容易，却往往以模糊细节为代价。锐化实验中，将 Sobel、Laplacian、LoG 与 Unsharp Mask 进行比较，可以清楚地看到不同算子对边缘和噪声的增强程度：高阶算子锐化更强，但也更容易放大噪声，而基于平滑+细节叠加的 Unsharp 方法视觉效果更自然。最后，自设计的 HSV 空间彩色增强方法，让我体会到在颜色空间中分开处理亮度和颜色的优势：只对 V 通道做对比度拉伸和锐化，同时适度提高 S 通道，可以在保持色彩真实的前提下显著提升图像通透感。这一实验整体上强化了我对增强并不等于简单变亮，而是需要在噪声、细节和对比度之间寻找平衡的理解。

**实验三 图像分割与描述**

**一、实验内容**

打开一幅图像 Image，使用 Matlab 图像处理函数，对其进行下列变换：

（1）将 Image 灰度化为 gray，对其进行阈值分割转换为 BW；

（2）对 BW 进行数学形态学滤波；

（3）对 BW 进行边缘跟踪，用红色线在图中标出；

（4）计算各区域边界点的傅里叶描绘子并用四分之一点重建边界；

（5）实验要求中的拓展内容：尝试不同的阈值选择方法，实现灰度图像二值化；变换参数实现形态学滤波，查看滤波效果；更改重建边界点数，查看效果； 自行设计方法实现图像分割，并计算分割区域相关参数。

**二、算法原理**

本实验主要使用的 Matlab 图像处理函数包括：im2bw函数、graythresh 函数、strel 函数、bwboundaries 函数、fft函数、ifft函数等。主要函数基本功能及自己实现算法原理说明如下。

（1）graythresh 函数

自动计算一个阈值，调用方法为：T = graythresh(I)。具体为使用 Otsu 方法获取阈值，plane图像中T = 0.4824。

Otsu 方法为基于直方图的自动全局阈值选择方法，原理为：在所有可能的阈值中，找到一个阈值 T，把像素分成：类 0：灰度 ≤ T（背景)；类 1：灰度 > T（前景）。希望两类之间的差异尽量大（类间方差最大），类内尽量集中（类内方差最小）。

1）对每个候选阈值t

计算两类像素的占比：为小于等于t的像素所占比例，；计算两类的平均灰度值：：类0的平均灰度，：类1的平均灰度，图像全局平均灰度。

2）计算类间方差



3）在所有阈值t中，选择使最大的那个t作为最佳阈值。

（2）im2bw函数

BW = im2bw(I, T)：以 T 为阈值把灰度图像 I 转变为二值图像。T：阈值，为0~1 之间的小数），输出 BW：逻辑型（0/1）的二值图。

具体计算逻辑为：如果 I(x,y) > T，则BW(x,y) = 1；否则 0。注意：当 I 是 uint8 时，T 仍然按 0~1 范围来解释，实际阈值是 T\*255。

（3）形态学滤波

数学形态学是一套基于“形状”的图像处理方法，特别适合处理二值图（也可以用于灰度图）。核心思想为：把图像看成一个由像素构成的“集合”，用一个小的模板即结构元素去“扫描”这个集合，通过集合运算实现几何上的变换。可分为腐蚀、膨胀、开运算、闭运算。下面介绍以3×3全1模板为例介绍。

1）腐蚀

对于输出图像中的某个像素 (x,y)：看输入图像中以 (x,y) 为中心的 3×3 区域；只要这个 3×3 区域中 有一个是 0，则输出位置就变成 0；只有当这个 3×3 区域 全部都是 1，输出才是 1。效果为物体边缘会被往里面“收缩”一圈；太小的白噪点直接被去除；细细的白线、尖锐的小凸起，很容易被腐蚀掉。

2）膨胀

对于输出图像中的某个像素 (x,y)：看输入图像中以 (x,y) 为中心的 3×3 区域；只要这个 3×3 区域里 有一个是 1，输出就设为 1；只有当整个 3×3 区域全是 0，输出才是 0。效果：物体边缘会向外扩一圈；小黑洞、小裂缝、窄缝隙容易被“填满”为白色；細线能变粗，断开的两端很近的话，可能粘在一起。

3）开运算：先腐蚀，再膨胀

调用方式为：Morph = imopen(BW, SE)。直观效果：去掉面积尺寸小于 SE 的那些“小白块噪声”；把原来的前景物体轮廓变得更“圆滑”，去掉很细的突出、锯齿。常用来去除小的孤立前景噪声；不显著影响比较大的前景目标。

4）闭运算：先膨胀，再腐蚀

调用方式为：Morph = imclose(Morph, SE)。直观效果：填补小的黑色孔洞、裂缝；连接距离很近的前景区域（让它们成为一个整体）；同样有平滑边界的效果。常用来填洞、连接断裂的前景区域；让区域内部更加连通、整体。代码中先开后闭可以理解为一种形态学平滑滤波。

（4）迭代阈值法

迭代阈值法适合灰度分布大致呈两团、且对噪声不太敏感的情况。假设图像

大致可以分为前景和背景两类，通过“先分两类 → 求均值 → 更新阈值”反复迭代，让阈值逐步收敛：

1）取初始阈值（通常用整幅图的平均灰度）；

2）用当前阈值把像素分成两类：类1：灰度≤（背景）；类2：灰度>

（前景）；

3）分别求两类的平均灰度，更新阈值；

4）若足够小则停止，得到最终阈值。

（5）最大熵阈值法

最大熵法适合纹理较多或多峰直方图的图像。把灰度直方图看成概率分布，

用某个阈值t把图像分成背景类和前景类，希望这两类的信息熵之和最大：

1）统计归一化直方图（0～255 灰度）；

2）对每个候选阈值t，计算：背景概率：，前景概率：；

条件概率，；背景熵、前景熵，总熵。

3）取使最大的t作为最佳阈值。

（6）局部阈值法

全局阈值对整幅图只用一个T，当图像存在光照不均匀、阴影、局部对比度变化时容易失效。自适应阈值法则为每个像素在其邻域窗口内计算一个“局部阈值” T(x,y)，根据局部亮度和对比度进行判决，更适合背景变化大的场景。MATLAB 里可以直接用 adaptthresh + imbinarize 实现局部阈值。

（7）strel 函数

功能：创建形态学结构元素。

SE = strel(shape,parameters)，创建一个由 shape 指定的结构元素，其中 shape 的种类有：arbitrary、pair、diamond、periodicline、disk、rectangle、line、square、octagon，参数 parameters 一般控制 SE的大小。

1）SE = strel('square', n)：生成一个 n × n 的全 1 正方形邻域；

2）SE = strel('rectangle', [m n])：占据 m × n 的矩形区域，全 1

3）SE = strel('disk', R)/SE = strel('disk', R, N)：R为半径（像素单位）、N：

可选参数，控制“圆”用多少边近似（默认自动），一般可以不写。

4）SE = strel('diamond', R)：生成一个以中心为顶点、曼哈顿距离（cityblock

distance）为 R 的菱形。

5）SE = strel('line', len, theta)：len：线段长度（像素）；theta：线段的角度

（单位：度，0° 表示水平向右，90° 表示垂直向上）。

6）SE = strel('octagon', n)：生成一个八边形的结构元素，本质上是对圆的

另一种近似；参数 n 大致控制八边形尺寸。

7）arbitrary ：任意形状结构元素

比如N = [0 1 0;1 1 1;0 1 0]; % 十字形

SE = strel('arbitrary', N);

N 是一个 0/1 矩阵，1 表示结构元素中“被包含”的位置。可以设计任意形状。

（8）bwboundaries 函数

功能：搜索二值图像 BW 的外边界和内边界。

常见调用方式：

B = bwboundaries(BW);

[B,L] = bwboundaries(BW);

[B,L,N,A] = bwboundaries(BW, CONN, OPTIONS);

输入BW为二值图像，0 为背景，1 为目标。CONN为连通性，4表示4 连通（上、下、左、右），8表示8 连通（再加左上、右上、左下、右下），默认是 8。OPTIONS分为两种：'holes'：搜索外边界 + 内部的孔洞边界（默认）；'noholes'：只搜索外边界。

输出：B：cell 数组，长度等于找到的边界数量。B{i} 是一个 Q×2 矩阵，Q 是第 i 个边界的像素点数。每一行是一个边界点的坐标：[row, col]（行、列）。

L：标记矩阵（label matrix），大小与 BW 相同。背景为 0；每一个连通区域会被标号为 1,2,3,…，便于后续对每一块区域单独处理。N：区域数（连通域的个数）。A：邻接矩阵，表示各个区域之间的上下包含、邻接关系（一般在做拓扑分析才用到）。

函数视 BW 中为 0 的元素为背景像素点，为 1 的元素为待提取边界目标。B 中的每个元素均为 Q×2 矩阵，矩阵中每一行包含边界像素点的行坐标和列坐标，Q 为边界所含像素点的个数。L，标识矩阵，标识二值图像中被边界所划分的区域；N，区域的数目 N；A，被划分的区域的邻接关系。CONN 取 4，搜索中采用 4 连通方法，默认取 8，即 8 连通方法。OPTIONS 指定算法的搜索方式，默认为‘holes’，搜索目标的内外边界，‘noholes’只搜索目标的外边界。

（9）fft函数以及ifft函数

1）1D DFT 与 fft：把序列分解成“不同频率的正弦/余弦”

对一个长度为N的复数序列，离散傅里叶变换定义为：



原序列为在“时间域”或“空间域”的信号（比如边界点坐标随点序号变化），变换后为在“频率域”的表示，每个对应一个频率分量，为这个频率分量的“强度”（幅值），为相位（决定波形的平移、形状细节）。

可以理解为DFT 把一个复杂的波形/曲线，分解成一堆不同频率的“正弦 + 余弦”成分；低频负责整体形状，高频负责细节或波动。fft(X)为 DFT 的快速算法。

2）1D DFT 与 fft2：对整幅图像做频域分析

当信号是二维的（比如一张灰度图像），可使用二维 DFT：



其中为空间域图像，为频率域表示，每个对应一个二维空间频率：为水平方向频率（x向），为垂直方向频率（y向）。

3）ifft / ifft2：把频域再还原回“图像/曲线”

1D 的逆变换公式是：



和DFT只差一个符号和。2D 的逆变换也类似，把变回。

（10）K-means 聚类分割

1. means 聚类是一种基于样本“相似性”的无监督学习方法，其目标是将数据自动划分为 K 个类别，使得同一类内部的样本尽量相似、不同类之间的样本差异尽量大。在灰度图像分割中，可以将每一个像素看作一个“样本点”，其灰度值作为特征，通过聚类将像素划分为若干灰度层次，从而实现前景与背景的自动分离。

设灰度图像共有N个像素，记第i个像素的灰度为，希望将所有像素划分为 K 个聚类，对应的类中心（平均灰度）为。K-means的优化目标是最小化类内平方误差和：



即希望同一聚类中的像素灰度尽量接近其类中心。

采用迭代方式求解，基本步骤如下：

1. 初始化类中心：从图像中选取 K 个像素灰度作为初始聚类中心，或随机

初始化类中心。

1. 样本分配：对每个像素，计算其与各类中心的距离（灰度差的绝对值

或平方），将其分配给距离最近的聚类：。

1. 更新类中心：对每一类，根据当前归类结果重新计算类中心为该类所

有像素灰度的平均值：



1. 迭代直至收敛：重复步骤 2 和 3，直至聚类中心变化很小或像素的类别

标记不再改变，认为算法收敛。

在本实验中，将灰度图像展开为一维向量，以灰度值作为聚类特征，选取 K=2和K=3 将像素自动划分为二/三类。聚类完成后，根据各类中心灰度大小，从中选取灰度均值最大的那一类作为前景（莲花）区域，其余类别视为背景，从而得到一幅基于 K-means 的二值分割图。

（11）区域生长法

核心思想为：从一个或多个“种子点”出发，把和种子“足够相似”的邻域

像素一点点并进来，长成一整块区域。

（12）区域合并法

核心思想为：从整幅图像作为一个大区域开始，检查它是否“足够均匀”（例如灰度方差<阈值）；如果不均匀，就把它分裂成 4 个子块（四叉树分裂）；对每个子块递归重复上面的步骤：够均匀就停，不均匀就继续分裂；分裂完成后，检查相邻的小块：如果它们“相似”，就合并成一个大块。

**三、程序流程**

****

图3-1 程序模块图

**四、主要代码**

1. 参考程序+不同形态学滤波代码

Image1 = im2double(imread("Images\plane.jpg"));

gray = rgb2gray(Image1);

%% 阈值分割，转化为BW

T = graythresh(gray); % 获取阈值 T=0.4824

BW = im2bw(gray,T); % 转为2值图像

figure,imshow(BW),title('二值化图像');

imwrite(BW,'二值化图像.png');

%% 形态学滤波

SE = strel('square',3); % 创建一个3x3全1模板

Moraph = imopen(BW,SE); % 开操作

Moraph = imclose(Moraph,SE); % 闭操作

figure,imshow(Moraph),title('形态学滤波');

imwrite(Moraph,'形态学滤波.png');

%% 不同形态学滤波

% 模板1：3x3 +形结构元素

N = [0 1 0;

1 1 1;

0 1 0];

SE\_1 = strel('arbitrary',N);

M\_1 = imopen(BW,SE\_1);

M\_1 = imclose(M\_1,SE\_1);

imwrite(M\_1,'模板1\_+形3x3.png');

% 模板2：3x3 X形结构元素

N\_2 = [1 0 1;

0 1 0;

1 0 1];

SE\_2 = strel('arbitrary', N\_2);

M\_2 = imopen(BW, SE\_2);

M\_2 = imclose(M\_2, SE\_2);

imwrite(M\_2, '模板2\_X形3x3.png');

% 模板3：1x5 水平线结构元素

N\_3 = [1 1 1 1 1]; % 一行五个 1

SE\_3 = strel('arbitrary', N\_3);

M\_3 = imopen(BW, SE\_3);

M\_3 = imclose(M\_3, SE\_3);

imwrite(M\_3, '模板3\_水平线1x5.png');

% 模板4：3x3 结构元素

N\_4 = [0 0 1;

0 1 0;

1 0 0];

SE\_4 = strel('arbitrary', N\_4);

M\_4 = imopen(BW, SE\_4);

M\_4 = imclose(M\_4, SE\_4);

imwrite(M\_4, '模板4\_3x3.png');

% 模板5：5x1 垂直线结构元素

N\_5 = [1;1;1;1;1]; % 一列五个 1

SE\_5 = strel('arbitrary', N\_5);

M\_5 = imopen(BW, SE\_5);

M\_5 = imclose(M\_5, SE\_5);

imwrite(M\_5, '模板5\_垂直线5x1.png');

%% 进行边缘跟踪，用红色线在图中标出

[B,L] = bwboundaries(1-Moraph);

figure,imshow(L),title('划分的区域');

hold on;

for i=1:length(B)

boundary = B{i};

plot(boundary(:,2),boundary(:,1),'r','LineWidth',2);

end

%% 计算各区域边界点的傅里叶描绘子并用四分之一点重建边界

M = zeros(length(B)); % 预分配

for k=1:length(B)

N = length(B{k}); % 第k条边界的点数

if N/2 ~= round(N/2) % 若N/2不是整数->N为奇数

B{k}(end+1,:) = B{k}(end,:); % 再复制最后一个点，让N变成偶数

N = N+1;

end

M(k) = [N\*3/4]; % 保存一个数 N\*3/4，丢掉 N\*3/4 个最不重要的频率分量（系数）

end

S = zeros(size(Moraph)); % 预分配

figure,imshow(S);

hold on;

for k=1:length(B) % 对每个边界做傅里叶平滑

z = B{k}(:,2)+1i\*B{k}(:,1); % 把 (x,y) 坐标变成复数形式 z = x + i\*y

Z = fft(z); % 对边界做一维 FFT，得到傅里叶系数（描述子）

[Y,I] = sort(abs(Z)); % 按系数的幅值从小到大排序，I 是排序后原索引

for count=1:M(k)

Z(I(count))=0; % 将幅值最小的 M(k) 个系数置零

end

zz = ifft(Z); % 逆 FFT，恢复到“平滑后”的边界坐标

plot(real(zz),imag(zz),'w'); % 在图上画出白色轮廓

end

1. 多种阈值选择方法实验代码

Image1 = im2double(imread("Images\lotus.bmp"));

gray = rgb2gray(Image1);

figure, imshow(gray), title('灰度图');

imwrite(gray,'莲花灰度图.png');

figure, imhist(gray), title('灰度直方图');

%% 1. Otsu 阈值分割，转化为 BW

T = graythresh(gray); % 获取阈值 T=0.3098

BW = im2bw(gray,T); % 转为2值图像

figure,imshow(BW),title('二值化图像');

imwrite(BW,'阈值分割\_Otsu.png');

%% 2. 迭代阈值法（全局阈值迭代）

T\_iter = mean(gray(:)); % 初始阈值：全图平均灰度

delta = 1e-3; % 收敛精度

while true

G1 = gray(gray <= T\_iter); % 类1：背景

G2 = gray(gray > T\_iter); % 类2：前景

% 防止某一类为空

if isempty(G1) || isempty(G2)

break;

end

m1 = mean(G1(:));

m2 = mean(G2(:));

T\_new = (m1 + m2) / 2;

if abs(T\_new - T\_iter) < delta

break; % 阈值收敛

end

T\_iter = T\_new;

end

BW\_iter = gray > T\_iter; % 用迭代得到的阈值二值化

imwrite(BW\_iter, '阈值分割\_迭代法.png');

%% 3. 最大熵阈值法

[counts, bins] = imhist(gray); % 对 [0,1] 灰度做直方图，bins 是对应灰度值

p = counts / sum(counts); % 归一化概率

H\_total = -inf(256,1); % 初始化为 -inf，防止未赋值

for t = 2:255 % 阈值在 bins(t) 处

Pb = sum(p(1:t)); % 背景概率

Pf = sum(p(t+1:end)); % 前景概率

if Pb == 0 || Pf == 0

continue; % 某一类为空则跳过

end

pb = p(1:t) / Pb; % 背景条件概率

pf = p(t+1:end) / Pf; % 前景条件概率

pb\_nz = pb(pb > 0);

pf\_nz = pf(pf > 0);

Hb = -sum(pb\_nz .\* log(pb\_nz));

Hf = -sum(pf\_nz .\* log(pf\_nz));

H\_total(t) = Hb + Hf; % 总熵

end

[~, t\_star] = max(H\_total); % 熵最大的索引

T\_entropy = bins(t\_star); % 对应的灰度阈值（0~1）

BW\_entropy = gray > T\_entropy;

imwrite(BW\_entropy, '阈值分割\_最大熵法.png');

%% 4. 自适应 / 局部阈值法（Adaptive Threshold）

T\_local = adaptthresh(gray, 0.5, 'NeighborhoodSize', [31 31]);

BW\_adapt = imbinarize(gray, T\_local);

imwrite(BW\_adapt, '阈值分割\_自适应局部阈值.png');

1. k-means实现图像分割算法代码

Image1 = im2double(imread("Images\lotus.bmp"));

gray = rgb2gray(Image1);

%% K-means 聚类实现图像分割（以灰度为特征）

% 1）准备数据：把灰度图展开成 N×1 向量

X = gray(:);

% 2）K-means 聚类

K = 3; % 聚 2 类（前景 / 背景）

rng(0); % 固定随机种子

[idx, C] = kmeans(X, K, ... % idx: 每个像素所属簇标签(1..K)，C: 每类中心灰度

'Replicates', 3, ... % 重复聚类 3 次，取最优结果

'MaxIter', 200);

% 3）把聚类标签还原成图像

L\_kmeans = reshape(idx, size(gray));

% 4）选出“最亮的一类”作为前景（莲花）

[~, fgCluster] = max(C); % 灰度中心最大的那一类

BW\_kmeans = (L\_kmeans == fgCluster); % 前景=1，背景=0 的二值图

figure, imshow(BW\_kmeans), title('K-means-2 聚类分割结果');

imwrite(BW\_kmeans, '分割\_Kmeans2聚类.png');

% 5）把不同聚类结果伪彩色显示

RGB\_label = label2rgb(L\_kmeans);

figure, imshow(RGB\_label), title('K-means-2 聚类标签伪彩色图');

imwrite(RGB\_label, 'Kmeans2\_聚类标签伪彩色.png');

%% 6）分割区域相关参数计算（连通域分析）

% 这里用清理后的结果 BW\_clean，也可以改成 BW\_kmeans 自己对比

[L, num] = bwlabel(BW\_kmeans); % L 是标签图，num 是连通区域数目

fprintf('K-means 分割后共检测到 %d 个连通区域。\n', num);

% 计算常见区域特征

stats = regionprops(L, ...

'Area', 'Perimeter', 'Centroid', ...

'BoundingBox', 'Eccentricity', 'EquivDiameter');

% 追加一个“圆度”指标 Roundness = 4\*pi\*Area / Perimeter^2

for k = 1:num

A = stats(k).Area;

P = stats(k).Perimeter;

if P > 0

stats(k).Roundness = 4\*pi\*A / (P^2);

else

stats(k).Roundness = NaN;

end

end

% 找面积最大的区域（一般就是整朵莲花）

[~, idx\_max] = max([stats.Area]);

main = stats(idx\_max);

fprintf('\n面积最大的区域为 #%d ：\n', idx\_max);

fprintf(' Area = %d 像素\n', main.Area);

fprintf(' Perimeter = %.2f 像素\n', main.Perimeter);

fprintf(' Centroid = (%.2f, %.2f)\n', main.Centroid(1), main.Centroid(2));

fprintf(' Eccentricity= %.3f\n', main.Eccentricity);

fprintf(' EquivDiameter = %.2f 像素\n', main.EquivDiameter);

fprintf(' Roundness = %.3f\n', main.Roundness);

**五、运行结果**

|  |  |  |
| --- | --- | --- |
| plane | 二值化图像 | 红色边缘图像 |
| （a）原始图像 | （b）二值化图像 | （c）滤波及划分的区域 |

图3-2 实验三参考代码运行结果图

图3-2为实验三参考代码运行结果图，原始图像（a）中的飞机目标与背景天空在亮度和颜色上差异明显，天空区域灰度较为均匀，目标飞机整体偏暗，具备较好的前景背景可分性。利用Otsu 法自动选取全局阈值（0.4824）后，对灰度图进行二值化处理，得到图（b）。可以看到，二值化结果中飞机被清晰地提取为黑色区域，背景基本为白色，仅在机翼边缘和发动机附近存在极少量细小噪声点，说明 Otsu 阈值在该类简单背景场景下能够较好地区分前景和背景。

在此基础上，引入 3×3 正方形结构元素，通过开运算和闭运算实现形态学滤波，得到图（c）所示的“滤波及划分的区域”。开运算首先去除了二值图中零散的孤立小区域和平滑了飞机轮廓的尖锐细节；随后的闭运算有效填补了飞机内部的微小孔洞和机翼局部断裂，使目标区域更加连通、边界更加规整。随后利用 bwboundaries 对过滤后的二值图进行边缘跟踪，在分割结果的基础上用红色轮廓标出每个连通区域的边界。从图中可以看出，主要连通区域与飞机的真实轮廓高度吻合，机身、机翼和尾翼均被完整勾勒，背景区域基本不再出现错误分割。

|  |  |  |
| --- | --- | --- |
| 形态学滤波 | 模板1_+形3x3 | 模板2_X形3x3 |
| （a）3×3全1模板 | （b）3×3“+”形模板 | （c）3×3“X”形模板 |
| 模板3_水平线1x5 | 模板5_垂直线5x1 | 模板4_3x3 |
| （d）水平线1×5模板 | （e）垂直线5×1模板 | （f）3×3反对角线模板 |

图3-3 不同元素实验结果图

图3-3为不同元素模板实验结果图，从图中可以看出，在本例中飞机目标较大、背景较干净，因此 3×3 全1模板、“+”形模板和“X”形模板得到的轮廓整体差别不大，均能完整保留机身与机翼，只是在局部边缘平滑方向上略有差异。水平线 1×5 和垂直线 5×1 模板体现出明显的方向性：前者更倾向于连接、平滑水平方向的断裂，使机翼更连续；后者主要平滑机身的竖直轮廓，对机翼略有收缩。3×3 反对角线模板则突出另一组斜向结构。

|  |  |  |
| --- | --- | --- |
| 重建图像1-8 | 重建图像1-4 | 重建图像1-3 |
| （a）重建图像1/8 | （b）重建图像1/4 | （c）重建图像1/3 |
| 重建图像1-2 | 重建图像3-4 |  |
| （d）重建图像1/2 | （e）重建图像3/4 |  |

图3-4 不同重建边界点数结果图

图3-4为不同重建边界点数结果图，图例中分数为保留的占原有的点数的占比。当仅保留 1/8 的边界点时（图（a）），轮廓已经能勾勒出飞机的大致形状，机身和机翼位置清晰，但局部细节明显被抹平，机翼末端和机身下方结构较为圆滑。随着重建点数占比从 1/4、1/3、1/2 增加到 3/4（图（b - e）），轮廓逐渐接近原始边界，机翼、发动机和机身底部的细小突起被逐步恢复，但平滑程度略有下降。总体来看，较小的点数占比适合获取平滑、简化的轮廓，较大的占比则更有利于保留边界细节，反映了傅里叶描绘子在“形状简化”和“细节保真”之间的权衡关系。

由于原始的 plane 图像前景与背景对比度较高、灰度分布呈明显双峰，用不同阈值方法得到的结果差异不够明显。为更充分比较阈值选择策略的优劣，本文选取纹理较丰富、背景较复杂的莲花图像进行实验，结果如图3-5所示。图（a）为莲花灰度图；图（b）给出的灰度直方图呈多峰分布。采用 Otsu 法得到的全局阈值（0.3098）对图像进行分割，如图（c）所示，莲花主体轮廓被较好地提取出来，但在右下角及部分背景叶片处仍保留了较多噪声和杂点；图（d）为迭代阈值法（0.3103）的分割结果，其总体效果与 Otsu 法相近；图（e）为最大熵阈值法（0.3098）的结果，从视觉上看与前两种基本一致。

与前三种全局阈值方法不同，自适应局部阈值法在每个像素邻域内根据局部亮度和对比度计算阈值，其结果如图（f）所示。可以看到，莲花的轮廓不仅被提取出来，花瓣内部的纹理结构、叶片边缘以及背景的细小纹理也被大量显现出来，产生了明显的过分割现象，噪声点和纹理线条显著增多。

|  |  |  |
| --- | --- | --- |
| 莲花灰度图 | 莲花灰度直方图 | 阈值分割_Otsu |
| （a）莲花灰度图 | （b）灰度直方图 | （c）Ostu阈值分割 |
| 阈值分割_迭代法 | 阈值分割_最大熵法 | 阈值分割_自适应局部阈值 |
| （d）迭代法阈值分割 | （e）最大熵法阈值分割 | （f）自适应局部阈值分割 |

图3-5 不同阈值分割实验结果图

图3-6为自设计聚类分割方法实验结果图，从图中可以看到，K-means 聚类在 K=2 和 K=3 两种设置下都能够将莲花与背景大致区分开来，但分割精细程度和区域形态存在明显差异。对于 K=2 的情况，聚类结果将图像划分为前景（亮类）和背景（暗类）两部分，结合伪彩色图可以看出，莲花主体基本被归入同一簇，仅在边缘和右下角叶片区域存在少量噪声分割。连通域分析表明，共检测到 135 个连通区域，其中面积最大的区域面积为 72653 像素，对应莲花主体。该区域的等效直径约为 304 像素，圆度为 0.162，偏心率为 0.673，说明整体轮廓较为饱满，形状相对紧凑，能够较好地反映整朵莲花的整体形态。当聚类数增加到 K=3 时，算法在亮度空间上进一步把图像细分为三类：莲花高亮区域、较暗的花瓣/叶片以及背景。伪彩色图显示，莲花内部被划分为多个颜色不同的子区域，局部阴影和叶片纹理被单独聚为一类，导致前景更加碎片化。此时检测到的连通区域增至 151 个，面积最大的区域面积为 46114 像素，明显小于 K=2 时的主体区域，而周长却增加到 3114.16 像素，圆度下降到 0.060、偏心率提高到 0.816，表明该区域边界更加曲折、形状更细长且不规则，出现一定程度的过分割现象。综合来看，K=2 时的聚类分割更适合作为“提取整朵莲花”的方案。

对于飞机图像，图 3-6 (d)、(e) 给出了在 K=2 时的聚类分割结果及其标签伪彩色图。可以看到，K-means 能够较为稳定地将飞机主体与天空背景分开。结合伪彩色图可以直观地看出，前景与背景在聚类空间中已经形成明显的两簇。说明设计算法在前后景明显情况下可较好地区分目标与背景，得到轮廓完整、结构清晰的分割结果。

|  |  |
| --- | --- |
| 分割_Kmeans2聚类 | Kmeans2_聚类标签伪彩色 |
| （a）聚类分割（2类） | （b）标签伪彩色图（2类） |
| 分割_Kmeans3聚类 | Kmeans3_聚类标签伪彩色 |
| （c）聚类分割（3类） | （d）标签伪彩色图（3类） |
| 分割_Kmeans2聚类 | Kmeans2_聚类标签伪彩色 |
| （d）plane聚类分割（2类） | （e）plane标签伪彩色图（2类） |

图3-6 聚类分割实验结果图

1. **实验心得**

实验三让我比较系统地走完了从灰度图到目标目标区域的整个流程：先用不同阈值方法做初始分割，再用腐蚀、膨胀、开闭运算对结果进行清理，最后提取边界并用傅里叶描绘子做形状描述。通过对 Otsu 全局阈值、迭代阈值、最大熵和局部自适应阈值的对比，我直观地感受到：灰度分布简单、前景背景差异明显时，全局阈值就足够；而光照不均或背景复杂时，必须借助自适应或基于熵的算法才能得到稳定分割，没有一种方法可以一招通吃。

在形态学部分，通过对二值图进行腐蚀、膨胀及开闭运算，我看到了这些操作在去除小噪声、填补孔洞、连通断裂和平滑轮廓方面的效果，也体会到结构元素大小、形状对结果影响很大。最后利用 bwboundaries 提取轮廓并计算傅里叶描绘子，让我第一次意识到形状可以在频域上用少量系数进行压缩表达，既能重建大致轮廓，又方便后续做形状匹配和识别。整体来说，实验三让我更清楚地认识到：图像分割不是单一步骤，而是阈值、形态学和形状特征等多种手段的组合，只有综合利用灰度、结构和形状信息，才能得到对后续分析真正有意义的分割结果。

**实验四 图像综合处理**

**一、实验内容**

利用所学各种图像处理技术，实现对图像的综合处理，加深对基础知识的理解和应用。选题为：目标与背景的分割与提取1、目标与背景的分割与提取2。

目标与背景的分割与提取1：主要要求为提取红苹果。难点为确定目标区域的特征，边界修复与区域分割。

目标与背景的分割与提取1：主要要求为从图像中检索出篮球。难点为篮球上的黑色花纹对边缘检测、色彩检测均造成干扰。

**二、算法原理**

（1）红苹果提取算法

本算法基于HSV 颜色空间阈值分割和形态学处理对红色苹果进行自动分割，并通过孔洞填充和面积滤波进一步去除小洞与小碎块，最后利用二值掩膜从原始图像中提取出苹果目标区域。

1. 图像预处理与颜色空间转换

首先将彩色图像读入并转换为 double 型，像素值归一化到 [0,1] ，便于后续处理和显示；接着使用标准差为的高斯滤波对图像进行平滑，削弱细小噪声和纹理干扰，同时尽量保留物体轮廓，为后续阈值分割提供更稳定的输入；

最后将平滑后的图像从 RGB 空间转换到 HSV 空间，并分别提取色调 H、饱和度 S、亮度 V 三个通道。HSV 空间将“颜色”（H、S）与“亮度”（V）分离，更适合根据颜色特征（如“红色”）进行分割。

1. 基于 HSV 的初始红色区域分割

在 HSV 空间中，红色的色调 H 分布有“绕 0 点”的特点：一部分接近 0，另一部分接近 1。故代码中设置了两个色调区间：当或 时认为颜色为红色；同时要求饱和度，滤除灰度、暗色等低饱和背景区域。

接着得到满足上述条件的初始二值掩膜，这里不限制V，可以保留苹果表面阴影区域，避免因亮度变化导致分割缺失。

1. 形态学闭运算与小洞、小碎块去除

初始二值掩膜中，苹果轮廓可能存在细小缺口、内部孔洞，以及背景中的零散噪点。为获得干净、连通的苹果区域，需要进行形态学运算和区域过滤。

首先使用半径为 5 的圆盘结构元素 SE，其形状与苹果轮廓相对匹配，先膨胀后腐蚀：膨胀：填补边缘小缺口，使苹果区域更连贯；腐蚀：基本恢复原有大小，同时保留“封住”的缝隙，使轮廓更平滑完整。

接着进行孔洞填充，对闭运算结果进行孔洞填充，将苹果内部因高光等原因产生的小黑洞补齐，令苹果区域成为实心连通区域。

最后面积滤波去小碎块、小洞边缘残留，这里使用利用 bwareaopen 删除面积小于 minArea（200像素）的连通区域，可有效去除：背景上的小白点噪声；苹果边缘附近的细小伪连通区域；极小的错误分割区域。

最终得到形态学精修后的苹果二值掩膜 BW\_clean，既连通完整，又不含明显小洞和小碎块。

1. 利用掩膜抠出苹果区域

将单通道二值图 BW\_clean 复制到 RGB 三个通道上，形成与原图大小一致的三通道逻辑掩膜 mask3；接着在掩膜为 0 的位置，将原图对应像素置为黑色，只保留掩膜为 1 的苹果区域。这样即可从复杂背景中抠出苹果目标。

（2）篮球检索算法

本实验的目标是在包含多种球类的彩色图像中自动检测并分割出篮球。观察原始图像可知：一方面，所有球体都位于近似纯白的背景上，目标与背景在灰度亮度上存在明显差异；另一方面，篮球、足球、棒球等球体整体轮廓均近似圆形，而橄榄球等物体呈现细长椭圆形；此外，篮球表面具有若干条窄的黑色分隔线，但黑色区域并非大面积分布，这一点与足球的大面积黑色拼块、网球和棒球几乎没有黑色纹理的情况明显不同。基于上述先验，本算法采用“灰度阈值分割 + 几何形状筛选 + 黑色像素比例判别”的分层策略实现篮球检测与分割。

首先，在预处理阶段将输入彩色图像转换为灰度图，并采用高斯滤波对灰度图进行平滑处理，以抑制噪声和细小纹理对后续阈值分割的干扰。在此基础上，利用 Otsu 自适应阈值算法自动选取最佳分割阈值，将图像划分为前景和背景两类。由于背景区域整体较亮，而球体相对偏暗，故将灰度低于阈值的区域视为前景。随后，通过孔洞填充操作去除球内部的小空洞，再利用面积滤波去掉面积过小的孤立噪声，从而得到较为干净的前景掩膜图像，这一结果对应实验图中的“掩膜图像”。

在获得前景掩膜之后，算法对其进行连通域分析，将每个连通区域视为一个候选目标。针对每个连通区域，计算其面积、离心率以及外接矩形的宽高比等几何特征。面积特征用于排除残余的小碎片；离心率反映了区域的“圆度”，理想圆形的离心率接近 0，而细长形状（如橄榄球）的离心率接近 1；外接矩形的长宽比则用于衡量目标是否近似等宽等高。综合考虑这三个指标，本算法设置阈值条件：区域面积大于一定阈值（2500）、离心率小于给定上限（0.5）且长宽比小于1.25的连通域被判定为“近似圆形的球体”。经过这一筛选步骤，可以有效保留篮球、足球、棒球等圆球，同时剔除橄榄球等非圆形目标，实验图中的“检测圆形”即为这一阶段的可视化结果。

在识别出所有圆球候选后，仍需在其中判断哪一个是篮球。为此，本算法进一步利用灰度分布特性，对每个圆球区域内部的黑色像素比例进行统计。具体做法是：根据经验选取一个较低的灰度阈值（0.25），将灰度值低于该阈值的像素视为“黑色或明显偏暗像素”，计算其在整个球区域内所占的比例。由于篮球仅有几条细黑线，因此黑色像素比例处于一个中等范围；足球具有大面积黑色拼块，黑色比例明显偏高；而网球和棒球表面几乎没有黑色纹理，其黑色比例接近于零。基于这种差异，本算法在圆球候选中选取黑色比例处于预设区间的目标（[0.01,0.25]），并在其中选择黑色比例最大的一个作为最终的篮球目标。当某些特殊图像中未出现符合区间条件的球体时，算法退化为在所有圆球中选择黑色比例最高者，以提高鲁棒性。该步骤的输出对应实验图中的“选定篮球”结果。

最后，为了实现对篮球的分割与展示，算法利用前一步确立的篮球连通域索引，从其几何特征中提取外接矩形参数，构造包含篮球的最小矩形区域。基于这一外接框，在与原图同尺寸的全白图像上，将原图中对应位置的像素拷贝过来，其余区域保持白色，从而得到只保留篮球、背景为纯白的分割结果，即实验图中的“篮球分割图”。通过上述一系列步骤，算法在不依赖复杂机器学习模型的前提下，仅凭灰度分割、几何特征和简单的灰度统计便实现了对篮球目标的自动检测和提取，具有实现简单、计算量小和易于解释的特点。

**三、程序流程**

（1）红苹果提取算法流程图



图4-1红苹果提取算法流程图

（2）篮球检索算法流程图

****

图4-2 篮球检索算法流程图

**四、主要代码**

（1）红苹果提取算法代码

%% 1. 读入彩色图像并轻度去噪

RGB = im2double(imread('redapple.jpg'));

figure, imshow(RGB), title('原始苹果图像');

RGB\_smooth = imgaussfilt(RGB, 1); % % 高斯滤波，去除细小噪声

%% 2. RGB -> HSV，提取 H/S/V 通道

HSV = rgb2hsv(RGB\_smooth);

H = HSV(:,:,1); % Hue 色调 [0,1]

S = HSV(:,:,2); % Sat 饱和度 [0,1]

V = HSV(:,:,3); % Value 亮度 [0,1]

figure;

subplot(1,3,1), imshow(H), title('H 通道');

subplot(1,3,2), imshow(S), title('S 通道');

subplot(1,3,3), imshow(V), title('V 通道');

%% 3. 构造“红色 + 高饱和度”阈值，得到初始苹果分割图像

% 红色在 H 的两端（0~0.05 和 0.95~1）

H\_low1 = 0.00; H\_high1 = 0.08;

H\_low2 = 0.92; H\_high2 = 1.00;

S\_min = 0.3; % 饱和度下限

% 用 H + S，苹果因光照产生的阴影也能保留

mask\_red = ( (H >= H\_low1 & H <= H\_high1) | ...

(H >= H\_low2 & H <= H\_high2) ) ...

& (S >= S\_min);

figure, imshow(mask\_red), title('初始阈值分割');

imwrite(mask\_red, '初始阈值分割图像.png');

%% 4. 形态学处理：闭运算->填补苹果小洞->面积滤波

% 4.1 闭运算

SE = strel('disk', 5); % 半径3的圆盘结构元素，可调

BW\_dilate = imdilate(mask\_red, SE); % 膨胀

BW\_close = imerode(BW\_dilate, SE); % 腐蚀

% 4.2 填充内部孔洞

BW\_fill = imfill(BW\_close, 'holes'); % 填充苹果内部的小孔洞

% 4.3 去除面积过小的连通区域

minArea = 200; % 面积阈值

BW\_clean = bwareaopen(BW\_fill, minArea);

figure, imshow(BW\_clean), title('形态学闭运算苹果分割图像');

imwrite(BW\_clean, '苹果分割\_闭运算结果.png');

%% 5. 利用掩膜从原图中提取目标（抠出苹果）

mask3 = repmat(BW\_clean, [1 1 3]); % 和 RGB 一样大小的 3 通道逻辑矩阵

RGB\_apple = RGB; % 复制一份原始图像

RGB\_apple(~mask3) = 0; % 掩膜为 0 的位置全部置为黑色，只保留苹果

figure, imshow(RGB\_apple), title('提取出的苹果区域');

imwrite(RGB\_apple, '苹果提取结果.png');

（2）篮球检索算法代码

%% 1. 读入图像

I = im2double(imread('basketball1.jpg'));

Igray = rgb2gray(I);

[H, W, ~] = size(I);

% 高斯滤波

Igray\_smooth = imgaussfilt(Igray, 1);

%% 2. 前景分割

level = graythresh(Igray\_smooth); % Otsu 阈值法

fgMask = Igray\_smooth < level;

fgMask = imfill(fgMask, 'holes'); % 填补空洞

fgMask = bwareaopen(fgMask, 2000); % 去除小区域

figure; imshow(fgMask); title('前景掩模');

imwrite(fgMask,'前景掩膜.png');

%% 3. 连通域分析：筛选近似圆形区域

[L, num] = bwlabel(fgMask); % 对前面的联通区域进行标记

% 对连通区域计算几何特征

stats = regionprops(L, 'Area','Eccentricity','BoundingBox','PixelIdxList');

isCircle = false(num,1); % 逻辑数组：标记第 k 个连通区域是不是近似圆形的球

for k = 1:num

A = stats(k).Area; % 区域面积

ecc = stats(k).Eccentricity; % 离心率，越接近0越圆

bb = stats(k).BoundingBox; % 得到外接矩形：[x, y, width, height]

w = bb(3); h = bb(4);

ratio = max(w,h) / min(w,h); % 长宽比

% 判定条件：1）面积足够大2）离心率低（不细长）3）长宽比接近 1（圆形）

if A > 2500 && ecc < 0.50 && ratio < 1.25

isCircle(k) = true;

end

end

% 可视化：蓝色框表示“圆球候选”

figure; imshow(I); title('检测到的圆球'); hold on;

for k = 1:num

if isCircle(k)

rectangle('Position', stats(k).BoundingBox, ...

'EdgeColor','b','LineWidth',2);

end

end

hold off;

%% 4. 对每个圆球计算黑色像素比例

blackRatio = zeros(num,1);

blackTh = 0.25; % 灰度< 0.25视为“黑色像素”

for k = 1:num

if ~isCircle(k), continue; end

pixelIdx = stats(k).PixelIdxList;

end

%% 5. 根据黑色比例选择篮球（有黑线但不大面积黑色）

minR = 0.01;

maxR = 0.25;

candidates = find(isCircle & blackRatio >= minR & blackRatio <= maxR);

if isempty(candidates) % 没候选，选圆球中黑比例最大的那个

circleIdx = find(isCircle);

[~, ord] = sort(blackRatio(circleIdx), 'descend');

idxBall = circleIdx(ord(1));

else % 从候选球中黑比例最大者即为篮球

[~, m] = max(blackRatio(candidates));

idxBall = candidates(m);

end

maskBall = (L == idxBall); % 用于显示轮廓（红色）

figure; imshow(I); hold on;

visboundaries(maskBall, 'Color', 'r', 'LineWidth', 2);

title('篮球');

hold off;

imwrite(I,'篮球.png');

%% 6. 抠图

bb = stats(idxBall).BoundingBox; % [x, y, w, h]

% 转换为整数像素范围

x1 = round(bb(1));

y1 = round(bb(2));

x2 = x1 + round(bb(3)) - 1;

y2 = y1 + round(bb(4)) - 1;

% 边界检查

x1 = max(1, x1);y1 = max(1, y1);x2 = min(W, x2); y2 = min(H, y2);

ball\_only = ones(H, W, 3);

% 将篮球区域从原图复制到新图

ball\_only(y1:y2, x1:x2, :) = I(y1:y2, x1:x2, :);

figure; imshow(ball\_only);title('分割篮球图');

imwrite(ball\_only,'分割篮球图.png');

grayVals = Igray(pixelIdx);

blackRatio(k) = sum(grayVals < blackTh) / numel(grayVals);

fprintf('球 %d：黑色比例 = %.3f\n', k, blackRatio(k));

**五、运行结果**

（1）红苹果提取实验结果

图4-3为三幅苹果样本图像在 HSV 色彩空间中的通道分解图。图4-4展示了基于 HSV 颜色与形态学处理的苹果分割全过程结果。每一行分别对应三幅不同场景的苹果图像：(a) 背景相对简单、苹果数量较少；(b) 果实密集、叶片较多；(c) 背景较亮、存在较多高光与杂色干扰。每一行由左到右依次为：原始图像、初始阈值分割结果、形态学闭运算与填洞后的二值图以及最终抠出的苹果区域。

1. 原始图像

第一列为三幅原始苹果图像，可以看到：(a) 中苹果颜色鲜红、与绿色叶片对比明显，背景相对单一；(b) 中存在多串苹果聚集，且部分果实与背景叶片、枝干存在遮挡；(c) 中背景较亮，局部叶片及枝条上也存在接近红色或棕红色的区域。这些差异为后续分割带来了不同程度的难度，尤其是 (b)、(c) 中红色或高饱和区域并不全部是目标苹果。

|  |  |
| --- | --- |
| (a) | apple_1 |
| (b) | apple_2 |
| (c) | apple_3 |

图4-3 HSV 色彩空间三通道转换图

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (a) | redapple | 初始阈值分割图像 | 苹果分割_闭运算结果 | 苹果提取结果 |
| (b) | redapple1 | 初始阈值分割图像 | 苹果分割_闭运算结果 | 苹果提取结果 |
| (c) | redapple2 | 初始阈值分割图像 | 苹果分割_闭运算结果 | 苹果提取结果 |

图4-4 红苹果提取实验结果图

1. 基于 HSV 的初始阈值分割

第二列为利用 H 通道红色区间 + S 通道下限阈值得到的初始二值掩膜 mask\_red：对三幅图像而言，绝大部分苹果区域已经被成功标记为白色前景，说明 HSV 阈值对红苹果具有较好的区分能力。

但可以观察到：(a) 中苹果内部存在小黑洞，边缘略有缺口，说明单纯颜色阈值仍然受亮度变化、高光和阴影的影响；(b) 中除了苹果本身，存在较多零散白点；(c) 中苹果边缘有一定锯齿现象。这一阶段的结果表明，仅采用 H+S 阈值能够粗略区分苹果与背景，但仍存在边界不连续、内部孔洞以及小块噪声等问题。

1. 形态学闭运算与孔洞填充

第三列为对初始掩膜执行膨胀+腐蚀（闭运算）并进行孔洞填充后的结果 BW\_fill（结合 bwareaopen 去小碎块后为 BW\_clean 的中间效果）：

在 (a) 中，闭运算有效弥合了苹果轮廓上的小缝隙，多个苹果边界更加平滑连续；孔洞填充后，苹果内部由于被遮挡的黑洞被填充为白色，使单个苹果区域呈现为完整实心的连通块；在 (b) 中，由于果实密集，闭运算在连接紧邻的苹果区域的同时，也进一步抹平了内部的小缺口，使每一簇苹果形成较大的连通区域，同时，部分零散的小白点在后续面积滤波中被删除，背景噪声明显减少；在 (c) 中，苹果主区域轮廓得到修复，内部孔洞基本消失。整体来看，形态学闭运算+填洞显著改善了目标区域的形状完整性，减少了由于光照和纹理导致的“断裂”和“空洞”。

1. 面积滤波与最终苹果抠图结果

第四列为利用 BW\_clean 掩膜从原图中抠出的苹果区域 RGB\_apple：

1. 中背景几乎全部被置为黑色，说明在背景简单的情况下，该方法能够实现较为理想的目标提取；(b) 中多串苹果整体被保留，绝大部分背景叶片与枝干被抑制，少数被叶片大部分遮挡的苹果也能捕捉到；(c) 中苹果主体被成功抠出，高光影响被显著抑制。

综合三幅图像的结果，可以看出：HSV 阈值分割能够较为稳健地提取红色苹果的大致区域；形态学闭运算 + 孔洞填充有效修复了目标区域的边缘与内部结构，使苹果区域连通、饱满；面积滤波 bwareaopen进一步抑制了小碎块噪声，尤其在背景较为复杂的图像 (b)、(c) 中对提升分割质量起到了关键作用，可见其在红色苹果目标分割任务中的有效性与一定的鲁棒性；但在背景中存在与苹果颜色、面积相近的非目标区域时，算法仍会产生少量误检。

1. 篮球检索算法实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| basketball1 | 前景掩膜 | b_3 | b_4 | 分割篮球图 |
| basketball2 | 前景掩膜 | b_1 | b_2 | 分割篮球图 |
| basketball3 | 前景掩膜 | b_5 | b_6 | 分割篮球图 |
| (a)原始图像 | (b)掩膜图像 | (c)检测圆形 | (d)篮球判别 | (e)篮球分割图 |

图4-5 篮球检索算法实验结果图

篮球检索算法实验结果如图4-5所示。图（a）为三组不同原始图像；图 (b) 为各自掩膜图像，基本完整覆盖了各类球体，，但对后续连通域特征筛选影响较小；图（c）为各自圆形筛选结果图，利用面积、离心率和外接矩形长宽比联合判别，可以有效区分圆球和橄榄球等非圆球目标，掩膜图像中所有圆形球均被蓝色框正确选出；图（d）为各自篮球判别结果图，由于黑色像素比例在不同球类之间呈现出明显差异，因此在圆球候选中引入黑色比例阈值能有效帮助区分篮球与其他球类，三幅测试图像中均能正确定位出目标篮球，说明该判别策略具有较好的鲁棒性；图（e）为各自的抠图结果图，通过外接矩形进行抠图，可以在保持原图大小的同时，仅保留篮球区域。

总体来看，本算法无需复杂的训练过程，只依赖灰度分割、几何形状特征与简单的灰度统计，即可在多种场景下稳定地检测并分割出篮球，具有实现简单、计算开销小、解释性强等优点。

1. **实验心得**

在苹果分割实验中，我主要采用了“颜色空间 + 形态学”的方案。实践中可以明显感觉到：若直接在 RGB 空间做阈值分割，阈值很难兼顾不同光照和阴影；而将图像转换到 HSV 空间后，只要合理设置 H 的红色取值范围，并给出 S 的下限，就能比较稳定地把苹果从背景中分离出来。随后，通过闭运算、填洞和面积滤波等形态学操作，又把边缘缝隙、小孔洞和零碎噪声一步步“修干净”，得到轮廓完整的苹果区域。同时也发现，当背景中存在大面积与苹果颜色接近的区域时，仅凭颜色和面积仍然会产生误检，说明后续可以考虑结合形状特征或引入简单的学习方法来进一步提升鲁棒性。

在篮球检测与分割实验中，我采用了“灰度阈值分割 + 几何特征筛选 + 黑色像素比例判别”的组合算法。整体实现过程比预想中顺利，但在参数选择和鲁棒性上也有不少体会。一开始我尝试仅靠颜色或纹理来区分篮球与其他圆球，效果并不稳定；改用“黑色像素比例”这一简单统计特征后，反而取得了较好的结果：足球具有大面积黑色区域，黑色占比明显偏高；网球和棒球几乎没有黑色，比例接近 0；篮球则处在一个中等区间。通过给黑色比例设定合理的上下限，并在圆球候选中选择黑色比例最大的一个，就可以在多幅测试图像中稳定地定位篮球。这让我意识到，有时候“针对任务设计一个合适的特征”，比一味追求复杂模型更有效。

总体来说，这两次实验强化了我对“分步式图像处理”思想的理解：先通过阈值分割在合适的颜色或灰度空间中粗略分离前景，再利用几何形状和形态学运算逐步剔除干扰，最后通过针对性的统计或判别特征完成精细分类。虽然当前方法仍然依赖于背景较为单一、目标外观特征较明显等前提，对更复杂场景的适应性有限，但在本次实验条件下已经取得了较高的准确率，也让我对颜色空间变换、阈值分割、形态学处理和特征设计的具体用法与调参思路有了更直观的工程体验，为后续在 HSV 空间或机器学习框架下进一步改进算法打下了基础。