1229 上课内容(2)

Matlab 图像处理

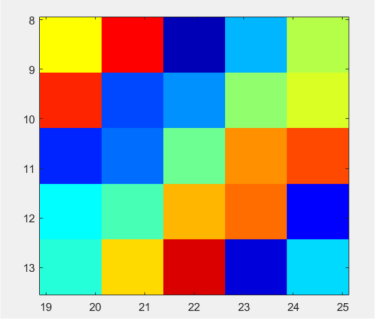
表达方式

- 像素索引：最为方便和直观的方法，图像被视为离散单元，按照空间顺序从上到下、从左到右排列，像素索引值为正整数
- 空间位置：将图像和空间位置联系，与像素索引没有实质区别，但使用空间位置连续值代替像素索引离散值进行表示

```
A=magic(5);
x=[19.5 24.5];
y=[8.5 13.0];
image(A,'XData',x,'YData',y);
axis image;
colormap(jet(25));
```

```
A =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```



✓ 利用空间位置表达进行图像构建和展示

```
>> help image
 image - 从数组显示图像
    此 MATLAB 函数 会将数组 C 中的数据显示为图像。C 的每个元素指定图像的 1 个像素的颜
    色。生成的图像是一个 m×n 像素网格，其中 m 和 n 分别是 C 中的行数和列数。这些元素的行
    索引和列索引确定了对应像素的中心。
```

36

图像的存储方式

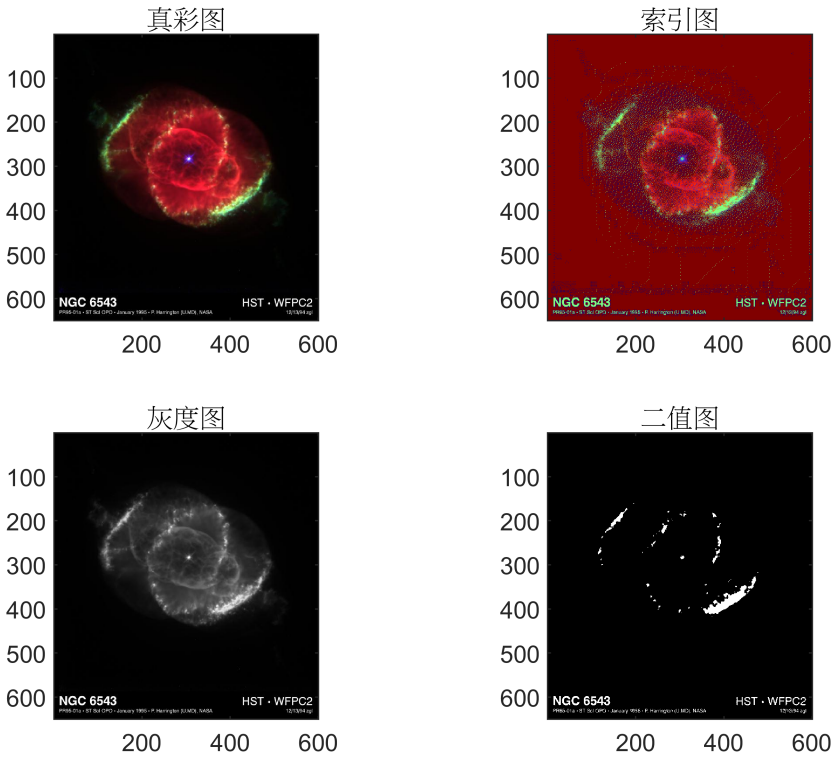
图像储存方式：

- 亮度图像：即灰度图像，使用二维矩阵进行储存，每个值为灰度信息，一般来说：0-黑色、255-白色
- RGB图像：使用3个一组数据表达每个像素的颜色，包含红绿蓝三个分量，储存在一个m×n×3的三维数组中
- 索引图像：包含两个数组，一个是图像数据矩阵，一个是颜色索引表；对应于图像中每个像素，数据数组都包含一个指向颜色索引表的索引值
- 二值图像：像素颜色两种取值（黑或白），储存为一个二值矩阵，包含0/1元素

一些可用的转换函数
gray2ind：灰度图像转索引图像　　grayslice：使用阈值法从灰度图像建立索引图
mat2gray：归一化方法将矩阵转为灰度图　　rgb2gray：RGB图转为灰度图
Rbg2ind：从RBG图像建立索引图

```
close all;
% 真彩图像转换为索引图像
RGB = imread('ngc6543a.jpg'); % 内置图像
map = jet(256);
X = dither(RGB,map);
subplot(2,2,1);subimage(RGB);
title('真彩图');
```

```
subplot(2,2,2);subimage(X,map);
title('索引图');
% 彩色图转换为灰度图
I = rgb2gray(RGB);
subplot(2,2,3);subimage(I);
title('灰度图');
% 彩色图转换为二值图像
BW = im2bw(RGB,0.5);
subplot(2,2,4);subimage(BW);
title("二值图");
```



图像的显示

图像的显示：

- 图像的显示过程是将数字图像从以为离散数据还原为一副可见图像的过程
- MATLAB中使用image函数创建一个句柄图形图像对象，包含设置该对象的各种属性调用方法，类似函数是imagesc（实现自动缩放）
- 还提供了imshow函数，创建句柄图形图像对象，同样包含句柄属性和图像特征

imshow函数：

```
>> help imshow
imshow - 显示图像
    此 MATLAB 函数 在图窗中显示灰度图像 I。imshow 使用图像数据类型的默认显示范围，并优化
    图窗、坐标区和图像对象属性以便显示图像。
```

常见用法：

imshow(I)：显示数据矩阵I的灰度图像，I也可以是文件名字

imshow(X, map)：显示索引图像，X为索引图像的数据矩阵，map为所以图像对应的颜色映射矩阵

另外，可使用Name和Value对于属性进行设置

```
I = imread('lena.bmp');
```

```
错误使用 imread>get_full_filename
文件 "lena.bmp" 不存在。

出错 imread (第 371 行)
    fullname = get_full_filename(filename);
```

```
figure;
subplot(1,2,1);imshow(I);
% 采用默认的灰度级显示灰度图像
subplot(1,2,2);imshow(I,[60,120]);
% 设置灰度上下为[60,120]显示灰度图像
```

image 和 imagesc 函数

image和imagesc函数：

- 两个函数的功能域imshow类似，可以显示图像并设置一些属性

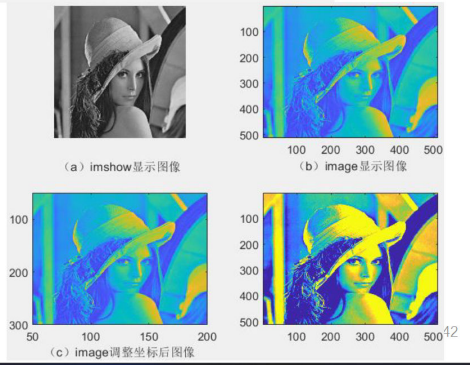
常见用法：

image(C)：显示数据矩阵C代表的图像

imagesc(x, y, C)：显示数据矩阵C的图像，并设置显示坐标轴的范围

```
I=imread('lena.bmp');
figure;
subplot(221);imshow(I);  %imshow函数显示
xlabel('（a）imshow显示图像');
subplot(222);image(I);  %image函数显示
xlabel('（b）image显示图像');
subplot(223);image([50,200],[50,300],I);  %image函数绘制调整坐标后的图像
xlabel('（c）image调整坐标后图像');
subplot(224);imagesc(I,[60,150]);  %利用imagesc显示经过灰度拉伸后的图像
```

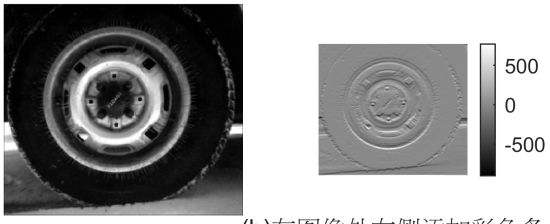
使用示例



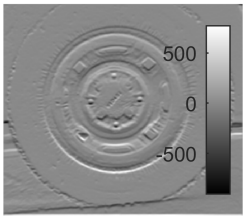
colorbar

添加一个彩色条，用来指示图像中不同颜色所对应的具体数值

```
close all;
I = imread('tire.tif');
H = [1,2,1;0,0,0;-1,-2,-1]; % 设置 sobel 算子
X = filter2(H,I); % 对灰度图像 I 进行 2 次滤波，实现边缘检测
subplot(1,3,1);imshow(I); % 原始图像
xlabel('(a)原始图像');
subplot(1,3,2);imshow(X,[]);
colorbar; % 默认位置
xlabel('(b)在图像外右侧添加彩色条');
subplot(1,3,3);imshow(X,[]);
colorbar('east'); % 在图像内右侧添加彩色条
xlabel('(c)在图像内右侧添加彩色条');
```



(a)原始图像　　(b)在图像外右侧添加彩色条



(c)在图像内右侧添加彩色条

```
% 设置 sobel 算子进行边缘检测
% 添加了对于图片中颜色值展示 colorbar
```

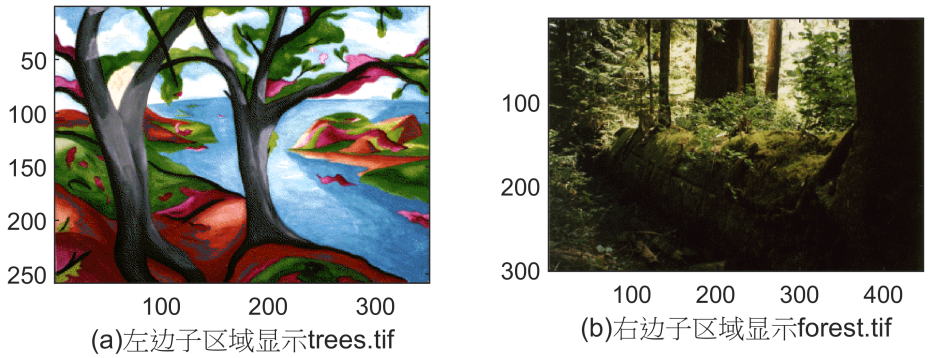
subimage 函数

将多个图像显示一个图形窗口中，方便进行比较和查看

常见用法：

subimage(X, map), subimage(I或BW或RGB)：分别对应索引图像，灰度图像、二值图像和真色彩图像

```
close all;
load trees;
[X2,map2] = imread('forest.tif');
subplot(1,2,1);subimage(X,map);
xlabel('(a)左边子区域显示 trees.tif');
subplot(1,2,2);subimage(X2,map2);
xlabel('(b)右边子区域显示 forest.tif');
```



图像的运算

- 图像的运算指的是多幅图像的数学运算，包括代数、逻辑、几何等

代数运算：

对于多幅图像进行加、减、乘、除和一般的线性运算，如检测运动物体的出现

常见用法：

Z = imadd(X, Y)：将X和Y相加得到Z，其中X、Y的维度相同（或者为常数）

Z = imsubtract(X, Y)：X中每个元素减去Y对应元素，得到Z

Z =immultiply(X, Y)：X和Y中每个对应元素相乘，得到Z

Z = imdivide(X, Y)：X中每个元素除以Y中对应位置，返回Z

Z = imabsdiff(X, Y)：X和Y中对应位置元素相减，并取绝对值得到Z

IM2 = imcomplement(IM)：对于图像矩阵IM中所有元素求补，得到Z（二值图像）

Z = imlincomb(K1, A1, K2, A2, ...)：计算图像矩阵的加权和，线性组合运算

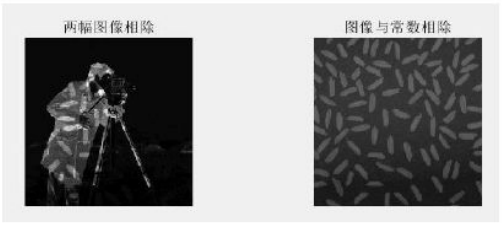
图像的运算例子

图像的运算-示例：

```
I = imread('rice.png');
J = imread('cameraman.tif');
subplot(131);imshow(I);
xlabel('(a) rice.png图像');
subplot(1,3,2);imshow(J);
xlabel('(b) cameraman.tif图像');
K = imadd(I,J,'uint16');
subplot(1,3,3);imshow(K,[]);
xlabel('(c) 图像的相加效果')
```

此处的rice和cameraman，进行相加也都是图像处理中常见的基准图像

```
I = imread('rice.png');
J = imread('cameraman.tif');
subplot(221);imshow(I);
subplot(222);imshow(J);
Ip=imdivide(I,J);  %两幅图像相除
subplot(223);imshow(Ip,[]);
K=imdivide(I,2);  %图像跟一个常数相除
subplot(224);imshow(K);
```



47

逻辑运算：

主要针对二值图像，以图像像素作为对象进行两幅或多幅图像间的操作

常见运算仍然是与、或、非、与非、异或等

```
I = imread('rice.png');
J = imread('cameraman.tif');
I1=im2bw(I);  %转化为二值图像
J1=im2bw(J);
K1=I1&J1;  %实现图像的逻辑“与”运算
K2=I1|J1;  %实现图像的逻辑“或”运算
K3=~I1;  %实现图像的“非”运算
K4=xor(I1,J1);  %实现图像的“异或”运算
subplot(231);imshow(I1);xlabel('(a) 二值图像1');
subplot(232);imshow(J1);xlabel('(b) 二值图像2');
subplot(233);imshow(K1);xlabel('(c)逻辑“与”运算');
subplot(234);imshow(K1);xlabel('(d)逻辑“或”运算');
subplot(235);imshow(K1);xlabel('(f)逻辑“非”运算');
subplot(236);imshow(K1);xlabel('(e)逻辑“异或”运算');
```

注意：先转换为二值图像(im2bw)

这里默认需要的操作是两张图片的**大小需一致**（可能需要裁剪）

48

几何运算：

引起图像几何形状发生变化的变换，包括插值、缩放、旋转、剪切等

插值：常见的数学运算，利用曲线拟合方法，通过离散采样点建立连续函数逼近真实曲线，再利用重建的函数求任意位置的函数值

MATLAB中可以通过插值，实现图像的旋转和缩放，如线性最近邻插值，在输入图像中插入与其最近邻的采样点的值，运算量很小；该方法能一定程度上提升图像的分辨率，但是有人工操作的痕迹

二维图像插值函数interp2

ZI = interp2(X, Y, Z, XI, YI, method) X、Y是图像Z的很坐标和纵坐标向量，XI和YI是插值后的横纵坐标向量，method是插值方法（包括nearest-线性最近邻插值、linear-线性插值、spline-三次样条插值）

49

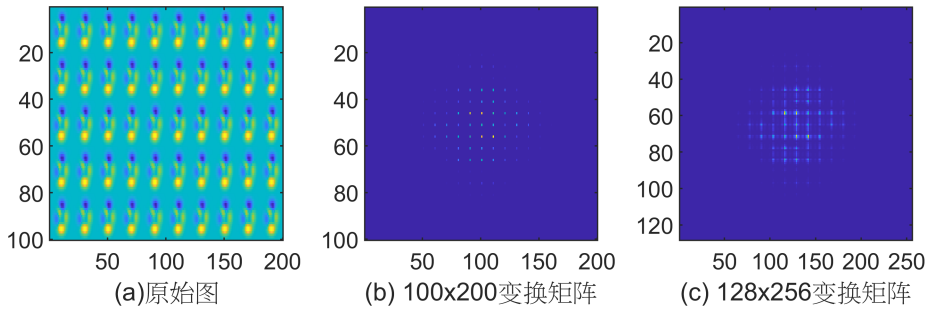
其余运算可以查 ppt

图像处理

傅里叶变换

```
close all;
P = peaks(20);
X = repmat(P,[5,10]);
subplot(1,3,1);imagesc(X);axis square;
xlabel('(a)原始图');
Y = fft2(X); % 傅里叶频率平移后得到的结果
subplot(1,3,2);imagesc(abs(fftshift(Y)));axis square;
xlabel('(b) 100x200 变换矩阵');
Y = fft2(X,2^nextpow2(100),2^nextpow2(200)); % 傅里叶频率平移后得到的结果
subplot(1,3,3);imagesc(abs(fftshift(Y)));axis square;
```

```
xlabel('(c) 128x256 变换矩阵');
```



(a)原始图 (b) 100x200变换矩阵 (c) 128x256变换矩阵

图像增强示例

- 图像增强示例：对于灰度图像进行灰度值调整，实现视觉效果提升

```
pout = imread('pout.tif');
pout_imadjust = imadjust(pout);
pout_histeq = histeq(pout);
pout_adapthisteq = adapthisteq(pout);
subplot(121);imshow(pout);
xlabel('(a)原始图像');
subplot(122), imshow(pout_imadjust);
xlabel('(b)调整值');
```

- 图像滤波：也是图像增强的一种技术，对于一幅图像进行滤波，可以强调一些特征，去掉另一些特征，实现图像的光滑，锐化和边缘检测
  图像的滤波是一种邻域操作，输出图像的像素值是对于输入图像相应像素的邻域值进行一定的处理得到的，可选的操作有线性滤波、空间域滤波，也可以自定义滤波器

56

图像滤波示例

- 图像滤波示例：对图像添加不同的滤波器进行邻域平均处理

```
I = imread('cameraman.tif');
subplot(2,2,1); imshow(I);
xlabel('(a)原始图像');
H = fspecial('motion',20,45);
MotionBlur = imfilter(I,H,'replicate');
subplot(2,2,2);imshow(MotionBlur);
xlabel('(b)运动滤波器');
H = fspecial('disk',10);
blurred = imfilter(I,H,'replicate');
subplot(2,2,3); imshow(blurred);
xlabel('(c)图形均值滤波器');
H = fspecial('unsharp');
sharpened = imfilter(I,H,'replicate');
subplot(2,2,4); imshow(sharpened);
xlabel('(d)掩模滤波器');
```

- ✓ 对于同一张输入图片，选择不同的滤波器能够获得不同的处理图像
- ✓ 注意能够获取不同的信息（如轮廓信息、细节信息、模糊处理等）

57

图像的其它处理

图像的其他处理：

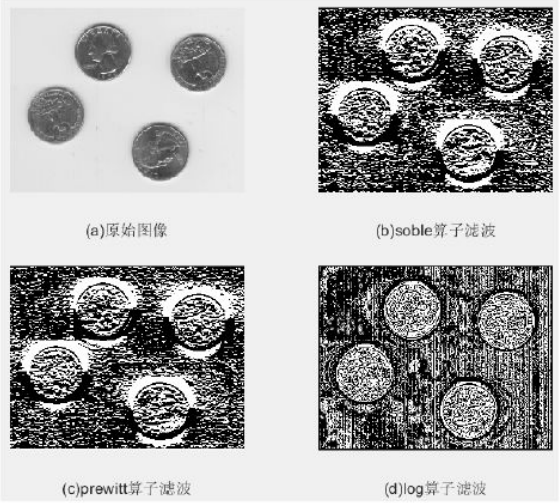
- 图像的边界：能够实现图像边缘检测、边界跟踪、Hough变换检测图像中的直线等；
- 这里简介图像边缘检测的步骤：
  （1）平滑滤波：噪声会影响梯度计算，通过滤波去除噪声，但是减低噪声的平滑能力越强则边界强度损失越大；
  （2）锐化滤波：须确定某点邻域中灰度的变化，锐化的目的是加强有意义的灰度局部变化位置像素点；
  （3）边缘判定：图像中存在很多梯度不为0的点，这里并非所有点都有意义，根据具体情况选择和去除部分点（如二值化处理、过零点检测等）；
  （4）边缘连接：将间断的边缘连接成有意义的完整边缘，去除假边缘，可使用Hough变换等方法

边缘检测能够辅助进行碰撞检测、确定有效区域，辅助进行复杂的图像处理任务

图像的其他处理：

- 图像边界示例：

```
I=imread('eight.tif');
subplot(221);imshow(I);
xlabel('(a)原始图像');
h1=fspecial('sobel');
I1=filter2(h1,I);
subplot(222);imshow(I1);
xlabel('(b)soble算子滤波');
h2=fspecial('prewitt');
I2=filter2(h2,I);
subplot(223);imshow(I2);
xlabel('(c)prewitt算子滤波');
h3=fspecial('log');
I3=filter2(h3,I);
subplot(224);imshow(I3);
xlabel('(d)log算子滤波');
```



- ✓ 不同算子获得的滤波（边缘检测）结果略有差异，其性能与输入图像特征有一定的关系
- ✓ 获得边缘信息后，能够进行进一步信息处理

59