Chap 10　图像处理、图论、金融

图像处理

MATLAB中的图像处理工具箱提供了一套全方位的标准算法和图形工具，用于进行图像处理、分析、可视化和算法开发。可用其对有噪图像或退化图像进行去噪或还原、增强图像以获得更高的清晰度、提取特征、分析形状和纹理，以及对两个图像进行匹配。工具箱中的大部分函数均以开放式MATLAB语言编写，这意味着可以检查算法、修改源代码和创建自定义函数。

图像处理工具箱在医学、机器视觉、遥感、监控、基因表达、显微镜技术、半导体测试、图像传感器设计、颜色科学及材料科学等领域，为工程师和科学家提供支持，同时也促进了图像处理技术的教学。

面临的问题有：

- 读写各种文件格式
- 创建和测试算法
- 查找错误
- 查看和解释计算结果
- 内存的利用
- 计算速度的提高

文件的读写

图像处理工具箱支持多种设备生成的图像，包括数码相机、图像采集系统、卫星和空中传感器、医学成像设备、显微镜、望远镜和其他科学仪器。用户可以用多种数据类型来可视化、分析和处理这些图像，包括单精度和双精度浮点和有符号或无符号的8、16和32位整数。

在MATLAB环境中，导入或导出图像进行处理的方式有几种。用户可以使用图像采集工具箱从Web　摄像头、图像采集系统、DCAM兼容相机和其他设备中采集实时图像。另外通过使用数据库工具箱，用户可以访问ODBC/JDBC兼容数据库中存储的图像。

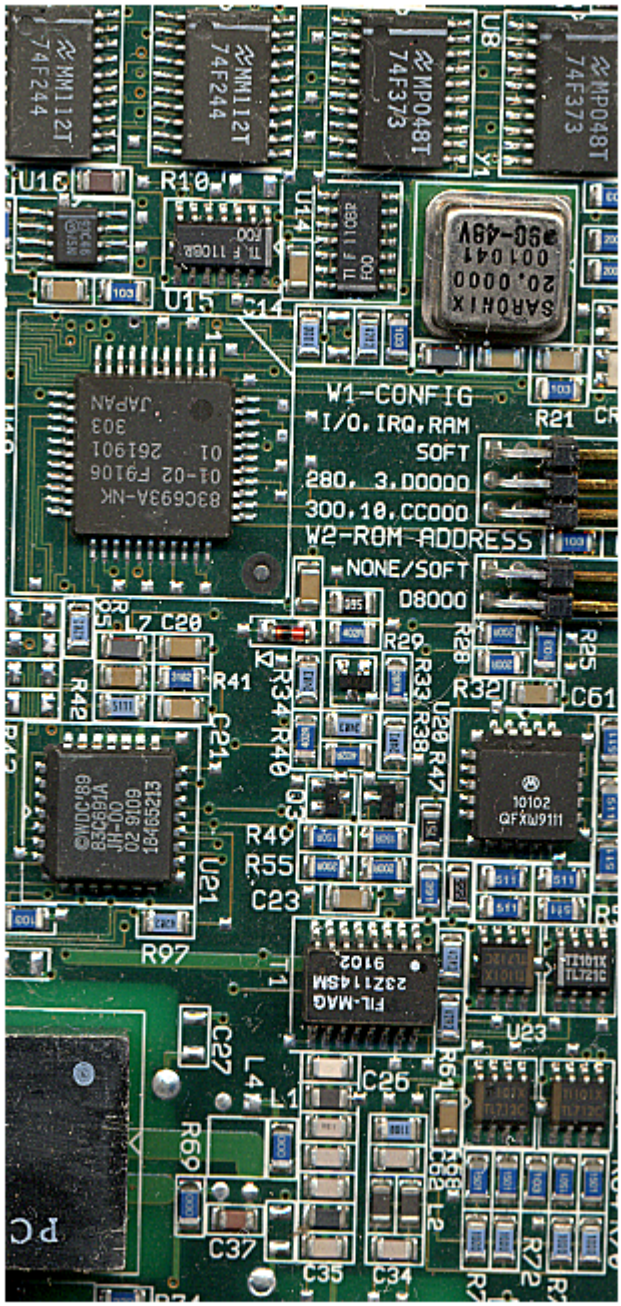
MATLAB支持标准数据和图像格式，包括JPEG、TIFF、PNG、HDF、HDF-EOS、FITS、Microsoft Excel、ASCII和二进制文件等。它还支持多频带图像格式，如LANDSAT。同时MATLAB提供了低级I/O函数，可以让用户开发用于处理任何数据格式的自定义程序。图像处理工具箱支持多种专用图像文件格式。例如对于医学图像，它支持DICOM文件格式，包括关联的元数据，以及Analyze 7.5和Interfile格式。另外此工具箱还可以读取NITF格式的地理空间图像和HDR格式的高动态范围图像等。

图像文件的读取：

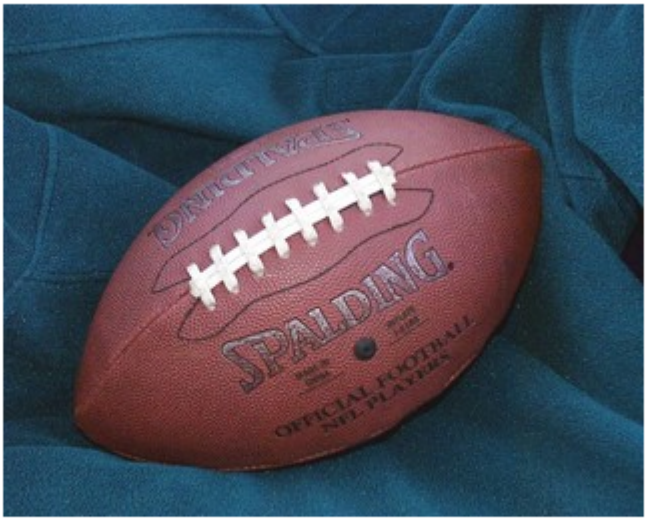
```
RGB = imread('football.jpg');                    %  读入jpg图像文件到RGB
[X,map] = imread('trees.tif');        %  读入tif文件到[X,map]
```

图像文件的显示

```
imshow('board.tif')
```

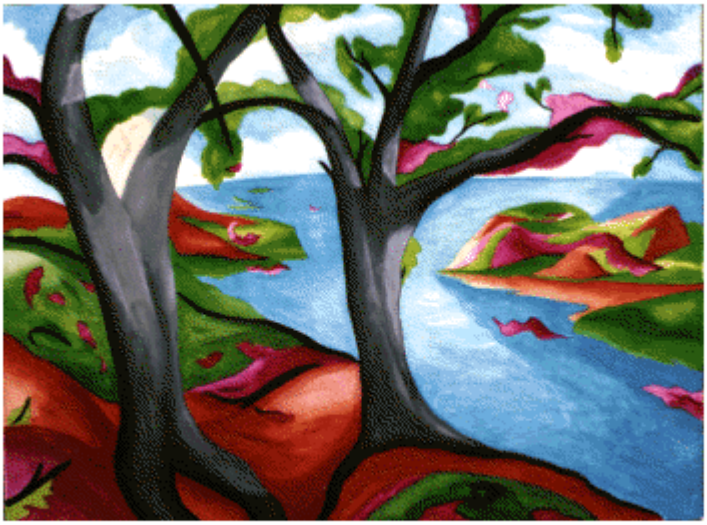



```
imshow(RGB)
```




```
imshow(X)
```

```
imshow(X,map)
```



图像的写入

```
imwrite(RGB,'footballtemp.jpg')              %  写入jpg文件
imwrite(X,map,'treestemp.tif')               %  写入tif文件
```

在MATLAB中，使用函数imfinfo能够获得图像处理工具箱所支持的任何格式的图像文件的信息，其调用语法为：

```
info = imfinfo(filename,fmt)
```

info = imfinfo(filename)

```
info = imfinfo('canoe.tif')                        %  canoe.tif 是系统自带的图片
```

```
info = struct with fields:
                    Filename: 'C:\Program Files\MATLAB\R2017b\toolbox\images\imdata\canoe.tif'
                FileModDate: '13-四月-2015 01:23:12'
                    FileSize: 71548
                      Format: 'tif'
                FormatVersion: []
                       Width: 346
                      Height: 207
                    BitDepth: 8
                   ColorType: 'indexed'
             FormatSignature: [73 73 42 0]
                   ByteOrder: 'little-endian'
              NewSubFileType: 0
               BitsPerSample: 8
                 Compression: 'PackBits'
    PhotometricInterpretation: 'RGB Palette'
                StripOffsets: [8 7986 15905 23749 31644 38954 45750 53036 60499]
             SamplesPerPixel: 1
```

```
              RowsPerStrip: 23
           StripByteCounts: [7978 7919 7844 7895 7310 6796 7286 7463 7410]
               XResolution: 72
               YResolution: 72
            ResolutionUnit: 'Inch'
                  Colormap: [256×3 double]
        PlanarConfiguration: 'Chunky'
                 TileWidth: []
                TileLength: []
               TileOffsets: []
            TileByteCounts: []
               Orientation: 1
                 FillOrder: 1
          GrayResponseUnit: 0.0100
            MaxSampleValue: 255
            MinSampleValue: 0
              Thresholding: 1
                    Offset: 69708
          ImageDescription: 'Copyright The MathWorks, Inc.'
```

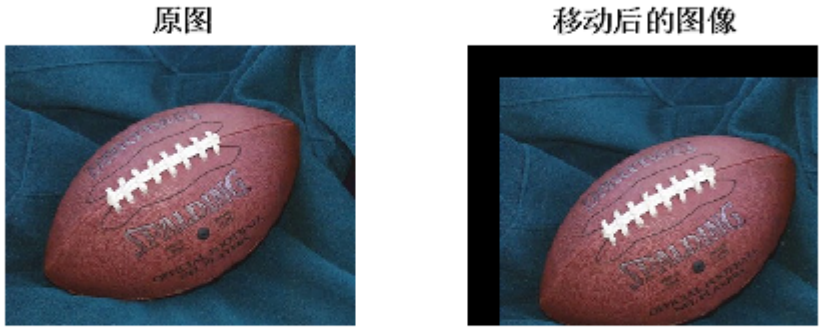
```
info1 = imfinfo('treestemp.tif')
```

```
info1 = struct with fields:
                      Filename: 'C:\Program Files\MATLAB\R2017b\bin\treestemp.tif'
                   FileModDate: '02-十二月-2017 20:53:20'
                      FileSize: 75764
                        Format: 'tif'
                 FormatVersion: []
                         Width: 350
                        Height: 258
                      BitDepth: 8
                     ColorType: 'indexed'
                FormatSignature: [73 73 42 0]
                     ByteOrder: 'little-endian'
                 NewSubFileType: 0
                  BitsPerSample: 8
                    Compression: 'PackBits'
    PhotometricInterpretation: 'RGB Palette'
                   StripOffsets: [8 6477 12250 19033 25262 32340 39385 46037 52664 59506 65724 72375]
                SamplesPerPixel: 1
                   RowsPerStrip: 23
                StripByteCounts: [6469 5773 6783 6229 7078 7045 6652 6627 6842 6218 6651 1555]
                    XResolution: 72
                    YResolution: 72
                 ResolutionUnit: 'Inch'
                       Colormap: [256×3 double]
             PlanarConfiguration: 'Chunky'
                      TileWidth: []
                     TileLength: []
                    TileOffsets: []
                 TileByteCounts: []
                    Orientation: 1
                      FillOrder: 1
               GrayResponseUnit: 0.0100
                 MaxSampleValue: 255
                 MinSampleValue: 0
                   Thresholding: 1
                         Offset: 73930
```

图像的平移

在MATLAB中，可以使用函数translate来实现图像的平移。其调用语法为：SE2 = translate(SE,V)。其中，SE为一个模板，使用函数strel来创建，V是一个向量，用来指定平移的方向。

```
I = imread('football.jpg');
se = translate(strel(1), [30 30]);    %  向下和向右移动30个位置
J = imdilate(I,se);                    %  利用膨胀函数平移图像
subplot(121);imshow(I), title('原图')
subplot(122), imshow(J), title('移动后的图像');
```



图像的镜像变换

图像的镜像变换分为两种：一种是水平镜像，另一种是垂直镜像。图像的水平镜像操作是将图像左半部分和右半部分以图像垂直中轴线为中心镜像进行变换，图像的垂直镜像操作是将图像上半部分和下半部分以图像水平中轴线为中心镜像进行变换。

对图像进行水平镜像和垂直镜像变换是通过对图像的像素数据做变换实现的。使用函数fliplr和flipud对像素矩阵进行水平和垂直反转，就可以完成图像的镜像变换。

```
I = imread('cameraman.tif');
Flip1=fliplr(I);                %  对矩阵I左右反转
subplot(131);imshow(I);title('原图');
```

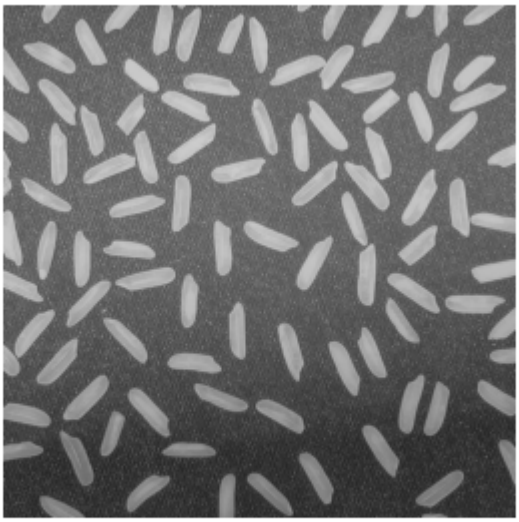

```
subplot(132);imshow(Flip1);title('水平镜像');
Flip2=flipud(I);                %  对矩阵I垂直反转
subplot(133);imshow(Flip2);title('竖直镜像');
```

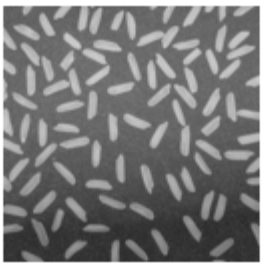

```

```

图像缩放

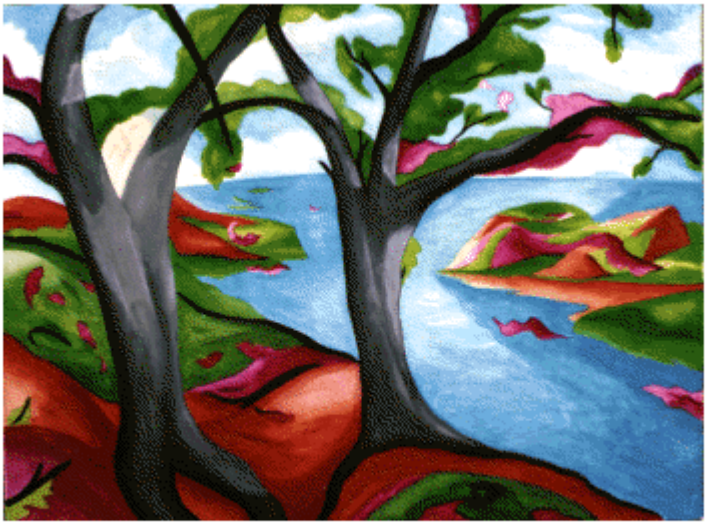
一般的方法是直接赋值为和它最相近的像素值，也可以通过一些插值算法来计算。后者处理的效果要好些，但是运算量也相应地会增加很多。MATLAB提供了imresize函数用于改变图像的尺寸。

```
I = imread('rice.png');
J = imresize(I, 0.5);                    %  缩小
figure, imshow(I), figure, imshow(J)
```

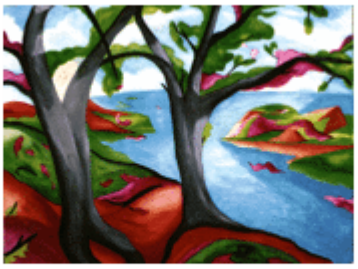




```
[X, map] = imread('trees.tif');
[Y, newmap] = imresize(X, map, 0.5);       %  索引图像的缩小
figure, imshow(X,map)
```



```
figure, imshow(Y, newmap)
```



```

```

图像的旋转

旋转通常的做法是以图像的中心为圆心旋转。MATLAB提供了imrotate函数用于实现图像的旋转。

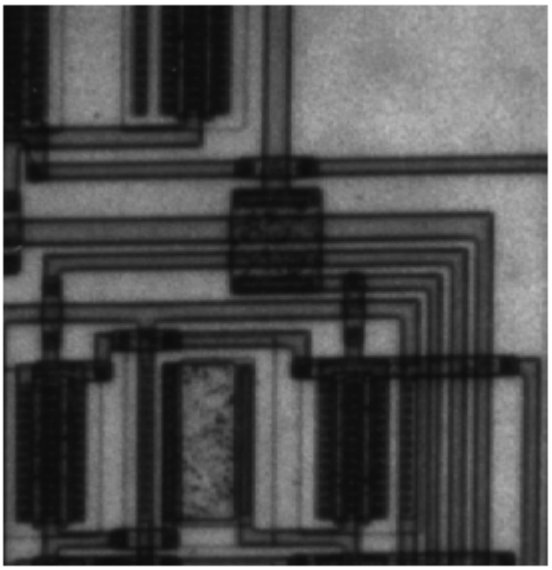
```
I=imread('cameraman.tif');
%  双线性插值法旋转图像，并裁剪图像，使其和原图像大小一致
B=imrotate(I,60,'bilinear','crop');
subplot(121),imshow(I),title('原图');
subplot(122),imshow(B),title('旋转图像60^{o}，并剪切图像');
```

原图　　旋转图像60°，并剪切图像

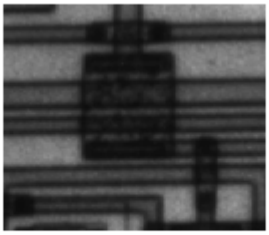

图像的剪切

对于要处理的图像，用户可能只关心图像的一部分内容，而不是整个图像。如果对整个图像进行处理，不仅要花费大量的时间，而且图像的其他部分可能会影响处理的效果，所以这时就要剪切出所要关心的部分图像，这样可以大大地提高处理的效率。MATLAB提供了imcrop函数用来实现图像的剪切。

```
I = imread('circuit.tif');
I2 = imcrop(I,[75 68 130 112]);          %  [75 68 130 112]为剪切区域
figure
imshow(I)
```



```
figure
imshow(I2)
```



```
```

对比度增强

对比度增强是增强技术中的一种比较简单但又十分重要的方法。这种方法是按一定的规则逐点修改输入图像每一像素的灰度，从而改变图像灰度的动态范围。MATLAB提供了imadjust函数用来对图像的强度进行调整。

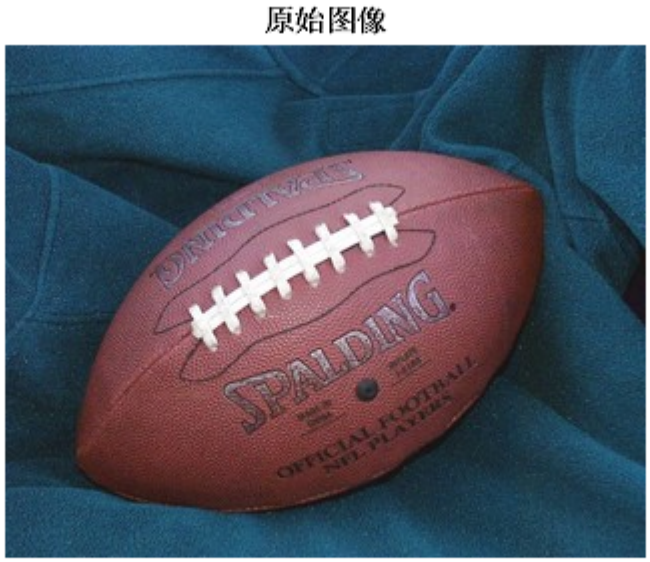
灰度图像

```
I = imread('pout.tif');
J = imadjust(I);
figure
imshow(I)
title('原始图像')
```


```
figure
imshow(J)
title('调整后图像')
```

彩色图像

```
RGB1 = imread('football.jpg');
RGB2 = imadjust(RGB1,[.2 .3 0; .6 .7 1],[]);
figure
imshow(RGB1)
title('原始图像')
```




```
figure
imshow(RGB2)
title('调整后图像')
```

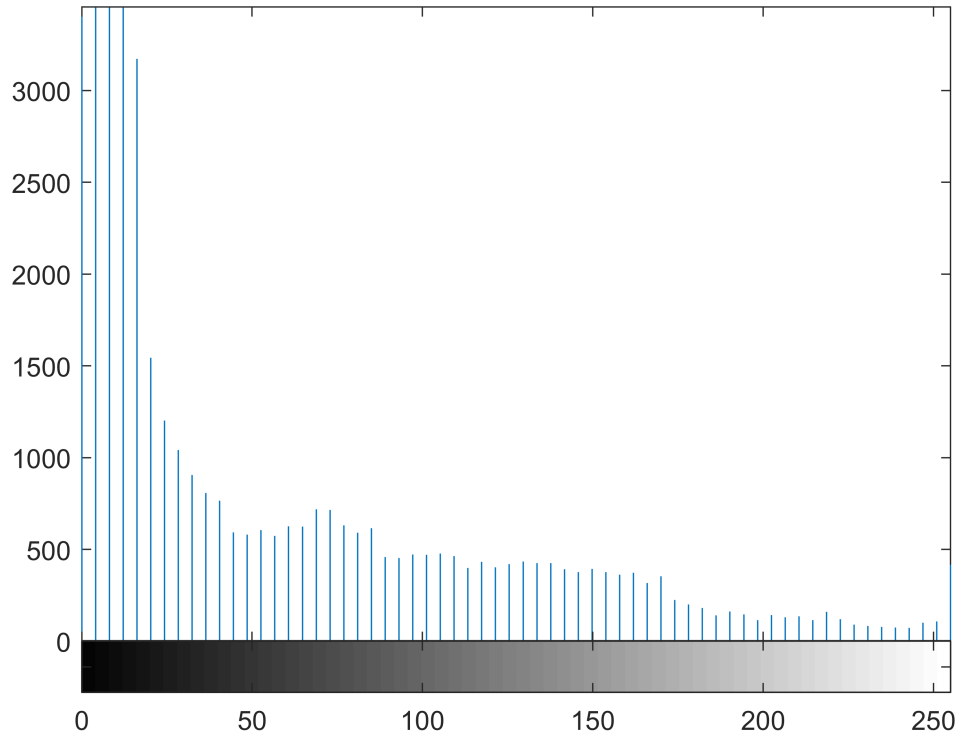
直方图均衡化

MATLAB提供了histeq函数通过直方图均衡化方法来增强对比度。

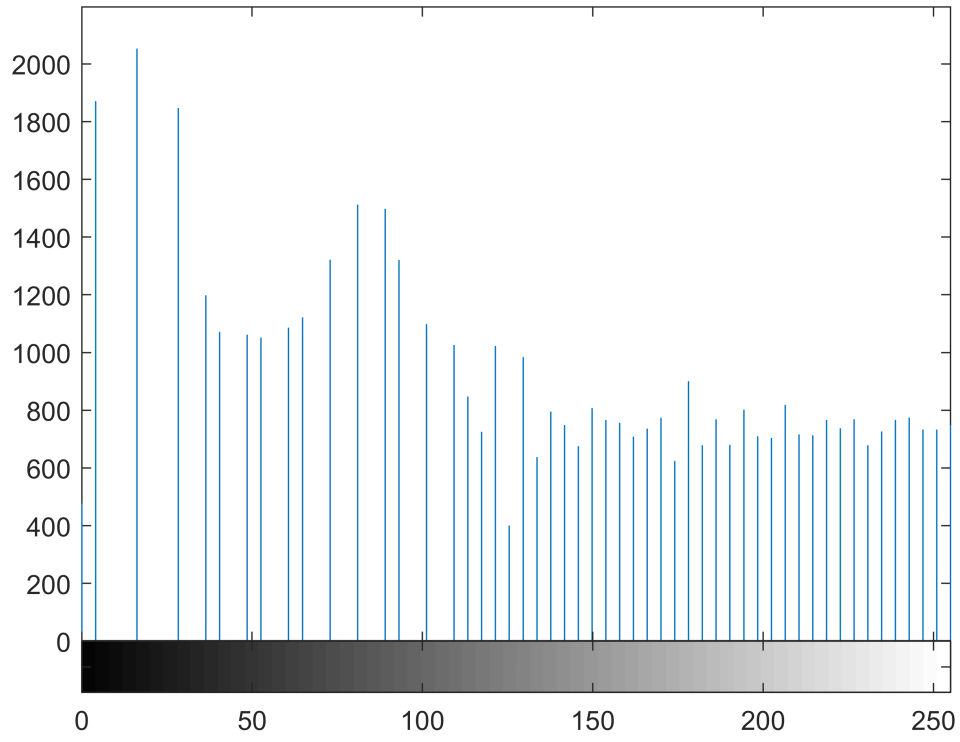
```
I = imread('tire.tif');          %  读入系统自带的测试图片
J = histeq(I);                  %  使用直方图均衡化方法增强对比度
imshow(I)                        %  原始图像
```

```
figure, imshow(J)                %  增强对比度之后的图像
```

```
figure; imhist(I,64)             %  原始直方图
```




```
figure; imhist(J,64)                % 增强对比度之后的直方图
```



空域滤波增强

使用空域模板进行的图像处理，被称为空域滤波。模板本身被称为空域滤波器。按空域滤波处理的效果分类，可以分为平滑滤波器和锐化滤波器。平滑的目的在于消除混杂图像干扰，改善图像质量，强化图像表现特征。锐化的目的在于增强图像边缘，以便对图像进行识别和处理。

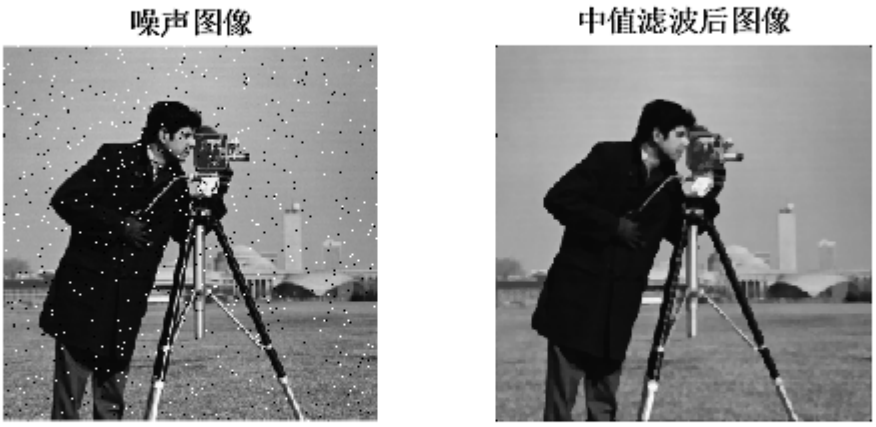
邻域平均法滤波。对图像中的每个像素进行3×3模板均值滤波，可以有效地平滑噪声。

```
I=imread('cameraman.tif');          %  cameraman.tif为系统自带的测试图片
figure
subplot(131),imshow(I);title('原图')
J=imnoise(I,'salt & pepper',0.01); %  添加椒盐噪声
subplot(132),imshow(J);title('噪声图像')
%  应用3×3邻域窗口法，fspecial函数用来实现一个均值滤波器
K1=filter2(fspecial('average',3),J,'full')/255;
subplot(133),imshow(K1);title('3×3窗的邻域平均滤波图像')
```

中值滤波

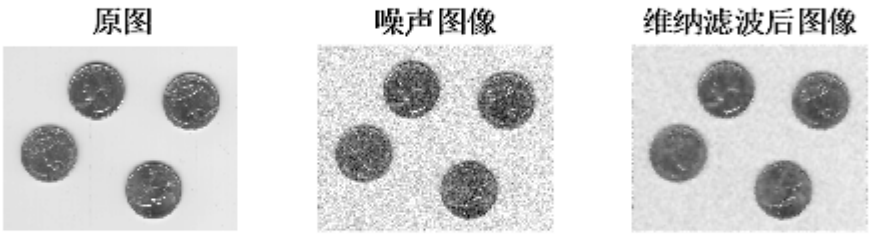
```
I=imread('cameraman.tif');
J=imnoise(I,'salt & pepper',0.02);            %  添加椒盐噪声
figure
subplot(121),imshow(J);title('噪声图像')
K=medfilt2(J);                                %  使用3×3的邻域窗的中值滤波
```

```
subplot(122),imshow(K);title('中值滤波后图像')
```



MATLAB还提供了函数wiener2用于实现维纳滤波

```
I=imread('eight.tif');
figure
subplot(131),imshow(I);title('原图')
J=imnoise(I,'gauss',0,0.01);                    %   添加高斯噪声
subplot(132),imshow(J);title('噪声图像')
[K1,noise]=wiener2(J,[5 5]);                    %   在5×5邻域内对图像进行维纳滤波
subplot(133),imshow(K1);title('维纳滤波后图像')
```

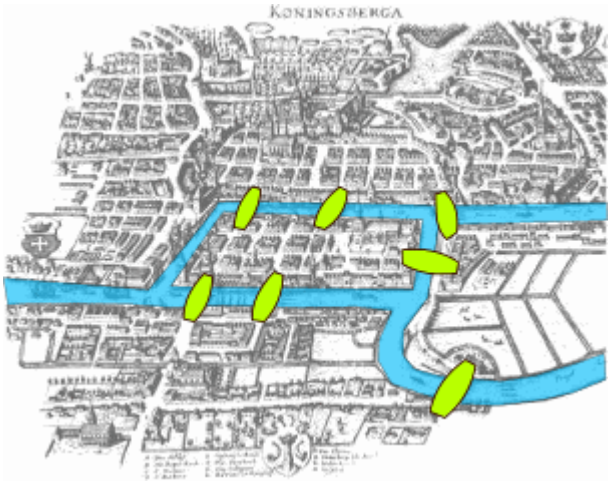


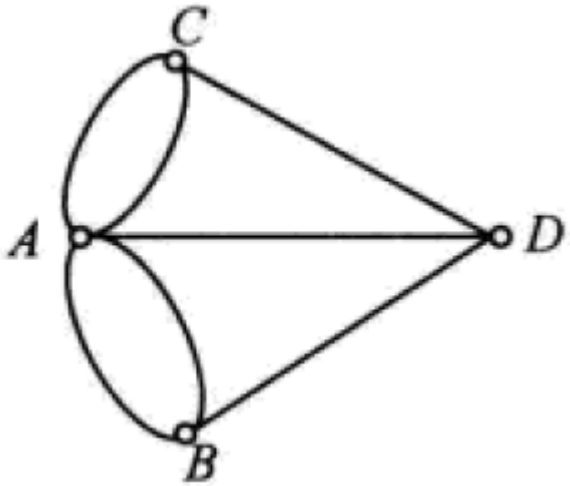
图论

图论（Graph theory）是数学的一个分支，它以图为研究对象，研究顶点和边组成的图形的数学理论和方法。图是区域在头脑和纸面上的反映，图论就是研究区域关系的学科。区域是一个平面，平面当然是二维的，但是，图在特殊的构造中，可以形成多维（例如大于3维空间）空间，这样的图已经超越了一般意义上的区域（例如一个有许多洞的曲面，它是多维的，曲面染色已经超出了平面概念）。

图论中的图是由若干给定的顶点及连接两顶点的边所构成的图形，这种图形通常用来描述某些事物之间的某种特定关系，用顶点代表事物，用连接两顶点的边表示相应两个事物间具有这种关系。

图论起源于著名的柯尼斯堡七桥问题。一般认为，于1736年出版的欧拉的关于柯尼斯堡七桥问题的论文是图论领域的第一篇文章。





图对象的创建

matlab提供了graph函数来创建无向图

```
figure
G = graph([1 1],[2 3])  % 创建
```

```
G =
  graph with properties:

    Edges: [2×1 table]
    Nodes: [3×0 table]
```

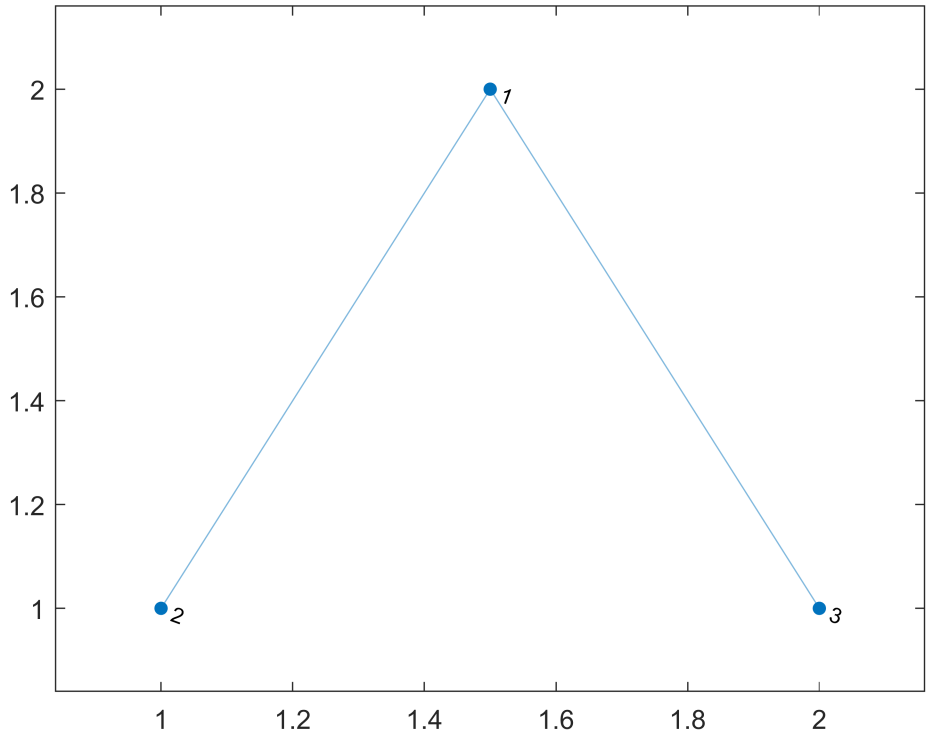
```
G.Edges             % 边
```

```
ans = 2×1 table
```

	EndNodes	
1	1	2

	EndNodes	
2	1	3

```
plot(G)
```

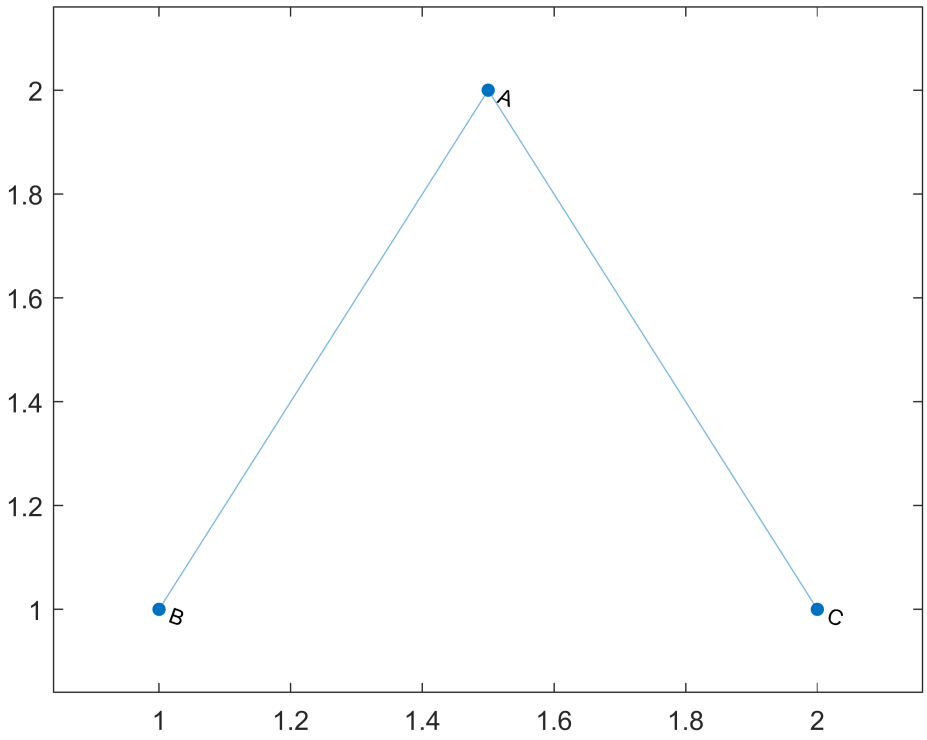



```
G.Nodes.Name = {'A' 'B' 'C'}';  % 节点名称
G.Edges
```

ans = *2×1 table*

	EndNodes	
1	'A'	'B'
2	'A'	'C'

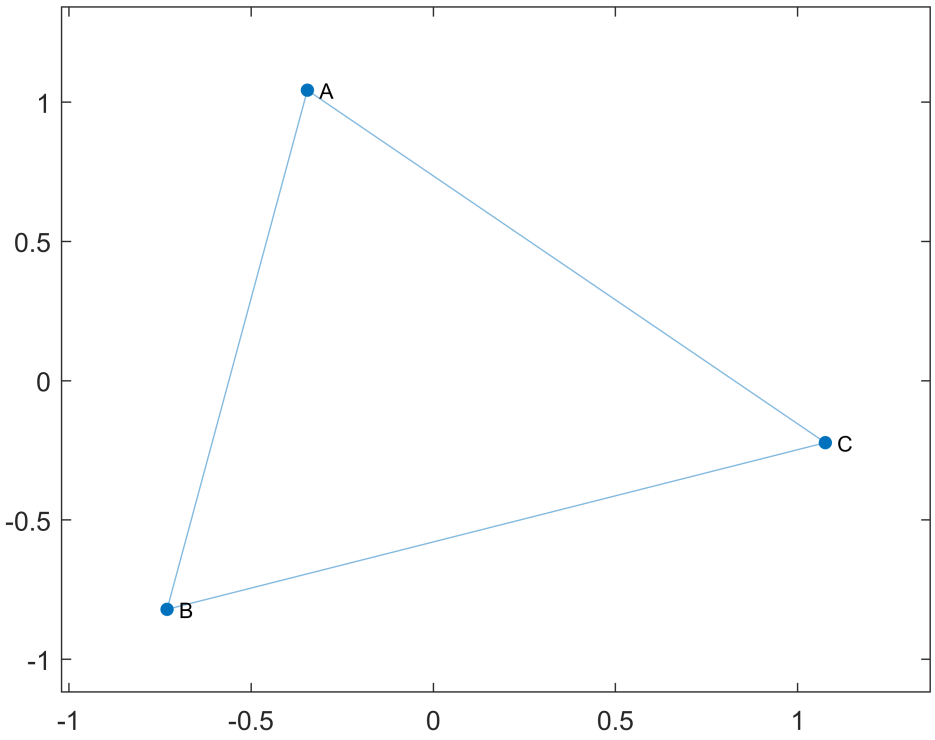
```
plot(G)
```




```
G = addedge(G,2,3)
```

```
G =
  graph with properties:

    Edges: [3×1 table]
    Nodes: [3×1 table]
```

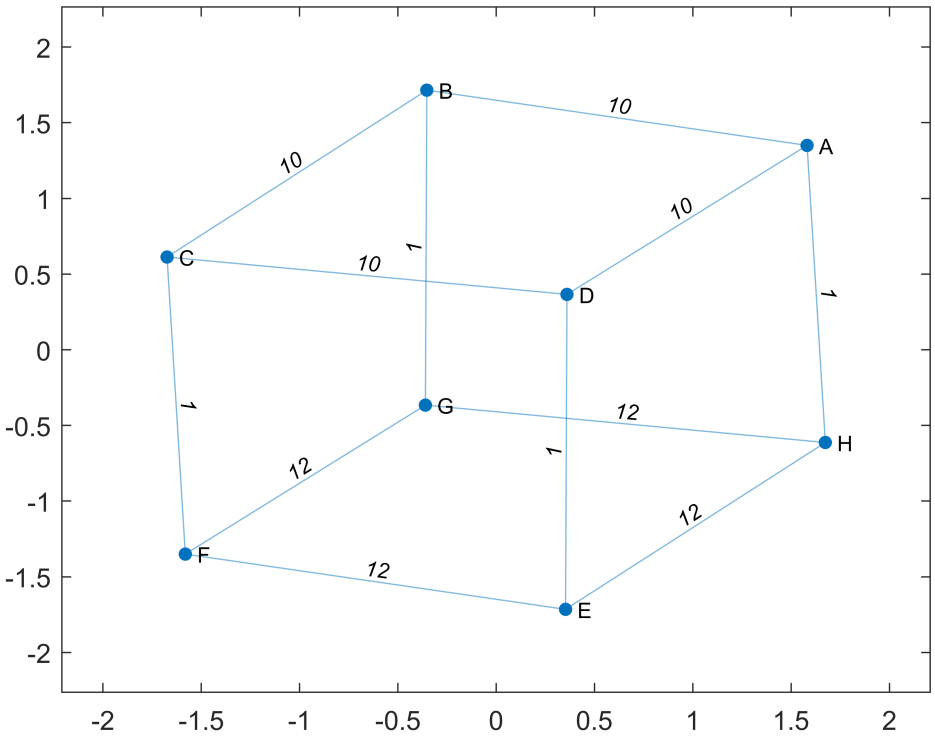
```
plot(G)
```




```
s = [1 1 1 2 2 3 3 4 5 5 6 7];
t = [2 4 8 3 7 4 6 5 6 8 7 8];
weights = [10 10 1 10 1 10 1 1 12 12 12 12];
names = {'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H'};
G = graph(s,t,weights,names)
```

```
G =
  graph with properties:

    Edges: [12×2 table]
    Nodes: [8×1 table]
```

```
plot(G,'EdgeLabel',G.Edges.Weight)
```



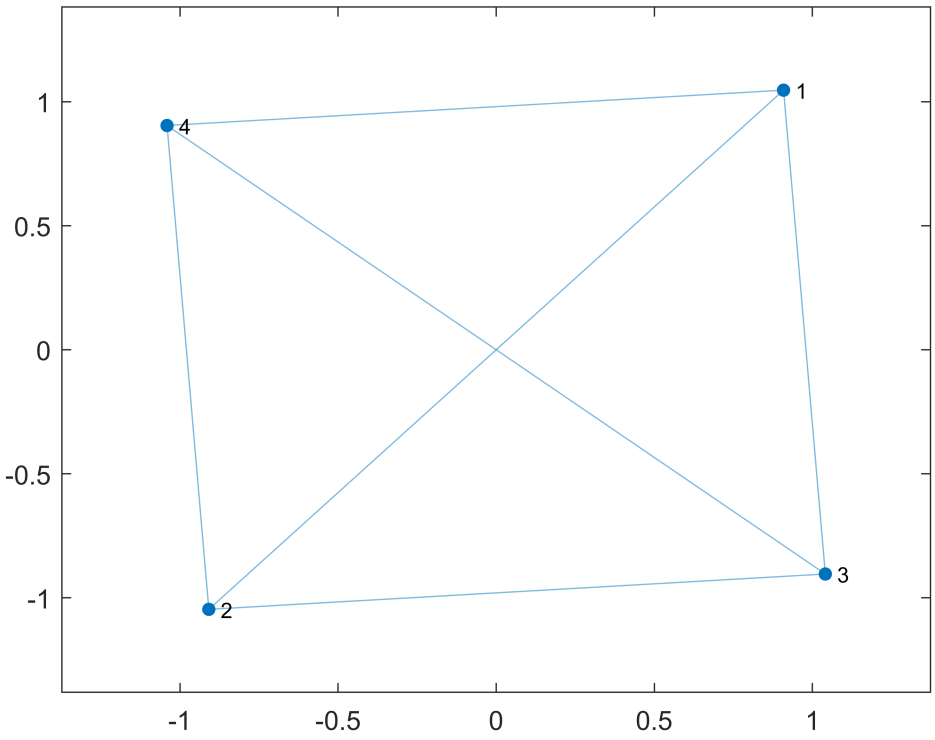

```
% 使用邻接矩阵来创建
A = ones(4) - diag([1 1 1 1])
```

```
A =
     0     1     1     1
     1     0     1     1
     1     1     0     1
     1     1     1     0
```

```
G = graph(A~=0)
```

```
G =
  graph with properties:

    Edges: [6×1 table]
    Nodes: [4×0 table]
```

```
figure
plot(G)
```




```
A = triu(magic(4))
```

```
A =
    16     2     3    13
     0    11    10     8
     0     0     6    12
     0     0     0     1
```

```
names = {'alpha' 'beta' 'gamma' 'delta'};
G = graph(A,names,'upper','OmitSelfLoops')
```

```
G =
  graph with properties:

    Edges: [6×2 table]
    Nodes: [4×1 table]
```

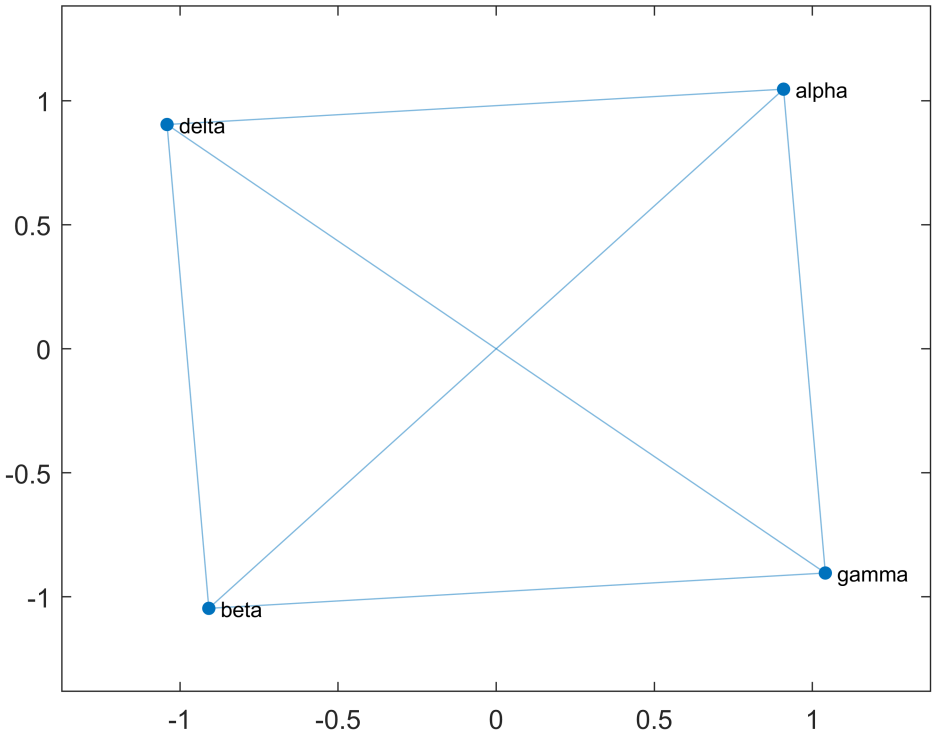
```
G.Edges
```

ans = *6×2 table*

	EndNodes		Weight
1	'alpha'	'beta'	2
2	'alpha'	'gamma'	3
3	'alpha'	'delta'	13

	EndNodes		Weight
4	'beta'	'gamma'	10
5	'beta'	'delta'	8
6	'gamma'	'delta'	12

```
figure
plot(G)
```




```
G.Nodes
```

ans = *4×1 table*

	Name
1	'alpha'
2	'beta'
3	'gamma'
4	'delta'

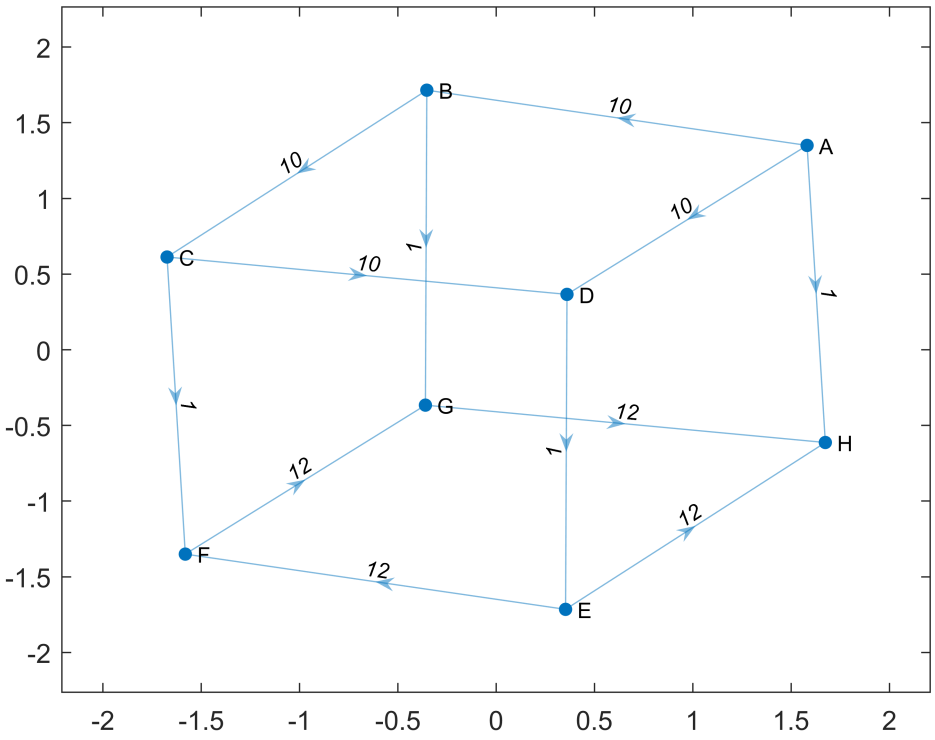
matlab提供了digraph函数来创建有向图

```
s = [1 1 1 2 2 3 3 4 5 5 6 7];
t = [2 4 8 3 7 4 6 5 6 8 7 8];
weights = [10 10 1 10 1 10 1 1 12 12 12 12];
```

```
names = {'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H'};
G = digraph(s,t,weights,names)
```

```
G =
  digraph with properties:

    Edges: [12×2 table]
    Nodes: [8×1 table]
```

```
plot(G,'Layout','force','EdgeLabel',G.Edges.Weight)
```




```
A = magic(4);
A(A>10) = 0
```

```
A =
     0     2     3     0
     5     0    10     8
     9     7     6     0
     4     0     0     1
```

```
names = {'alpha' 'beta' 'gamma' 'delta'};
G = digraph(A,names,'OmitSelfLoops')
```

```
G =
  digraph with properties:

    Edges: [8×2 table]
```

```
  Nodes: [4×1 table]
```

```
G.Edges
```

ans = *8×2 table*

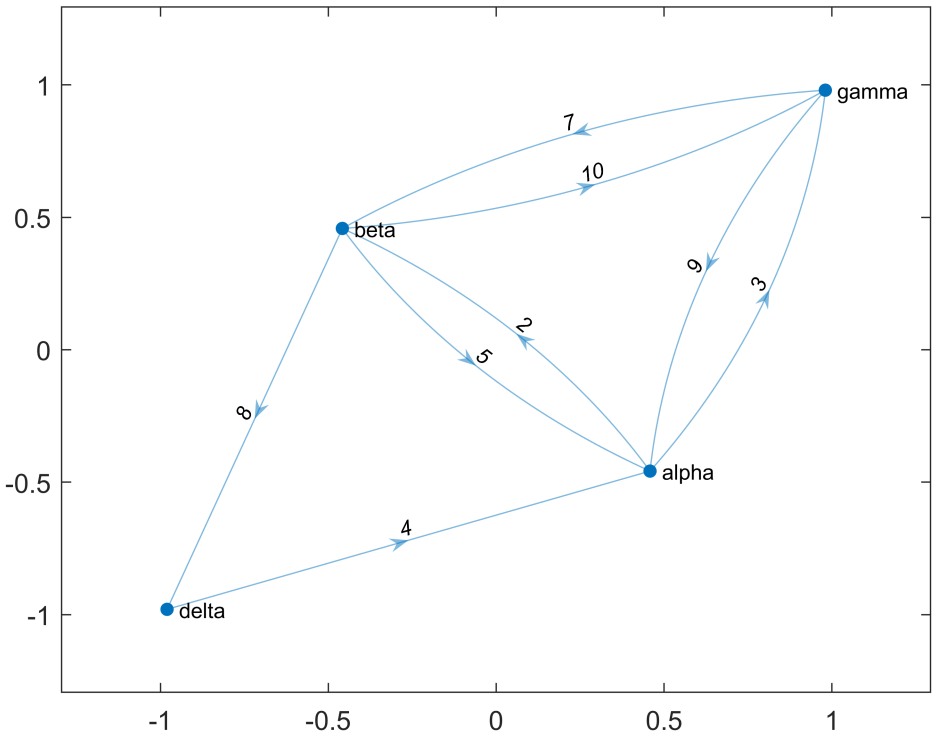
	EndNodes		Weight
1	'alpha'	'beta'	2
2	'alpha'	'gamma'	3
3	'beta'	'alpha'	5
4	'beta'	'gamma'	10
5	'beta'	'delta'	8
6	'gamma'	'alpha'	9
7	'gamma'	'beta'	7
8	'delta'	'alpha'	4

```
G.Nodes
```

ans = *4×1 table*

	Name
1	'alpha'
2	'beta'
3	'gamma'
4	'delta'

```
plot(G,'EdgeLabel',G.Edges.Weight)
```



邻接矩阵

```
s = [1 1 1 2 2 3];
t = [2 3 4 5 6 7];
G = digraph(s,t)
```

```
G =
  digraph with properties:

    Edges: [6×1 table]
    Nodes: [7×0 table]
```

```
A = adjacency(G)
```

```
A =
   (1,2)        1
   (1,3)        1
   (1,4)        1
   (2,5)        1
   (2,6)        1
   (3,7)        1
```

```
full(A)
```

```
ans =
     0     1     1     1     0     0     0
     0     0     0     0     1     1     0
```

```
     0     0     0     0     0     0     1
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

关联矩阵

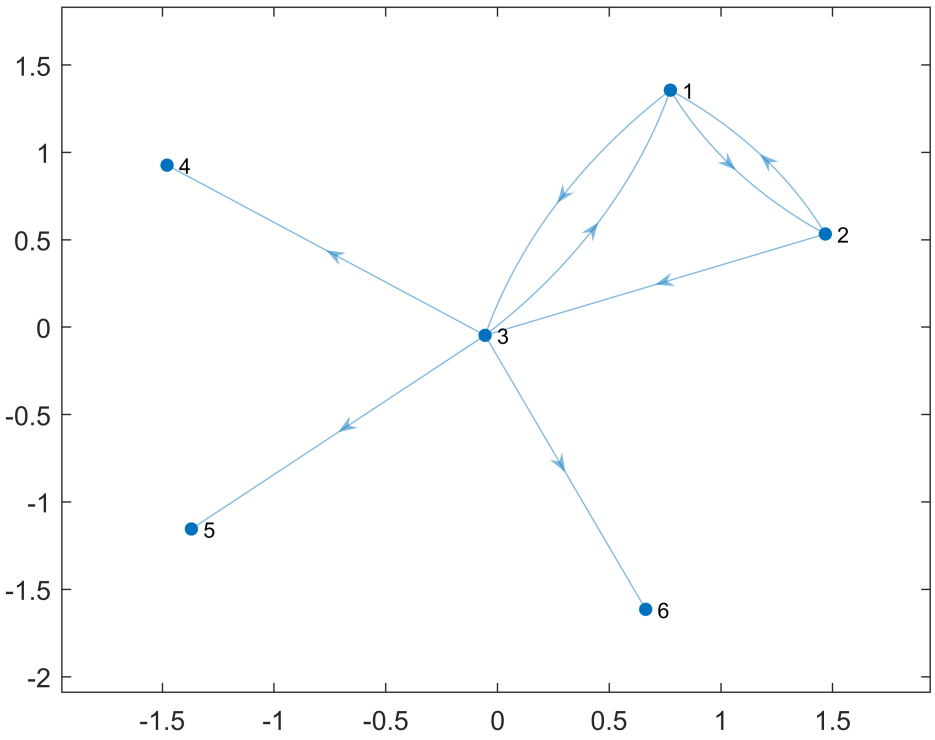
```
s = [1 2 1 3 2 3 3 3];
t = [2 1 3 1 3 4 5 6];
G = digraph(s,t);
I = incidence(G)
```

```
I =
   (1,1)       -1
   (2,1)        1
   (1,2)       -1
   (3,2)        1
   (1,3)        1
   (2,3)       -1
   (2,4)       -1
   (3,4)        1
   (1,5)        1
   (3,5)       -1
   (3,6)       -1
   (4,6)        1
   (3,7)       -1
   (5,7)        1
   (3,8)       -1
   (6,8)        1
```

```
full(I)
```

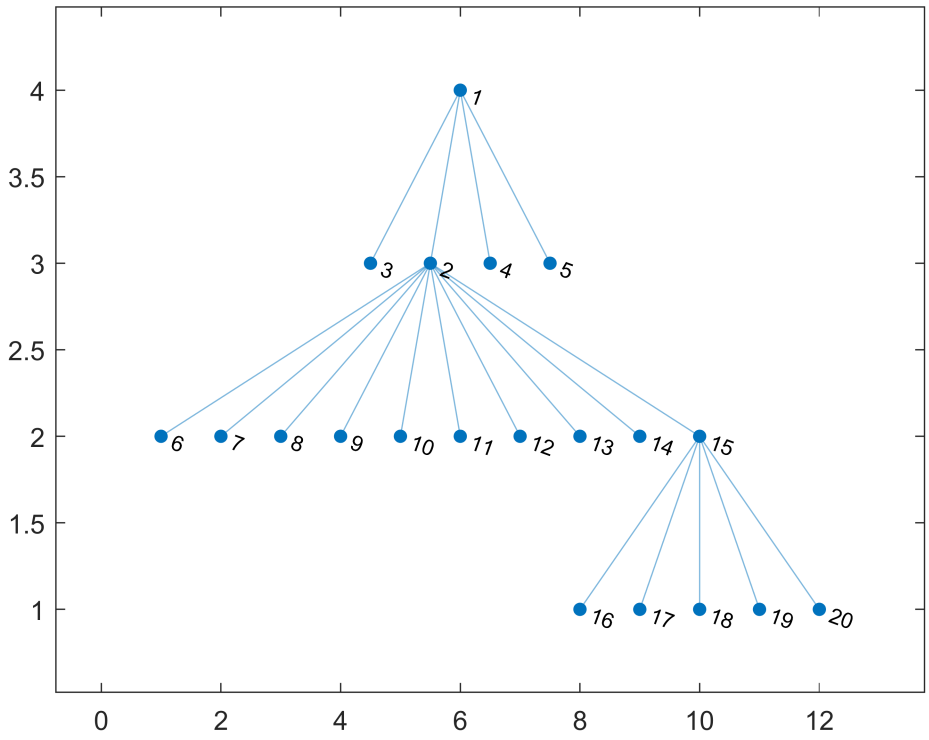
```
ans =
    -1    -1     1     0     1     0     0     0
     1     0    -1    -1     0     0     0     0
     0     1     0     1    -1    -1    -1    -1
     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     1
```

```
plot(G)
```



广度优先搜索

```
s = [1 1 1 1 2 2 2 2 2 2 2 2 2 15 15 15 15 15];
t = [3 5 4 2 14 6 11 12 13 10 7 9 8 15 16 17 19 18 20];
G = graph(s,t);
plot(G)
```

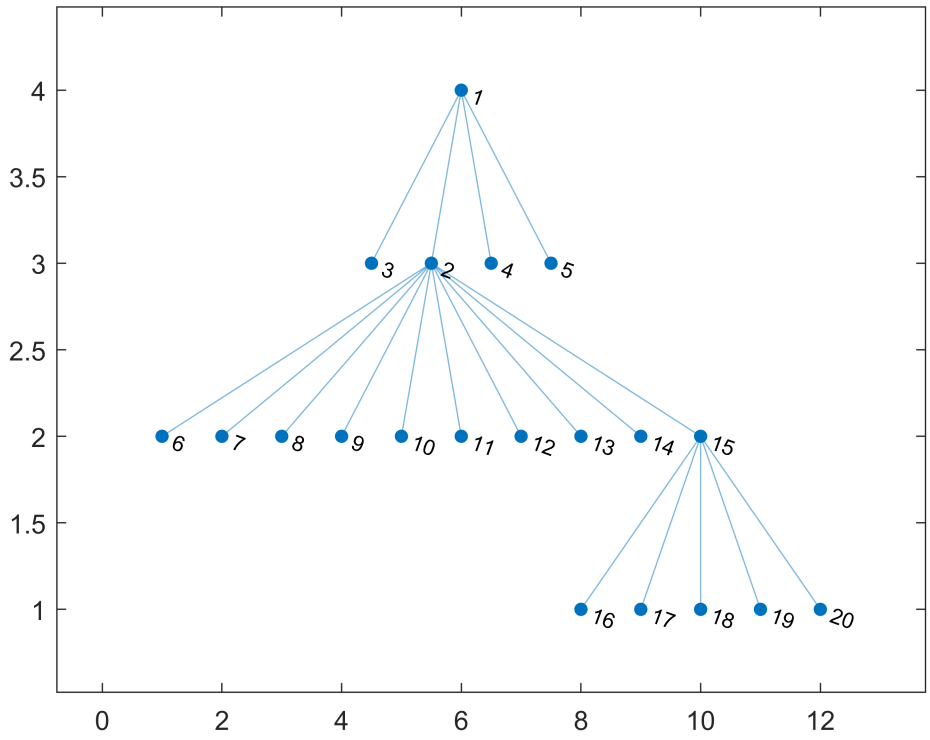



```
v = bfsearch(G,2)
```

```
v =
     2
     1
     6
     7
     8
     9
    10
    11
    12
    13
     ⋮
```

深度优先搜索

```
s = [1 1 1 1 2 2 2 2 2 2 2 2 2 15 15 15 15 15];
t = [3 5 4 2 14 6 11 12 13 10 7 9 8 15 16 17 19 18 20];
G = graph(s,t);
plot(G)
```

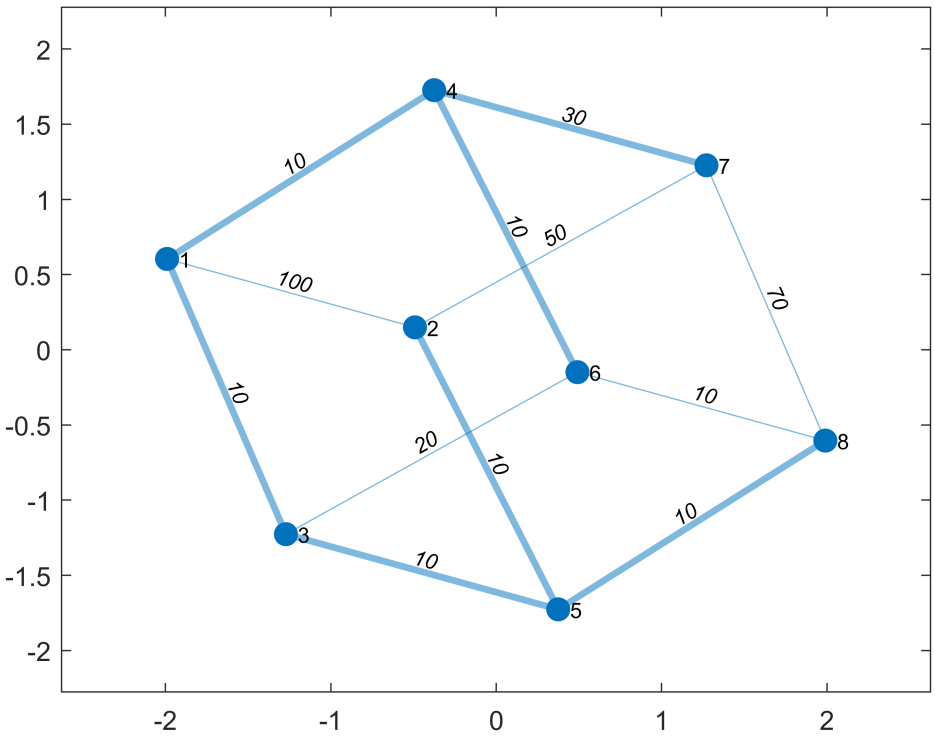



```
v = dfsearch(G,17)
```

```
v =
    17
    15
     2
     1
     3
     4
     5
     6
     7
     8
     ⋮
```

最小生成树

```
s = [1 1 1 2 5 3 6 4 7 8 8 8];
t = [2 3 4 5 3 6 4 7 2 6 7 5];
weights = [100 10 10 10 10 20 10 30 50 10 70 10];
G = graph(s,t,weights);
p = plot(G,'EdgeLabel',G.Edges.Weight);
[T,pred] = minspantree(G);
highlight(p,T)
```



最短路问题

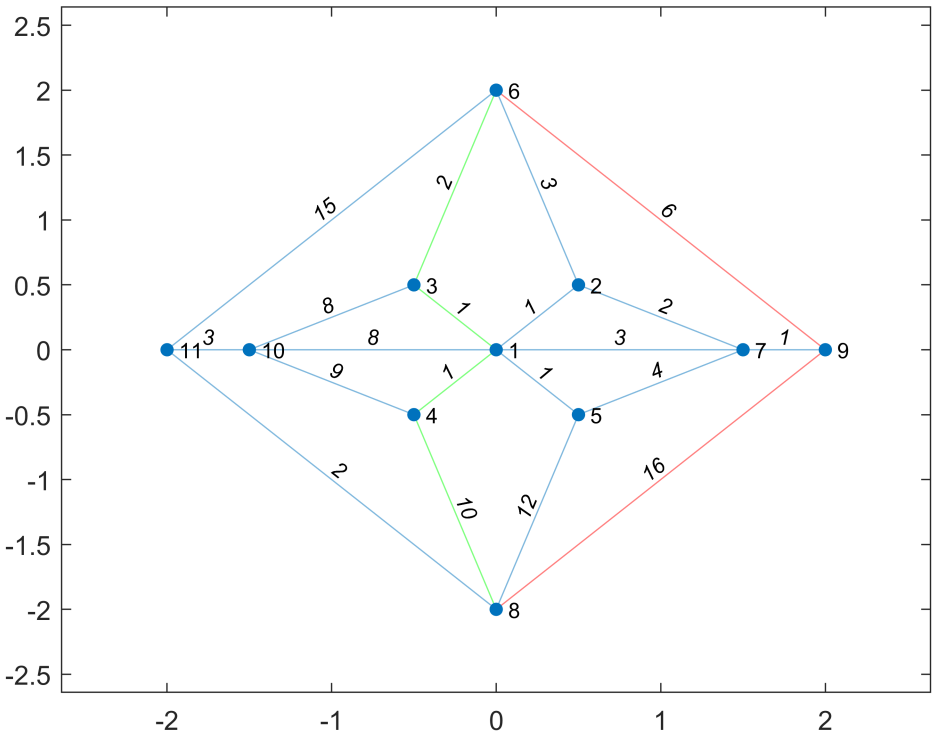
```
s = [1 1 1 1 1 2 2 7 7 9 3 3 1 4 10 8 4 5 6 8];
t = [2 3 4 5 7 6 7 5 9 6 6 10 10 10 11 11 8 8 11 9];
weights = [1 1 1 1 3 3 2 4 1 6 2 8 8 9 3 2 10 12 15 16];
G = graph(s,t,weights);

x = [0 0.5 -0.5 -0.5 0.5 0 1.5 0 2 -1.5 -2];
y = [0 0.5 0.5 -0.5 -0.5 2 0 -2 0 0 0];
p = plot(G,'XData',x,'YData',y,'EdgeLabel',G.Edges.Weight);

[path1,d] = shortestpath(G,6,8)
```

```
path1 =
     6     3     1     4     8
d = 14
```

```
highlight(p,path1,'EdgeColor','g')
[path2,d] = shortestpath(G,6,8,'Method','unweighted')
```

```
path2 =
     6     9     8
d = 2
```

```
highlight(p,path2,'EdgeColor','r')
```



金融--现金流的计算

内部收益率

```
Stream = [-20000,  2000,  2500,  3500, -5000,  6500,  9500,  9500,  9500];
ROR = irr(Stream)
```

```
ROR = 0.1172
```

$$\frac{cf_1}{(1+r)} + \frac{cf_2}{(1+r)^2} + \ldots + \frac{cf_n}{(1+r)^n} + Investment = 0,$$

Year (n)	Cash flow (C_n)
0	-123400
1	36200
2	54800
3	48100

$$\mathrm{NPV} = -123400 + \frac{36200}{(1+r)^1} + \frac{54800}{(1+r)^2} + \frac{48100}{(1+r)^3} = 0$$

房贷还款

```
Payment = payper(Rate,NumPeriods,PresentValue,FutureValue,Due)
```

```
Payment = payper(0.056/12, 300, 1000000, 0, 0)
```

```
Payment = 6.2007e+03
```

现值计算

```
PresentVal = pvfix(Rate,NumPeriods,Payment,ExtraPayment,Due)
```
固定

```
PresentVal = pvfix(0.06/12, 5*12, 200, 0, 0)
```

```
PresentVal = 1.0345e+04
```

```
PresentVal = pvvar(CashFlow,Rate)
```
非固定

```
PresentVal = pvvar([-10000 2000 1500 3000 3800 5000], 0.08)
```

```
PresentVal = 1.7154e+03
```

终值计算

```
FutureVal = fvfix(Rate,NumPeriods,Payment,PresentVal,Due)
```
固定

```
FutureVal = fvfix(0.09/12, 12*10, 200, 1500, 0)
```

```
FutureVal = 4.2380e+04
```

```
FutureVal = fvvar(CashFlow,Rate)
```
非固定

```
FutureVal = fvvar([-10000 2000 1500 3000 3800 5000], 0.08)
```

```
FutureVal = 2.5205e+03
```