

Matlab编程与应用

第四讲

中国科技大学信息学院

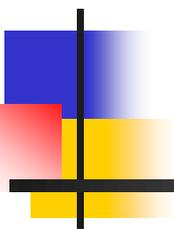
陆伟

luwei@ustc.edu.cn



本讲内容

- part1: 图像句柄
- part2: 字符串
- part3: 单元数组与结构体
- part4: 稀疏矩阵



Part1:

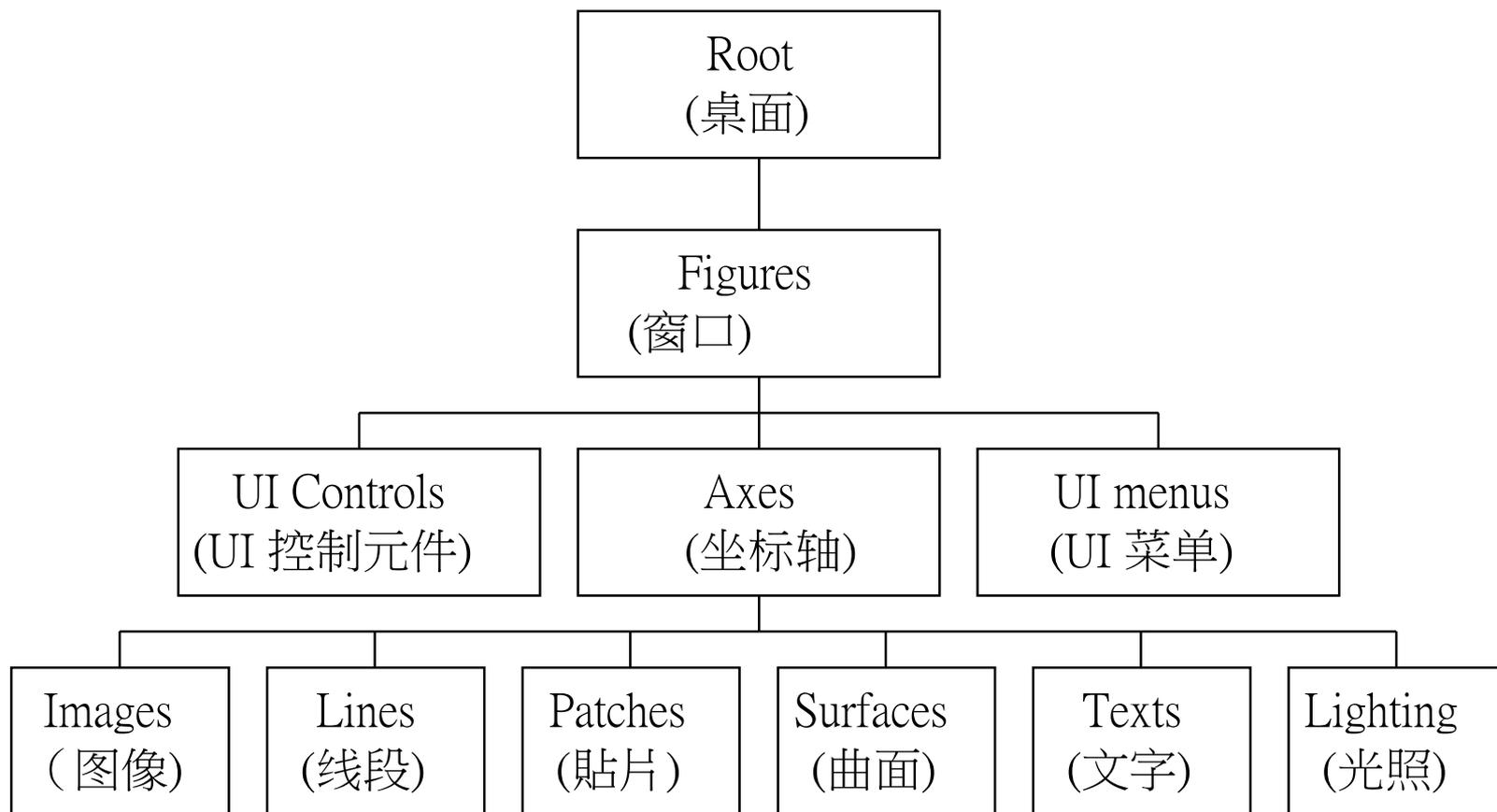
图像句柄



图形对象

- 每个构成图形的基本单位都被视为一个对象(Object), 例如:
 - 曲线、曲面、坐标轴、文字...
- 每个对象都分配有一个独一无二的句柄(handle), 就像每个人都有独一无二的身份证号码
- 根据对象的句柄, 就可以修改图形对象的所有属性

图形对象的层次结构



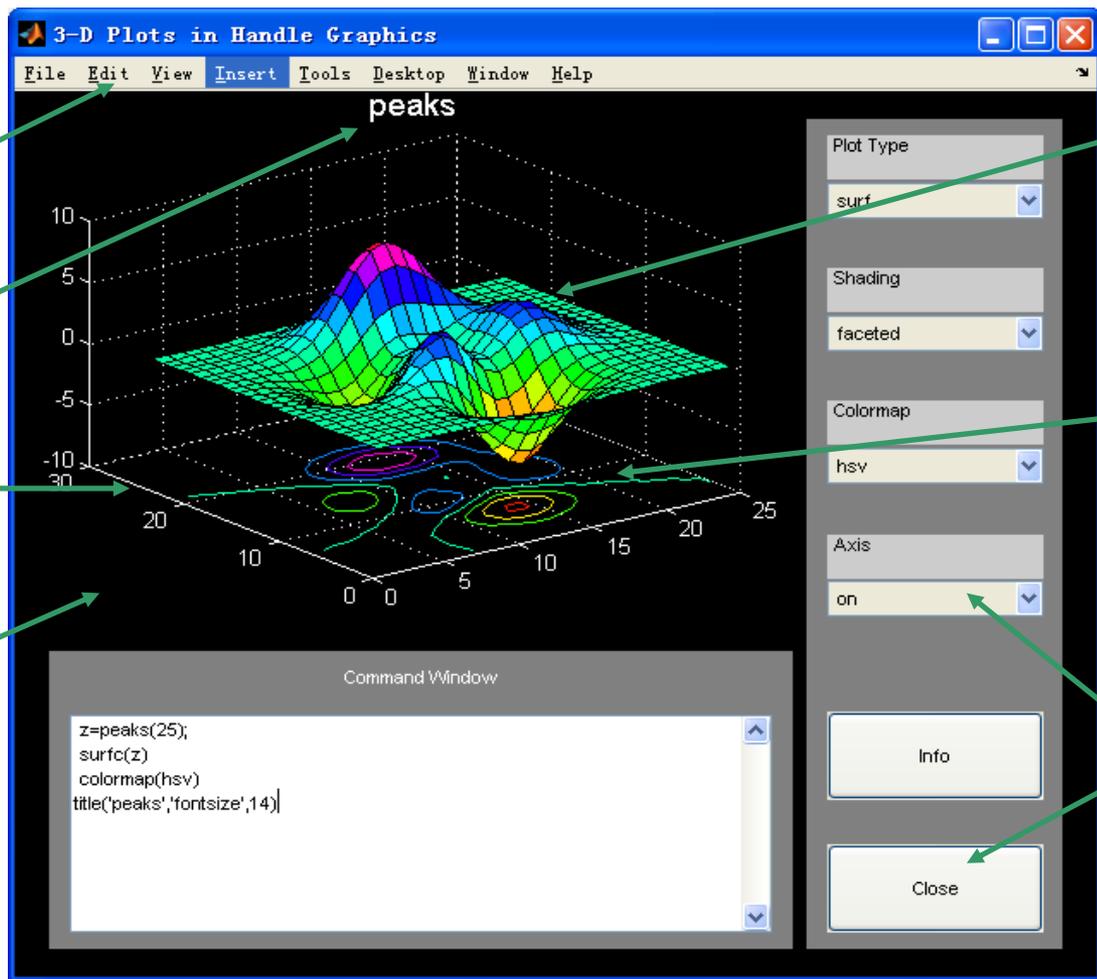
图形对象的层次结构

UI菜单对象
uimenu

文字对象
text

坐标轴对象
axes

窗口对象
figure



曲面对象
surface

线段对象
line

uicontrol对象



图形对象的属性编辑器

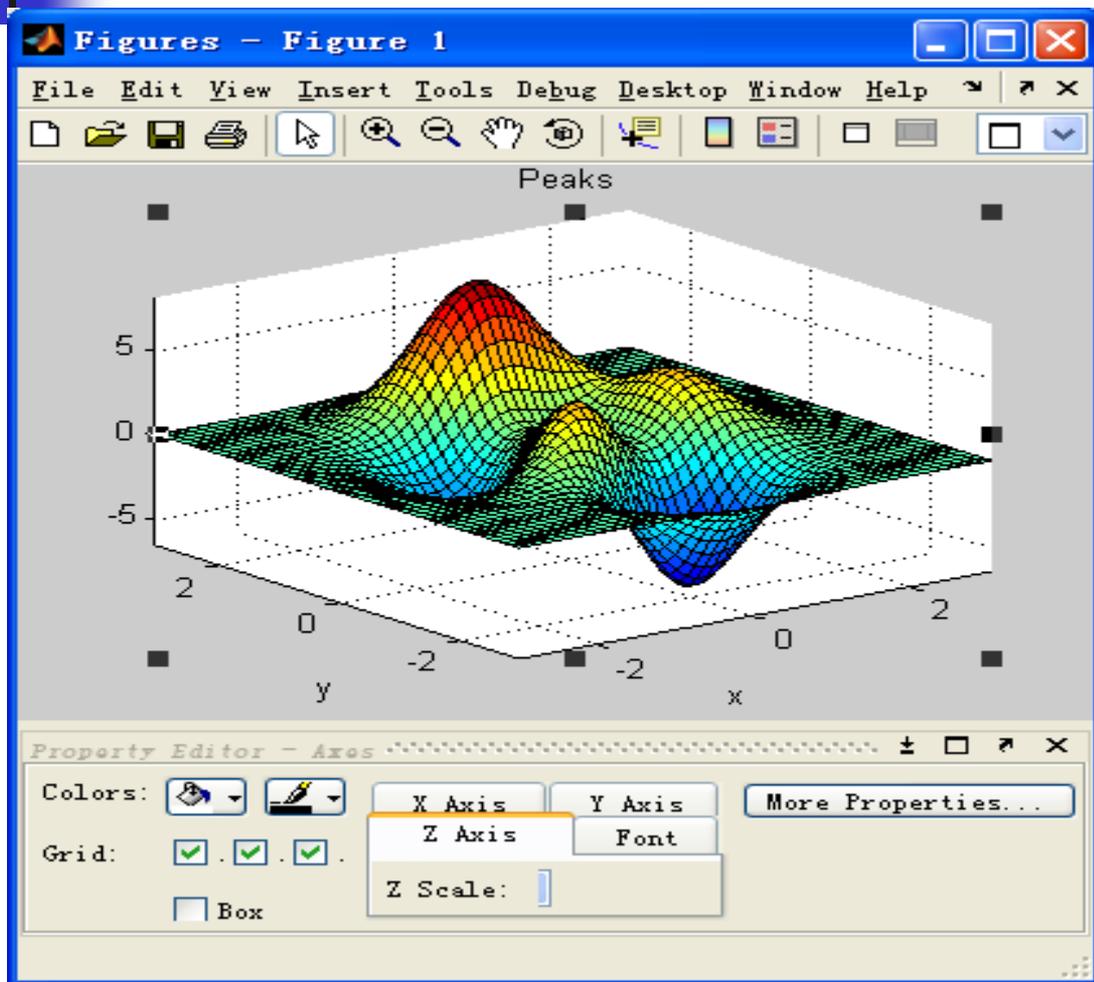
- 先画图，再利用**propedit**开启属性编辑器

```
>>peaks; %画出peaks 3D图
```

```
>>propedit; %开启属性编辑器
```

- 也可以点选工具行上面的图示  开启图形编辑器

图形对象编辑器



- 鼠标左键点击希望编辑的对象（如 figure、axes、text、进行修改。



图形对象的属性编辑

- 常常需要在命令行或m文件中对图形对象的属性进行修改

set : 设定某图形对象的某个属性值

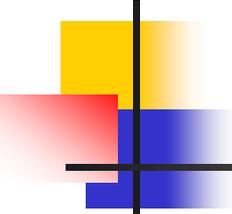
get: 获得某个图形对象的现有属性值

findobj: 在句柄图形的层次结构中，找到想要进行编辑的图形对象。

gcf :

gca :

gco :



set

```
t = 0 : 0.1 : 4*pi;  
y = exp(-t/4) .* sin(t);  
h = plot(t,y);  
set(h,'linewidth', 3);  
set(h,'Marker','o');  
set(h,'markersize',20);
```



get

```
>>get(h,'linewidth');
```

```
>>get(h,'color');
```

```
>>get(h)
```



findobj

```
>>plot(rand(10,2));
```

```
>>h = findobj(0,'type','line');
```

```
>>set(h,'linewidth',3);
```



axes与axis的区别

- axes是创建坐标轴，axis是设定其范围。

```
clear all;
```

```
x=0:10*pi;
```

```
y=sin(x);
```

```
% 创建一个坐标系。起点是左边占到显示窗口的十分之一处，  
%下边占到十分之二处，宽占十分之三，高占十分之四。
```

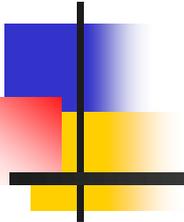
```
axes('position',[0.1 0.2 0.3 0.4]);
```

```
plot(x,y); %画图。
```

```
% 设置x的坐标范围是0到 $2\pi$ ，y的范围是-0.5到0.5。看横纵坐标
```

```
axis([0 2*pi -0.5 0.5]);
```

```
%axes;
```



Part2:

Matlab字符串



主要内容

- 字符串生成
- 与数值之间转化
- 查找
- 匹配
- 连接
- 比较



字符串

- 字符串是**ASCII**数值型数组，每个字符用两个字节表示，只是显示为字符形式。

- 应用：

给用户提示：‘please enter the r:’

显示结果： ‘The distance is 8 mm.’

图形注释：



字符串

- 字符串：用单引号括起来的一个或多个字符。

```
>> str1 = 'Hello world!'
```

```
str1 =
```

```
Hello world!
```

```
>> size(str1)
```

```
ans =
```

```
     1     12
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
str1	1x12	24	char	

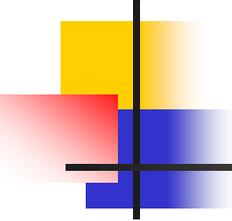


字符串

- 字符串显示为对应的ASCII码值:

```
>> double(str1)
ans =
  Columns 1 through 10
    72 101 108 108 111 32 119 111 114 108
  Columns 11 through 12
    100    33
```

```
>> double('abc 0123')
ans =
  97  98  99  32  32  48  49  50  51
```



ASCII码

- ASCII : **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

美国信息交换标准代码

- 基于拉丁字母的一套电脑编码系统，主要用于显示现代英语和其他西欧语，统一规定了上述常用符号用哪些二进制数来表示。
- 汉字编码 **GB2312**标准 双字节

ASCII 字符代码表 一

高四位 低四位		ASCII非打印控制字符										ASCII 打印字符											
		0000					0001					0010	0011		0100	0101		0110		0111			
		0					1					2	3		4	5		6		7			
		+进制	字符	ctrl	代码	字符解释	+进制	字符	ctrl	代码	字符解释	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符
0000	0	0	BLANK NULL	^@	NUL	空	16	▶	^P	DLE	数据链路转意	32		48	0	64	@	80	P	96	`	112	p
0001	1	1	☺	^A	SOH	头标开始	17	◀	^Q	DC1	设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q
0010	2	2	☹	^B	STX	正文开始	18	↕	^R	DC2	设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r
0011	3	3	♥	^C	ETX	正文结束	19	!!	^S	DC3	设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s
0100	4	4	♦	^D	EOT	传输结束	20	¶	^T	DC4	设备控制 4	36	\$	52	4	68	D	84	T	100	d	116	t
0101	5	5	♣	^E	ENQ	查询	21	♠	^U	NAK	反确认	37	%	53	5	69	E	85	U	101	e	117	u
0110	6	6	♠	^F	ACK	确认	22	■	^V	SYN	同步空闲	38	&	54	6	70	F	86	V	102	f	118	v
0111	7	7	●	^G	BEL	震铃	23	↑↓	^W	ETB	传输块结束	39	'	55	7	71	G	87	w	103	g	119	w
1000	8	8	◻	^H	BS	退格	24	↑	^X	CAN	取消	40	(56	8	72	H	88	X	104	h	120	x
1001	9	9	○	^I	TAB	水平制表符	25	↓	^Y	EM	媒体结束	41)	57	9	73	I	89	Y	105	i	121	y
1010	A	10	◻	^J	LF	换行/新行	26	→	^Z	SUB	替换	42	*	58	:	74	J	90	Z	106	j	122	z
1011	B	11	♂	^K	VT	竖直制表符	27	←	^[ESC	转意	43	+	59	;	75	K	91	[107	k	123	{
1100	C	12	♀	^L	FF	换页/新页	28	└	^\ FS	文件分隔符	44	,	60	<	76	L	92	\	108	l	124		
1101	D	13	🎵	^M	CR	回车	29	↔	^] GS	组分隔符	45	-	61	=	77	M	93]	109	m	125	}	
1110	E	14	🎵	^N	SO	移出	30	▲	^6 RS	记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~	
1111	F	15	☼	^O	SI	移入	31	▼	^- US	单元分隔符	47	/	63	?	79	O	95	_	111	o	127	Δ	*Back space

注：表中的ASCII字符可以用:ALT + “小键盘上的数字键”输入



字符串

- **ASCII**码中数值转换为字符串: **char**

```
>> x = 'a' %变量x是字符串，含有一个字符a
x =
a
>> y=double(x) %将字符a转换为ASCII码
y =
    97
>> z = char(y) %将ASCII码转换为字符
z =
a
```



字符串

■ ASCII码中数值转换为字符串: **char**

```
>> x =[];
>> for k = 1:10
        x = [x, 'a'+k-1];
    end
>> x
x =
    97    98    99   100   101   102   103   104   105   106
>> char(x)
ans =
abcdefg hij
```

字符串

- 双精度数转换成字符串: `num2str`

```
>> x=31.2;
```

```
>> y = num2str(x)
```

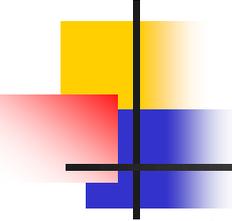
```
y = 31.2
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
x	1x1	8	double	
y	1x4	8	char	

```
x=39; a = num2str(x); b = char(x);
```

问: a是否等于b?

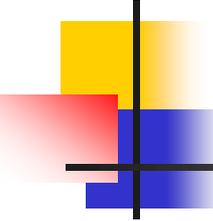


字符串

- 字符串转换成双精度数: `str2num`

```
>> str = '36' ;  
>> x=str2num(str) %就是36  
x = 36  
>> y = double(str) %变成对应的ASCII码  
y = 51      54  
>> whos
```

Name	Size	Bytes	Class
str	1x2	4	char
x	1x1	8	double
y	1x2	16	double



字符串

- 字符串显示: **disp** **sprintf**

```
disp('Now ,do the second step...')
```

%用于提示程序进程

```
for k = 1:10
```

```
... %循环主体
```

```
disp(['Loop ' int2str(k) ' is done'])
```

%提示程序进程

```
end
```



字符串

- `sprintf(formatSpec,A1,...,An)`

`str1 = sprintf('The value of pi is %.2f',pi)`

`str2 = sprintf('Some numbers:%5d,%2d',33,2^3)`

<code>%d</code> 整数
<code>%f</code> 浮点数
<code>%c</code> 单个字符
<code>%s</code> 字符串

<code>%5d</code> 整数
<code>%6.2f</code> 浮点数共6位，小数2位
<code>%.3f</code> 指定浮点数小数位数为3



字符串

- 字符串数组: char

```
>> kemu = char('math','english','DSP')
kemu =
math
english
DSP      %字符串长度不等, matlab会填充空格使其相等
>> kemu(1)
ans = m
>> kemu(1,:)
ans = math
>> a =kemu(1,:)
a = math
>> a = deblank(a) %清除空格
a =  math
```



字符串

- 字符串操作

strcat: 连接两个或多个字符串

strcmp: 比较两个字符串是否完全一致

strstr: 在一个字符串中查找特定子字符串

strrep: 用一个子字符串代替特定字符串



字符串

```
a = 'hello '
```

```
b = 'goodbye'
```

```
strcat(a, b)
```

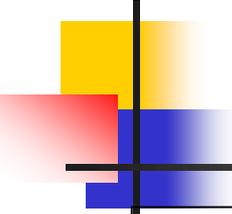
```
ans =
```

```
hellogoodbye
```

```
[a b]
```

```
ans =
```

```
hello goodbye
```



字符串

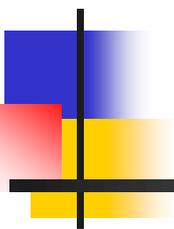
```
strcmp('Yes', 'No')  
ans = 0  
strcmp('Yes', 'Yes')  
ans = 1
```

```
S = 'Find the starting indices of the ...  
    pattern string';  
strfind(S, 'in')  
ans = 2 15 19 45  
strfind(S, 'In')  
ans = []
```



字符串

```
s1 = 'This is a good example.';  
str = strrep(s1, 'good', 'great')  
str =  
This is a great example.
```



Part3:

单元数组（cell array）与 结构体（structures）



单元数组（cell array）

- 将不同类型的、相关的数据集成在一个变量中。
- 该变量称为单元数组，其中的每个元素称为单元（**cell**）。
- **cell**可以是任何数据类型：字符串、双精度数组、其他单元数组。不同的**cell**可以包含不同的数据类型。



单元数组 (cell array)

3-by-3 Cell Array

cell 1,1 <table border="1"><tbody><tr><td>1</td><td>4</td><td>3</td></tr><tr><td>0</td><td>5</td><td>8</td></tr><tr><td>7</td><td>2</td><td>9</td></tr></tbody></table>	1	4	3	0	5	8	7	2	9	cell 1,2 'Anne Smith'	cell 1,3 []
1	4	3									
0	5	8									
7	2	9									
cell 2,1 $3+7i$	cell 2,2 [-3.14...3.14]	cell 2,3 []									
cell 3,1 []	cell 3,2 []	cell 3,3 5									



单元数组（cell array）

- 用途：使得大量的相关数据的处理与引用变得简单而方便。

比如：一段处理的语音信号，除了语音数据外，还希望记录相关信息，如说话人；性别；采样率；录制时间；
- GUI、Simulink程序中数据的传递



创建单元数组

- 创建一个2*2的单元数组:

方法一：大括号在右边：

```
A(1,1) = { [1 2 3 ; 4 5 6] };
```

```
A(1,2) = {3+4i};
```

```
A(2,1) = { 'hello world!'};
```

```
A(2,2) = {1:10};
```



创建单元数组

方法二：大括号在左边：

```
A{1,1} = [1 2 3 ; 4 5 6];
```

```
A{1,2} = 3+4i;
```

```
A{2,1} = 'hello world!';
```

```
A{2,2} = 1:10 ;
```



创建单元数组

方法三：直接赋值

```
B = { [1 2], '张三', 2+3i, 5 };
```

```
C = { [1:10], 'USTC', 4-2j, 2 };
```



创建单元数组

- 方法四：首先生成一个空单元数组，再添加数据。

```
C = cell(2,3)
```

```
C(1,1) = 'this is wrong' ;
```

```
C(1,1) = {'this is right'} ;
```

```
C{2,2} = 'this works too';
```



单元数组的内容显示

>> A

A =

[2x3 double]	[3.0000 + 4.0000i]
'hello,world!'	[1x10 double]



单元数组的内容显示(celldisp)

```
>> celldisp(A)
```

```
A{1,1} =
```

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

```
A{2,1} =
```

```
hello,world!
```

```
A{1,2} =
```

```
3.0000 + 4.0000i
```

```
A{2,2} =
```

```
    1    2    3    4    5    6    7    8    9   10
```



单元数组的内容显示

```
>> A{:}
```

```
ans =
```

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

```
ans =
```

```
hello,world!
```

```
ans =
```

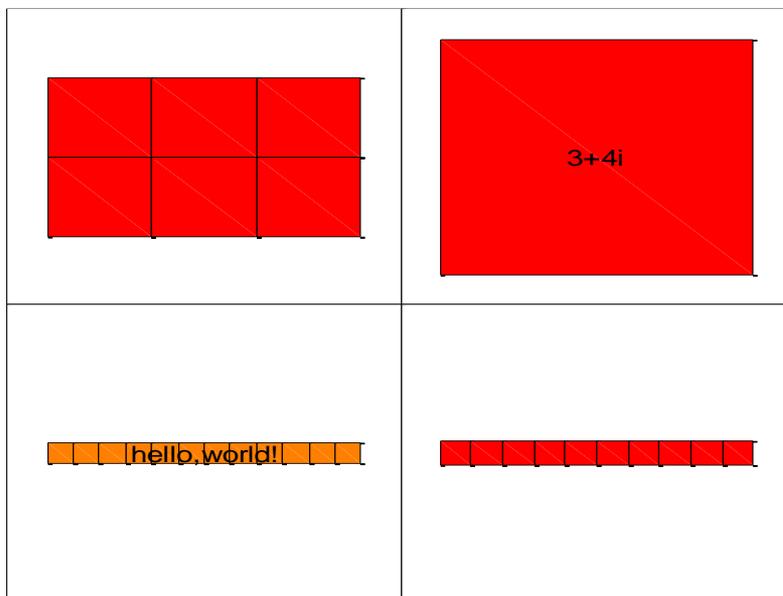
```
3.0000 + 4.0000i
```

```
ans =
```

```
1 2 3 4 5 6 7 8 9 10
```

单元数组的内容显示

```
>> cellplot(A)
```





单元内容获取

```
>> x = A(2,2)
```

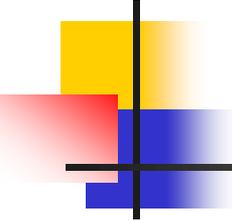
```
x =
```

```
[1x10 double]
```

```
>> x=A{2,2}
```

```
x =
```

```
1 2 3 4 5 6 7 8 9 10
```



单元内容删除

```
>> A(2,1)=[]
```

??? A null assignment can have only one non-colon index.

```
>> A{2,1}=[]
```

```
A =
```

```
[2x3 double] [3.0000 + 4.0000i]
      []      [1x10 double]
```



结构体(structures)

- 类似与单元数组，将不同类型的、相关的数据集成在一个变量中；
- 结构体中每个元素称为字段(**field**)；
- 每个字段(**field**)可以是任何数据类型：字符串、双精度数组、其他单元数组。不同字段可以包含不同的数据类型。



创建结构体

- 创建一个包含学生个人资料的结构体 `student`，可能的字段有：`name`、`id`、`scores`等。
- `student.name = '小明' ;`
`student.id = 'PB1234567' ;`
`student.scores = [98,92,90];`

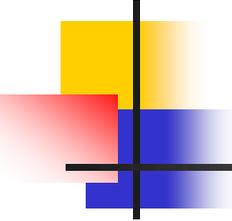


创建结构体

```
student =  
    name: '小明'  
    id: 'PB1234567'  
    scores: [98 92 90]
```

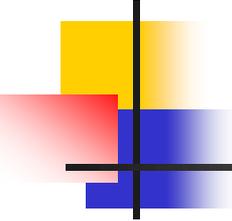
要加入第二个学生的资料，可以：

```
student(2).name = '小刚';  
student(2).id = 'PB2345678';  
student(2).scores = [75,100,86];
```



创建结构体

```
student =  
    1x2 struct array with fields:  
    name  
    id  
    scores
```

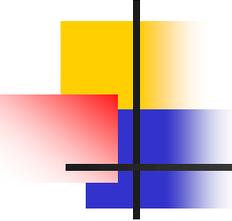


结构体

- 创建结构体变量circle:

```
>> circle.radius = 2.5;  
>> circle.center = [0,1];  
>> circle.linestyle = '--';  
>> circle.color = 'red'
```

```
circle =  
    radius: 2.5000  
    center: [0 1]  
linestyle: '--'  
    color: 'red'
```



结构体

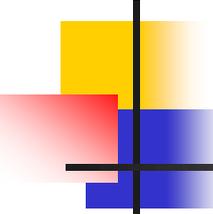
```
>> size(circle)
```

```
ans =
```

```
    1    1
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
circle	1x1	530	struct	



结构体

希望再加入一个圆：

```
>> circle(2).radius = 3.4;  
>> circle(2).color = 'green';  
>> circle(2).linestyle = ':';  
>> circle(2).center = [2.3 -1.2]
```

circle =

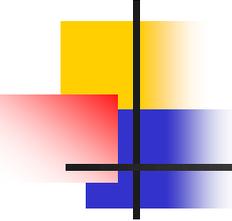
1x2 struct array with fields:

radius

center

linestyle

color



字段内容的获取

```
>> rad2 = circle(2).radius
```

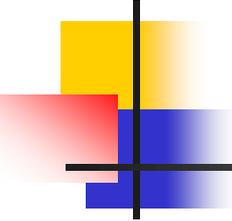
```
rad2 =
```

```
3.4000
```

```
>> area1 = pi*circle(1).radius^2
```

```
area1 =
```

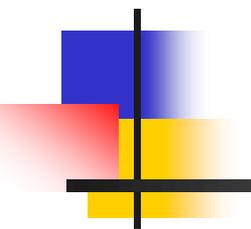
```
19.6350
```



获取结构体的字段信息

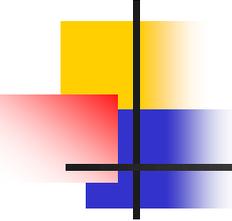
- `s(1,1).name = 'alice';`
- `s(1,1).ID = 0;`
- `s(2,1).name = 'lw';`
- `s(2,1).ID = 1;`

- `names = fieldnames(s)`



Part3:

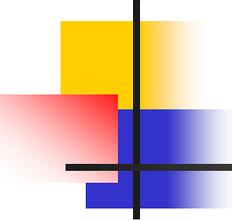
稀疏矩阵



稀疏矩阵

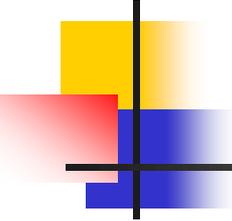
- **稀疏矩阵**：指大多数元素为0，只有少数非零元素的矩阵。

0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	6	0	0	0	0
0	0	0	0	0	0	0	0
0	7	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0



稀疏矩阵

- 实际应用中常常会遇到：
描述电路拓扑结构的矩阵、电网、交通。
- 若按普通方式(**Full Matrix**)存储稀疏矩阵，占用大量内存空间和运算时间。
- 稀疏矩阵的存储方式：
只存储其中的非0元素，并记录相应的行、列位置。



稀疏矩阵的创建

- 将全矩阵(Full Matrix)转换为稀疏矩阵.

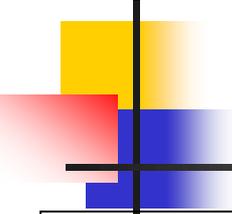
```
S=sparse(A)
```

例:

```
A=[0 0 5 0; 3 0 3 0; 0 0 0 1; 0 4 3 0]
```

```
S=sparse(A)
```

```
whos
```

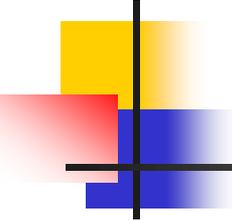


稀疏矩阵的创建

A = 0 0 5 0
3 0 3 0
0 0 0 1
0 4 3 0

S =(2,1) 3
(4,2) 4
(1,3) 5
(2,3) 3
(4,3) 3
(3,4) 1

Name	Size	Bytes	Class	Attributes
A	4x4	128	double	
S	4x4	92	double	sparse



稀疏矩阵的创建

■ 直接创建稀疏矩阵:

S=sparse(i,j,s,m,n)

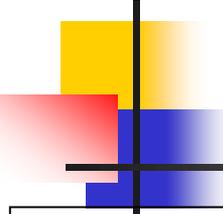
i 和 j 分别是矩阵非零元素的行和列指标向量，

s 是非零元素值向量，

m, n 分别是矩阵的行数和列数。

i、j、s 是长度相同的向量。

s(k) 的二维下标即是 i(k) 与 j(k)。



稀疏矩阵的创建

```
>> S = sparse([1 3 2], [1 2 4], [2 4 1], 3, 4)
```

```
S = (1,1)    2  
     (3,2)    4  
     (2,4)    1
```

- 也可以在sparse指令中加入第六个参数:

```
S=sparse(i,j,s,m,n,nzmax)
```

- 最后一个参数nzmax告诉matlab该稀疏矩阵最多有多少个非零元素，便于预先分配内存。

稀疏矩阵的创建

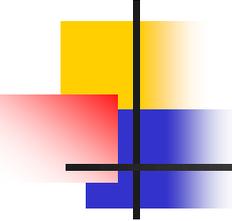
从文件中创建稀疏矩阵

利用`load`和`spconvert`函数可以从包含一系列下标和非零元素的文本文件中输入稀疏矩阵。

例：设文本文件 **T.txt** 中有三列内容，第一列是一些行下标，第二列是列下标，第三列是非零元素值。

```
1 3 5
2 1 3
2 3 3
3 4 1
4 2 4
4 3 3
```

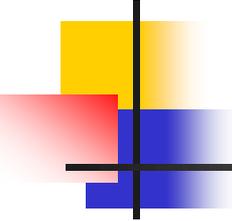
```
load T.txt
S=spconvert(T)
```



稀疏矩阵

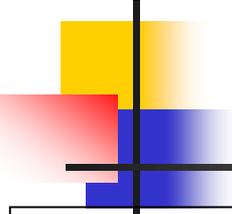
- 稀疏矩阵转换为全矩阵：

```
A=full(S)
```



稀疏矩阵信息查看

- **nnz(S)** : 返回S中非零元素的个数
- **nonzeros(S)** : 返回列向量, 含所有非零元素;
- **nzmax(S)** :
- **spy(S)** : 图形化形式表达稀疏矩阵;
- **[i, j, s]=find(S)**



稀疏矩阵运算

```
>> S = sparse([1 3 2], [1 2 4], [2 4 1], 3, 4);
```

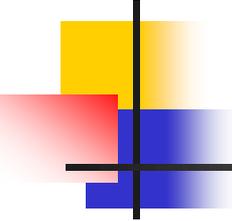
```
>> x = sin(S);
```

```
>> x = size(S);
```

```
>> A = full(S);
```

```
>> B = A.*S;
```

```
>> C = A + S;
```

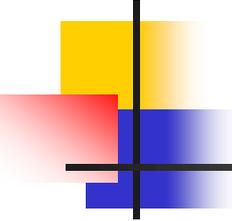


稀疏矩阵运算

- MATLAB提供了许多可以应用于稀疏矩阵处理的函数

Linear algebra.

- eigs - A few eigenvalues, using ARPACK.
- svds - A few singular values, using eigs.
- ilu - Incomplete LU factorization.
- luinc - Incomplete LU factorization.
- cholinc - Incomplete Cholesky factorization.
- normest - Estimate the matrix 2-norm.
- condest - 1-norm condition number estimate.
- sprank - Structural rank.



Matlab音频信号相关函数

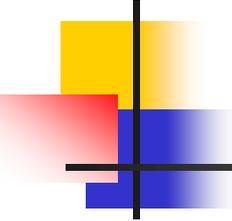
audioread %从文件中读取音频信号

audiowrite %将音频数据写入文件

sound %播放声音

soundsc %播放声音

audiorecorder %录音



录音

- **audiorecorder(Fs,nBits,nChannels)**

`recObj = audiorecorder;` % 默认: 单声道, 采样率`fs`
`= 8000bps`, 量化精度`nbits = 8bit`

`recordblocking(recObj, 5)` %录音5秒钟

`play(recObj);` %播放声音

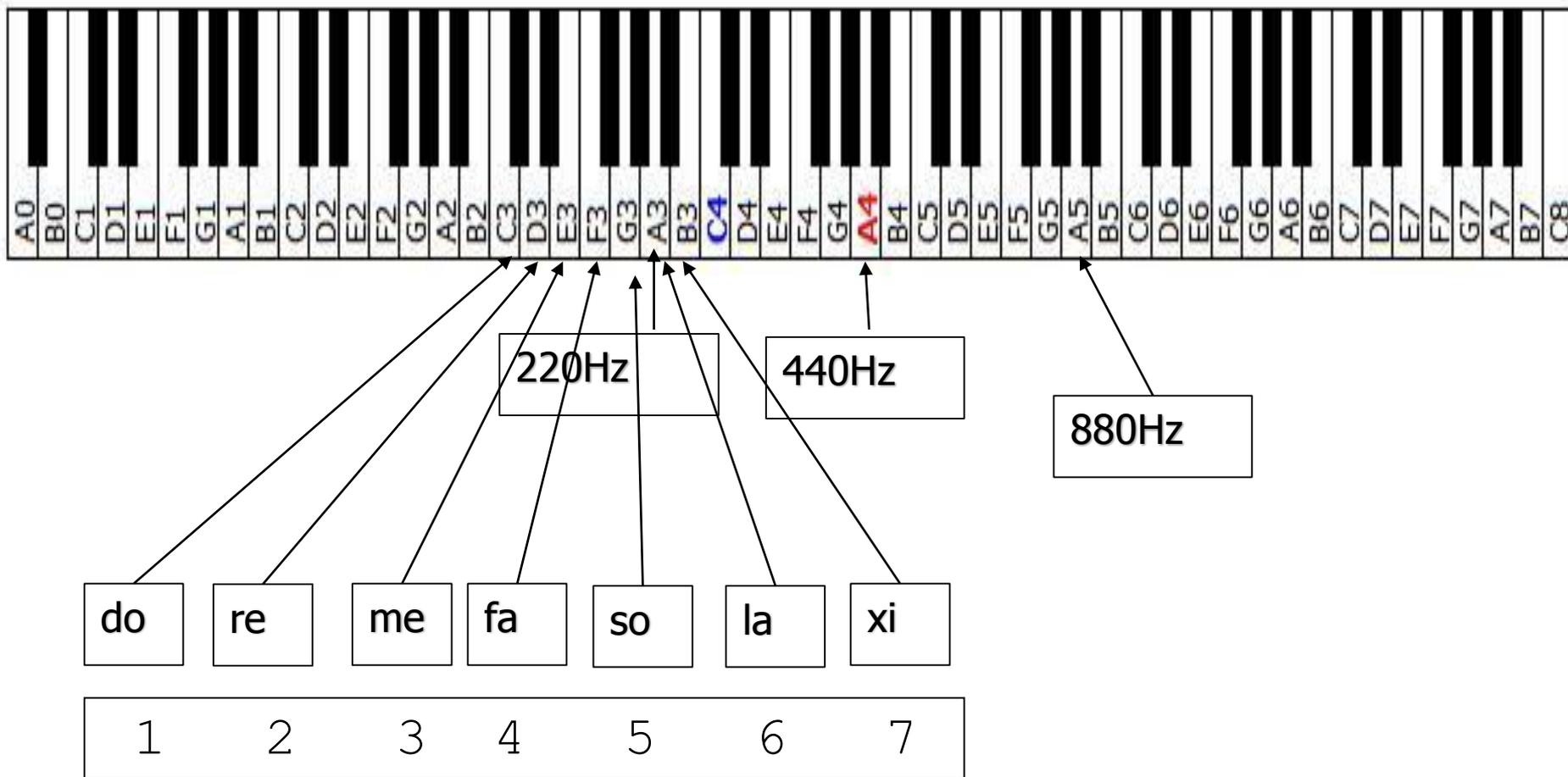
`y = getaudiodata(recObj);`

`plot(y);`

Matlab音频信号相关函数 (老版matlab)

wavread %从wav文件中读取语音信号
wavrecord %录音
wavwrite %将音频数据写入wav文件
sound %播放声音
soundsc %播放声音

音阶合成



音阶合成

■ 十二平均律

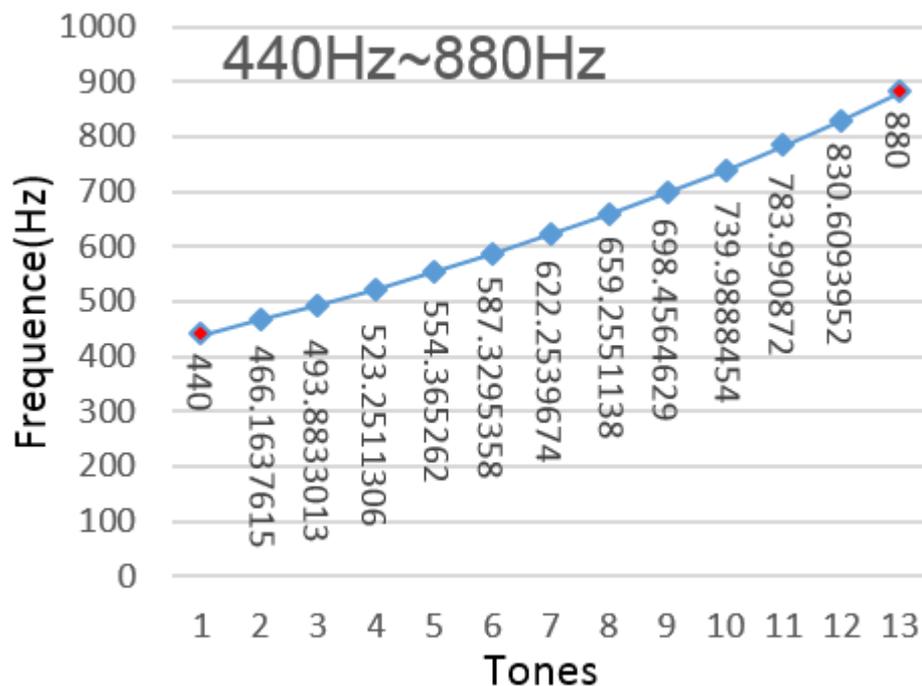
频率 (Hz) :

110-220-440-880

两个相邻的频率为一个八度

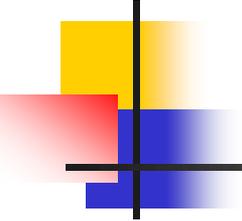
比如在440Hz-880Hz之间，
按等比关系分成12个单音，
每个单音的频率为：

$$440 \times 2^{\frac{k}{12}} \quad k = 0, 1, 2 \dots 12$$



图片来源: [《程序控》](#) 博客 --

<http://www.cnblogs.com/devymex/>



```
Fs = 44100; T = 1/Fs; t = 0:T:0.4;
fA=440; x = [0:12];
f = fA*2.^(x/12); flow = fA/2*2.^(x/12); fhigh = fA*2*2.^(x/12);
note = @(n) sin(2*pi*f(n)*t);
notelow = @(n) sin(2*pi*flow(n)*t);
notehigh = @(n) sin(2*pi*fhigh(n)*t) ;
notec = @(n) 0.6*notelow(n)+note(n)+0.6*notehigh(n);
m123 = [note(1),note(3),note(5)];
m123_c = [notec(1),notec(3),notec(5)];
mall_c =
[notec(1),notec(3),notec(5),notec(6),notec(8),notec(10),notec(12)];
sound(m123_c, Fs)
```

音阶合成



两只老虎

(大众乐谱网制谱)

法国古歌曲调
佚名填词

小快板



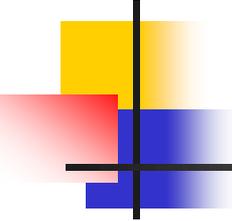
1=G 1 2 3 1 1 2 3 1 3 4 5 — 3 4 5 —

两只老虎，两只老虎，跑得快，跑得快。



5 6 5 4 3 1 5 6 5 4 3 1 1 5̣ 1 — 1 5̣ 1 —

一只没有耳朵，一只没有尾巴，真奇怪！真奇怪！



图像信号

灰度图像

- 灰度图像 \Leftrightarrow 二维矩阵A

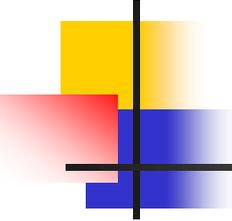
$$0 \leq A(i,j) \leq 255$$

(黑)

(白)

49	55	58	59	57	53
60	67	71	72	72	70
102	108	111	111	112	112
157	167	169	167	165	164
196	205	208	207	205	205
199	208	212	214	213	216
190	192	193	195	195	197
174	169	165	163	162	161





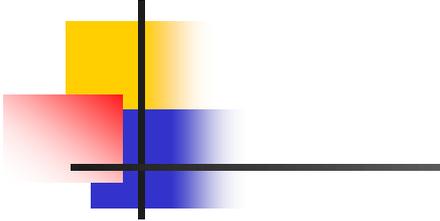
彩色图像(BMP格式)

- 彩色图像 (BMP格式) \Leftrightarrow 三维数组

$0 \leq A(i, j, 1) \leq 255$ 红色 (Red)

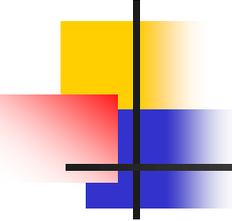
$0 \leq A(i, j, 2) \leq 255$ 绿色 (Green)

$0 \leq A(i, j, 3) \leq 255$ 蓝色 (Blue)



Cornell University Law School
Photograph by Cornell University Photography





其他图像格式

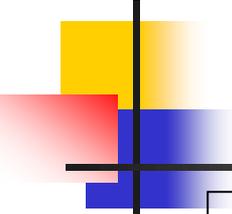
- JPEG (Joint Photographic Experts Group)
- GIF (Graphics Interchange Format)

基本目的：数据压缩

图像翻转

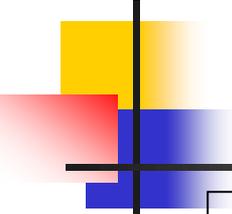
■ 图像左右翻转





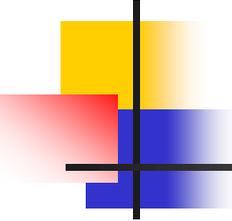
图像翻转

```
A = imread('LawSchool.jpg');  
[nr,nc,np] = size(A);  
for r = 1:nr  
    for c = 1:nc  
        for p = 1:np  
            B(r,c,p) = A(r,nc -c+1);  
        end  
    end  
end
```



图像翻转

```
A = imread('LawSchool.jpg');  
[nr,nc,np] = size(A);  
for r = 1:nr  
    for c = 1:nc  
        for p = 1:np  
            B(r,c,p) = A(r,nc -c+1);  
        end  
    end  
end
```



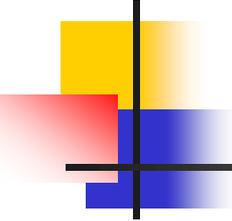
图像翻转

```
>> A = imread('LawSchool.jpg');  
>> B(:, :, 1) = fliplr(A(:, :, 1));  
>> B(:, :, 2) = fliplr(A(:, :, 2));  
>> B(:, :, 3) = fliplr(A(:, :, 3));  
>> imshow(B)
```

彩色图像->灰度图像



Cornell University Law School
Photograph by Cornell University Photography

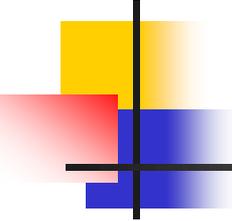


彩色图像->灰度图像

```
A =imread('lawschool.jpg');  
gA  = (A(:, :, 1)+A(:, :, 2)+A(:, :, 3))/3;  
imshow(gA);
```

彩色图像->灰度图像





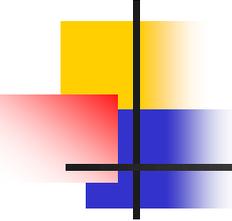
彩色图像->灰度图像

```
A =imread('lawschool.jpg');  
A = double(A);  
gA = (A(:, :, 1)+A(:, :, 2)+A(:, :, 3))/3;  
imshow(uint8(gA));
```

彩色图像->灰度图像



Cornell University Law School
Photograph by Cornell University Photography



彩色图像->灰度图像

- 由于人眼对红绿蓝三色敏感程度不同，常乘以不同比例因子。

```
A = imread('lawschool.jpg');  
A = double(A);  
gA = 0.3*A(:, :, 1) + 0.59*A(:, :, 2) + 0.11*A(:, :, 3);  
imshow(uint8(gA));
```

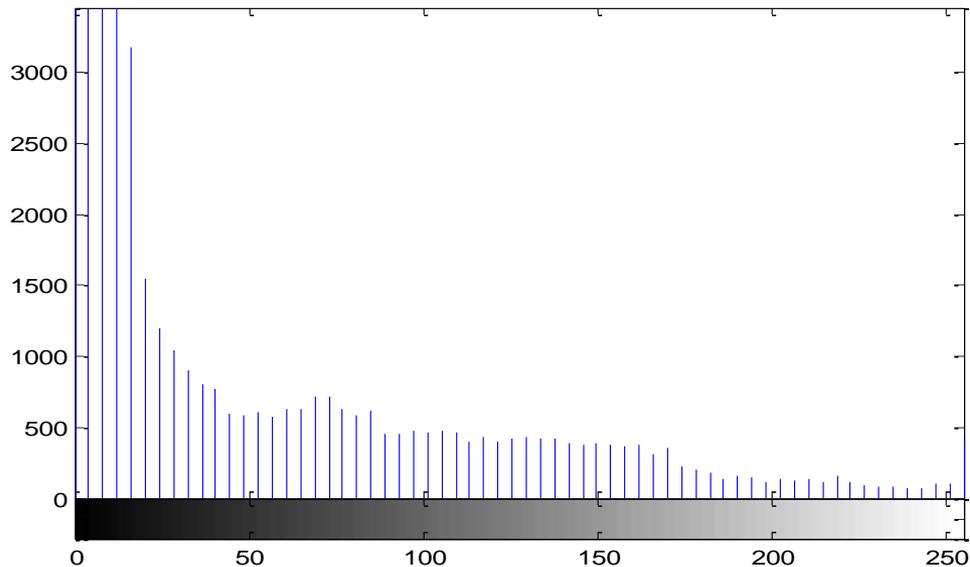
彩色图像->灰度图像

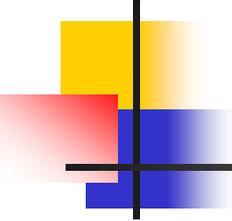


Cornell University Law School
Photograph by Cornell University Photography

图像直方图

- 对于一张灰度图，该图的直方图就是占各个灰度值的像素点的个数的统计；
- 直方图是图像的一种统计特性

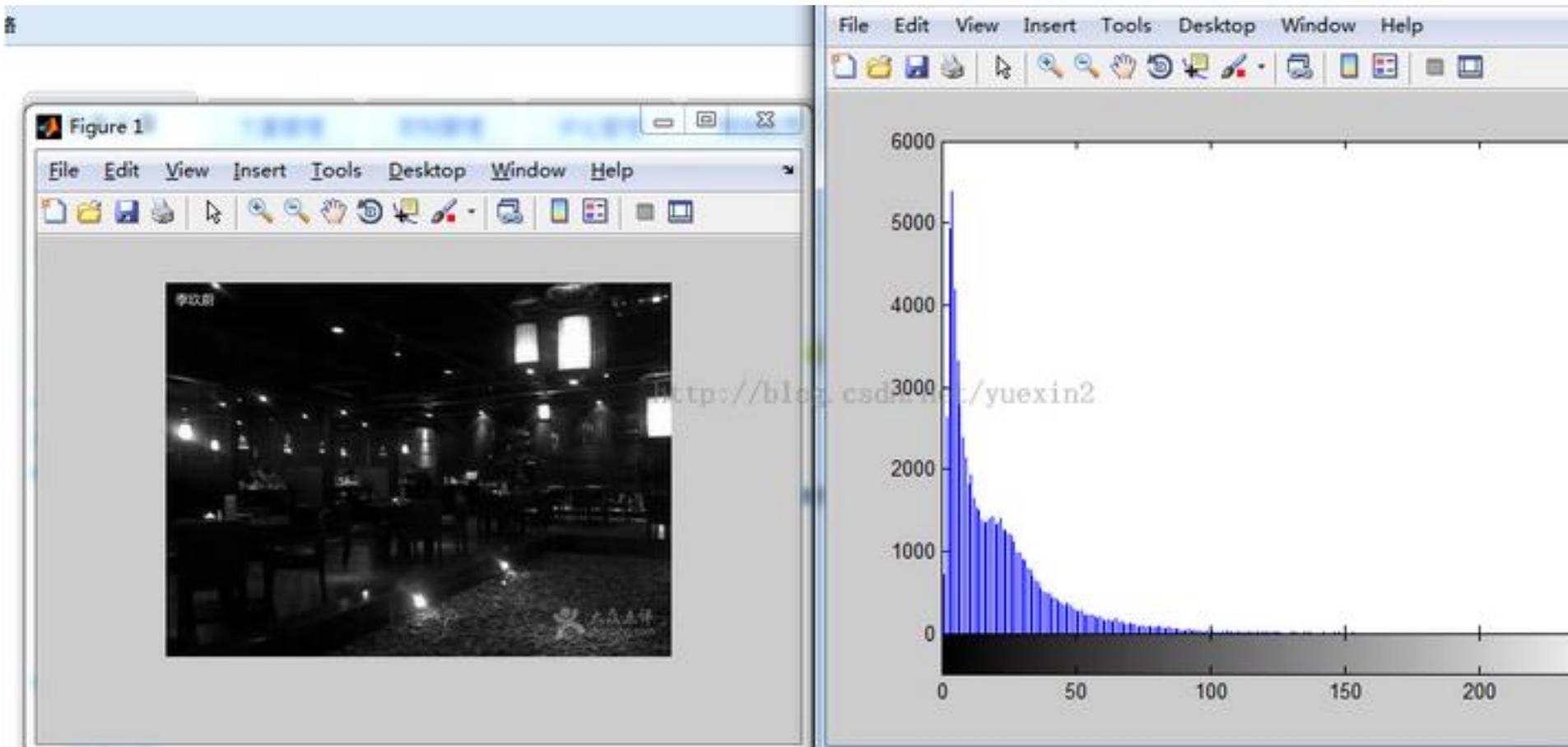




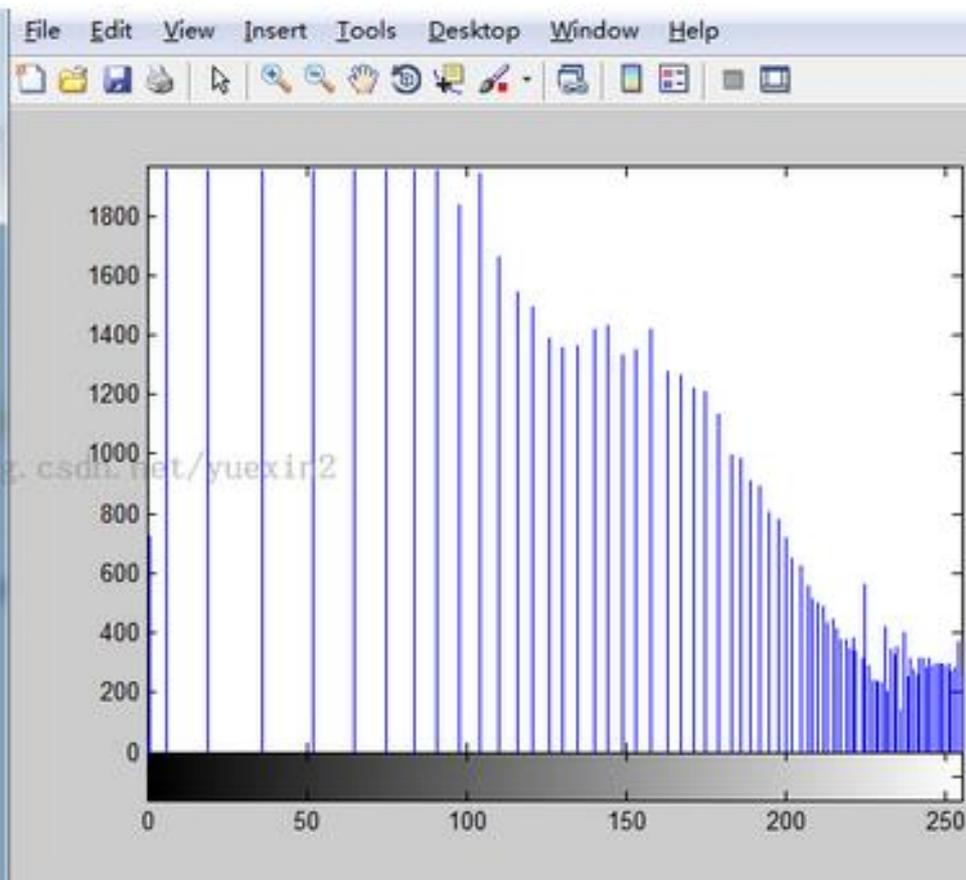
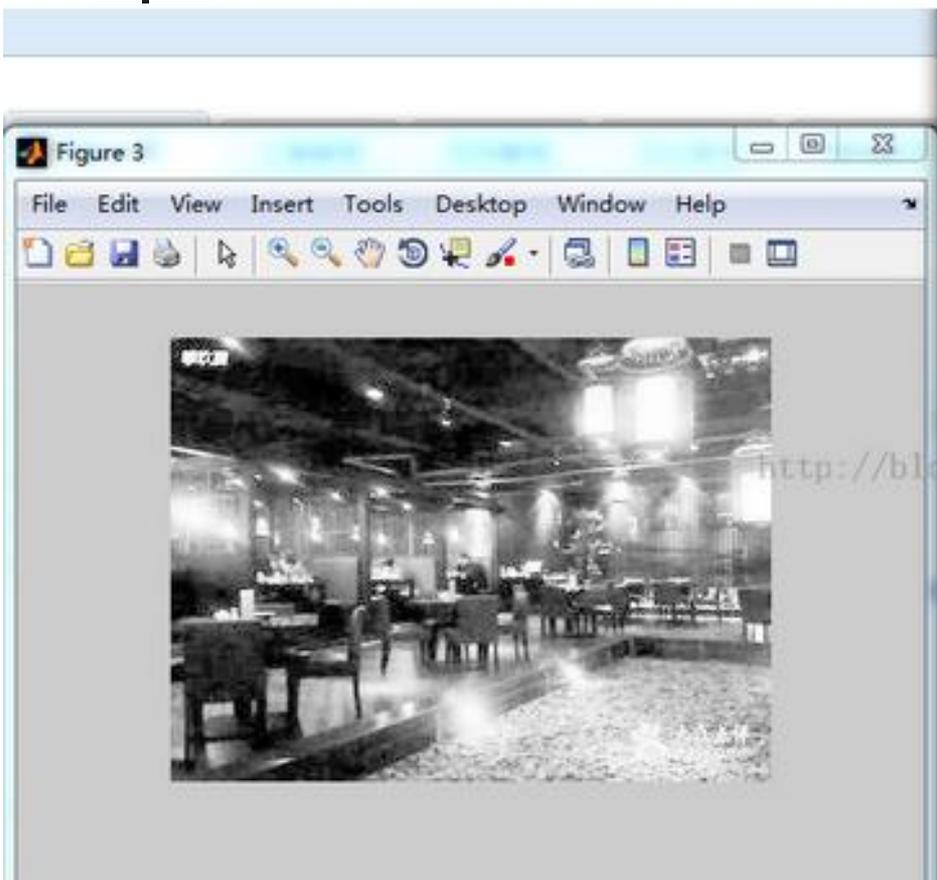
直方图均衡

- 直方图均衡化是通过拉伸像素强度分布范围来增强图像对比度的一种方法
- 均衡化指的是把一个分布 (给定的直方图) *映射* 到另一个分布 (一个更宽更统一的强度值分布), 所以强度值分布会在整个范围内展开

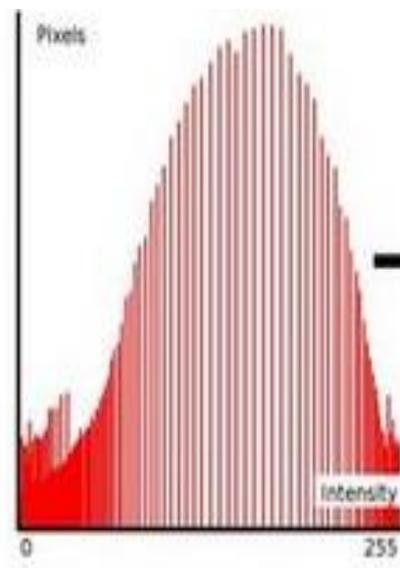
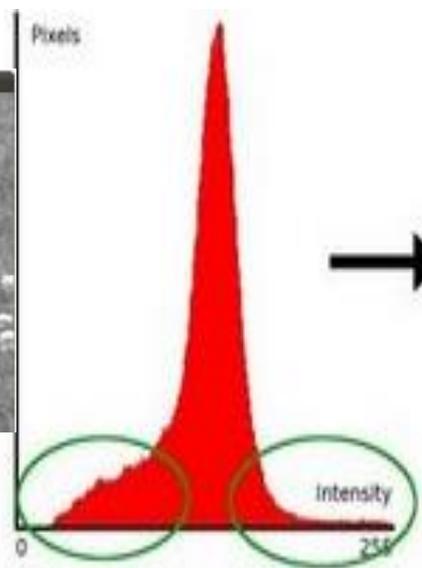
直方图均衡



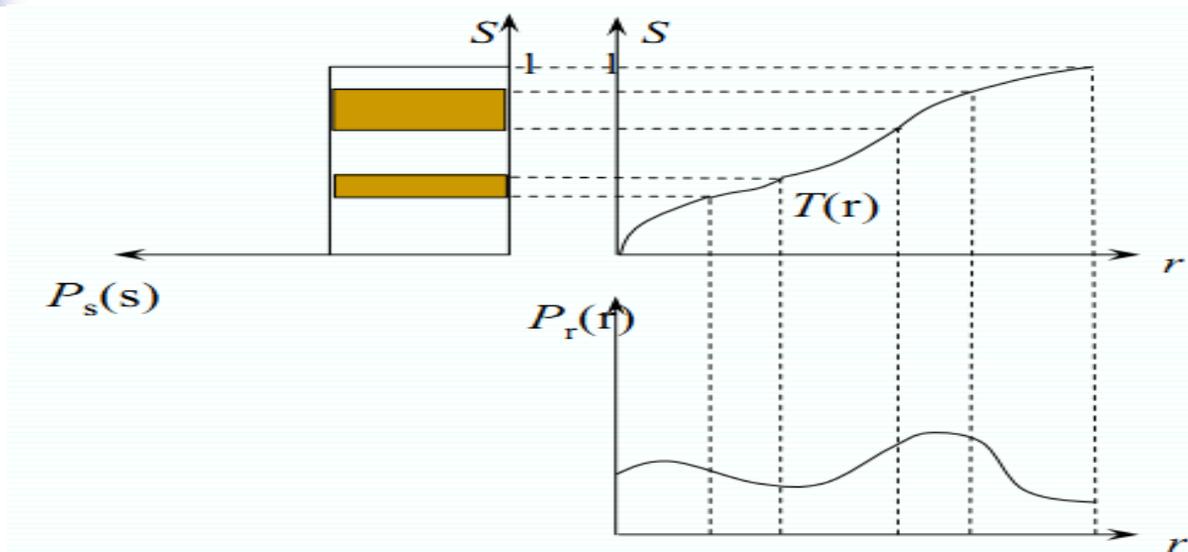
直方图均衡



直方图均衡

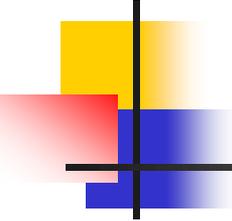


直方图均衡



对于 $s = T(r)$ 假定:

- (1) 在 $0 \leq r \leq 1$ 区间内, $T(r)$ 为单调递增函数, 且满足 $0 \leq T(r) \leq 1$
- (2) 变换 $r = T^{-1}(S)$ 存在, $0 \leq S \leq 1$, 也满足类似(1)的条件, $0 \leq r \leq 1$ 有 $0 \leq T(r) \leq 1$



直方图均衡

对于连续的函数， $P_r(r)$ 和 $P_s(s)$ 分别是灰度 r 和 s 的概率密度函数，可知：

$$P_s(s) = P_r(r)dr/ds$$

直方图均衡化的目的是保证每个灰度级的概率密度相等，即是一个常数：

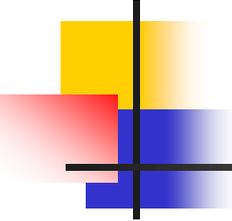
$$P_s(s) = 1/L$$

L 是均衡化后灰度的变化范围，在这里归一化为1，即：

$$P_s(s)=1 \Rightarrow ds = P_r(r)dr \Rightarrow s = \int P_r(r)dr$$

此式表明，当变换函数为原图像密度函数的分布函数时，能达到直方图均衡化目的。

对于离散的情况有： $s_k = \sum_{j=0}^k n_j/n$ *



直方图均衡

- $h(r_k) = n_k$
- 灰度级为 r_k 发生的概率估计值
$$p(r_k) = n_k/n \quad k = 0, 1, \dots, 256$$
- 希望新的灰度级分布: $p_s(s) = 1/256$
- 映射函数 $s = T(r)$