
بسم الله الرحمن الرحيم

پردازش تصاویر دیجیتالی
Digital Image Processing

❑ منابع:

۱. پردازش تصویر رقمی تألیف: رافائل گونزالس
ترجمه: دکتر مرتضی خادمی

2. **Digital Image Processing**, Rafael C. Gonzalez, Richard E. Woods,
2nd Edition, Prentice Hall

3. ***Digital Image Processing Using Matlab***, *Rafael C. Gonzalez, Richard E. Woods, Steven Eddins,* 1st Edition, Pearson, 2003.

4. ***Digital Image Processing*** , *William K. Pratt*, 4th Edition, John Wiley, 2007.

۵. اسلاید دکتر هادی سیدعربی- استادیار دانشکده مهندسی برق و کامپیوتر- دانشگاه تبریز

مقدمه

می توان یک تصویر را با یک تابع دوبعدی f(x,y) که x و y مختصات مکانی (Spatial Coordinates) و مقدار f برای هر زوج (x,y) شدت روشنایی (Intensity) یا سطح خاکستری (Gray level) نامیده می شود.

وقتی مقدار x و y و f با مقادیر محدود و گسسته دیجیتالی بیان شوند، تصویر را یک تصویر دیجیتالی می نامند.

یک تصویر دیجیتالی از تعدادی عناصر محدود با مقدار و موقعیت مختلف تشکیل شده است. این عناصر، عناصر تصویر (Picture element) یا پیکسل (Pixel) نامیده می شوند.

پردازش تصاویر دیجیتالی به معنی اعمال پردازش های مختلف روی یک تصویر دیجیتالی با استفاده از کامپیوتر دیجیتالی است.

مقدمه

مرز مشخصی بین پردازش تصویر از یک طرف و بینایی ماشین (Computer Vision) از طرف دیگر نمی توان مشخص کرد.

با این حال می توان سه نوع پردازش سطح پایین (Low level)، سطح میانی (Mid level) و سطح بالا (High level) تشخیص داد.

- پردازش سطح پایین شامل پردازش های ابتدایی مانند پیش پردازش هایی برای حذف نویز، بهبود کنتراست (Contrast) و فیلتر کردن تصویر است. مشخصه این نوع پردازش این است که ورودی و خروجی آن تصویر هستند.

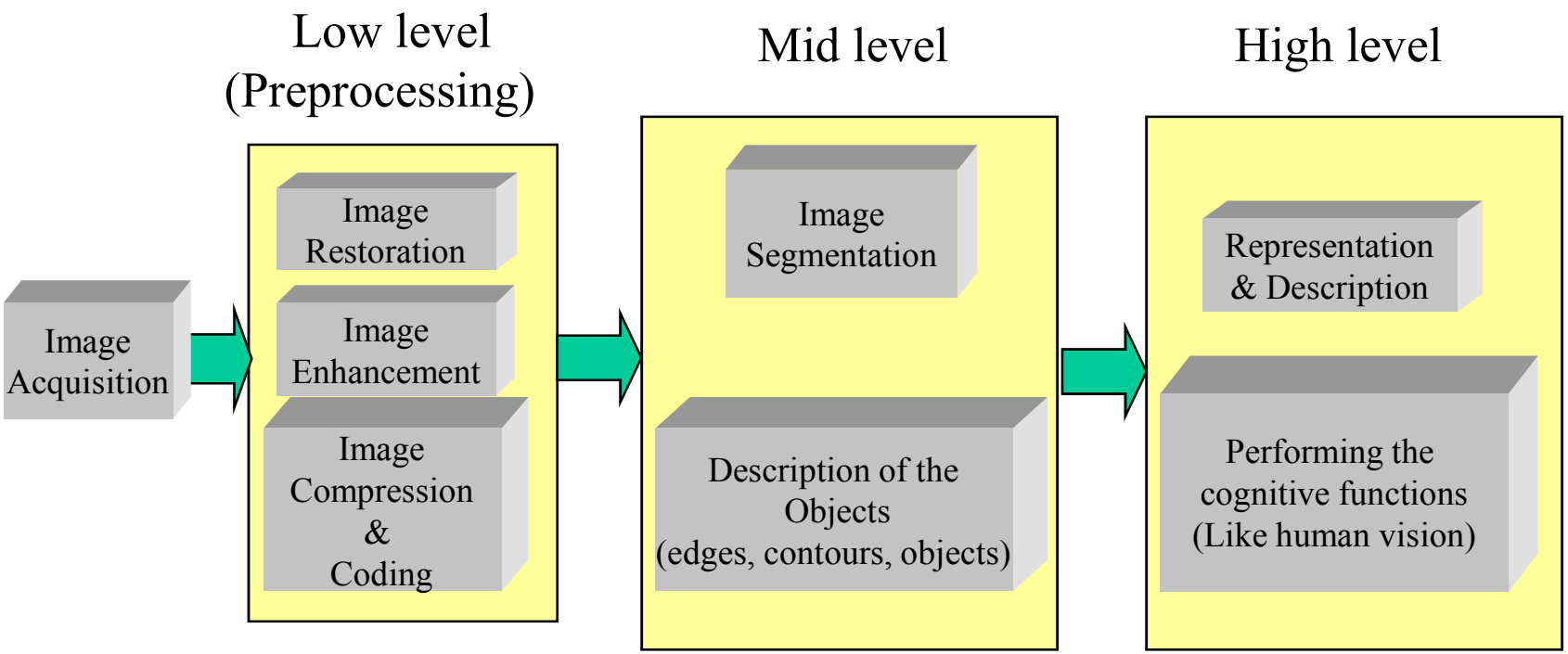
- پردازش سطح میانی شامل بخش بندی تصویر (Image Segmentation) به منظور تقسیم آن به نواحی و اشیاء مختلف، توصیف اشیاء به فرمی که برای پردازش کامپیوتر مناسب باشند و طبقه بندی یا تشخیص اشیاء مختلف است. ویژگی این پردازش این است که ورودی آن معمولا تصویر و خروجی آن صفاتی از اشیاء تصویر مانند لبه ها، کانتورها و تشخیص اشیاء است.

مقدمه

- پردازش سطح بالا شامل فهمیدن روابط بین اشیاء تشخیص داده شده (Making sense) ، استباط و تفسیر صحنه و انجام تفسیر و تشخیص هایی است که سیستم بینایی انسان انجام می دهد. بسیاری از پردازش های سطح بالا در حیطه بینایی ماشین قرار می گیرند.

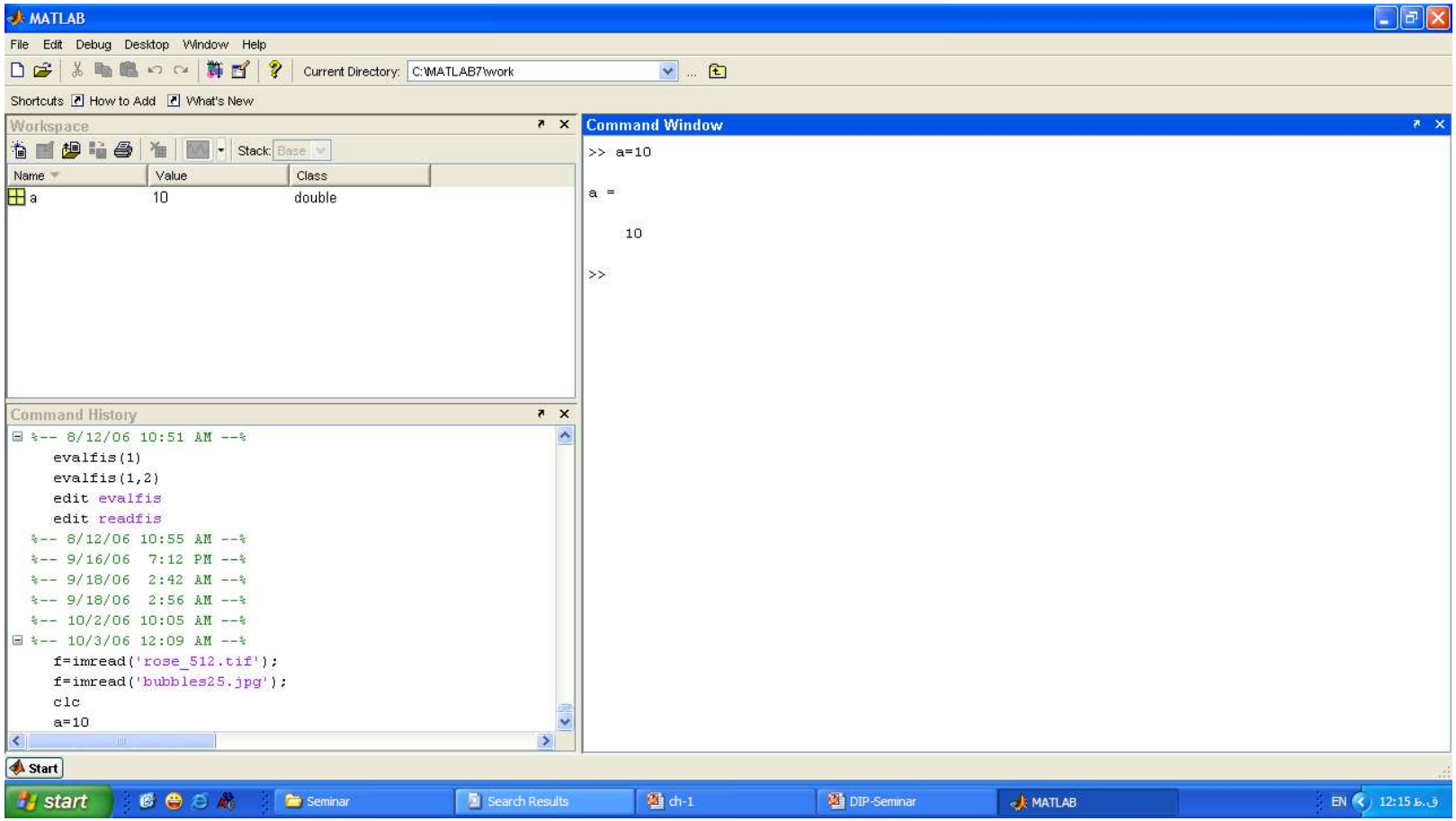
به طور مثال یک سیستم تشخیص اتوماتیک متن را در نظر بگیرید. پردازش برای مشخص کردن ناحیه حاوی متن، پیش پردازش تصویر حاصل، استخراج کاراکترها مختلف و تشخیص آن کاراکترها در حیطه بحث پردازش تصویر قرار دارند. استباط مفهوم و محتوای متن مورد نظر در حیطه بینایی ماشین یا آنالیز تصویر قرار می گیرد.

مقدمه


Low level
(Preprocessing)
Mid level
High level
Image
Acquisition
Image
Restoration
Image
Enhancement
Image
Compression
&
Coding
Image
Segmentation
Description of the
Objects
(edges, contours, objects)
Representation
& Description
Performing the
cognitive functions
(Like human vision)

مفاهیم پایه

❑ Matlab Desktop



Matlab Desktop

با وارد کردن دستور help در محیط Command Window تمام جعبه ابزارهای Matlab معرفی می شوند.

با وارد کردن دستور help images دستورات جعبه ابزار پردازش تصویر با یک توضیح مختصر لیست می شوند. برای توضیح بیشتر برای هر دستور، می توان از help به همراه نام دستور استفاده کرد.

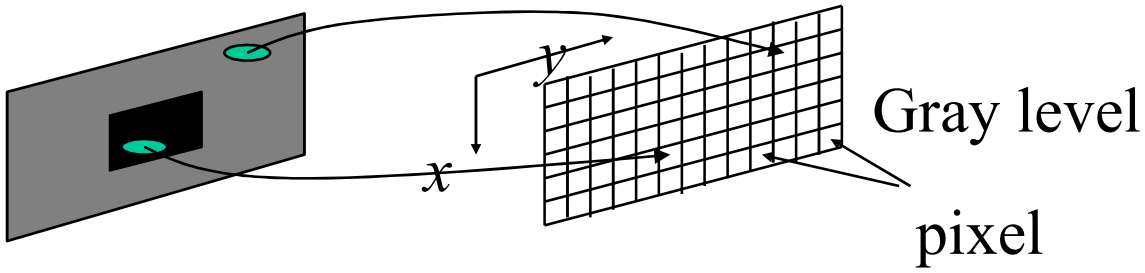
مثال:

```
>> help imshow
```

مفاهیم پایه

❑ نمایش تصویر دیجیتالی

می توان یک تصویر را با یک تابع دوبعدی f(x,y) که x و y مختصات مکانی و مقدار f برای هر زوج (x,y) شدت روشنایی یا سطح خاکستری (در تصاویر تکرنگ) نامیده می شود، نمایش داد.



Original picture
$f(x, y)$

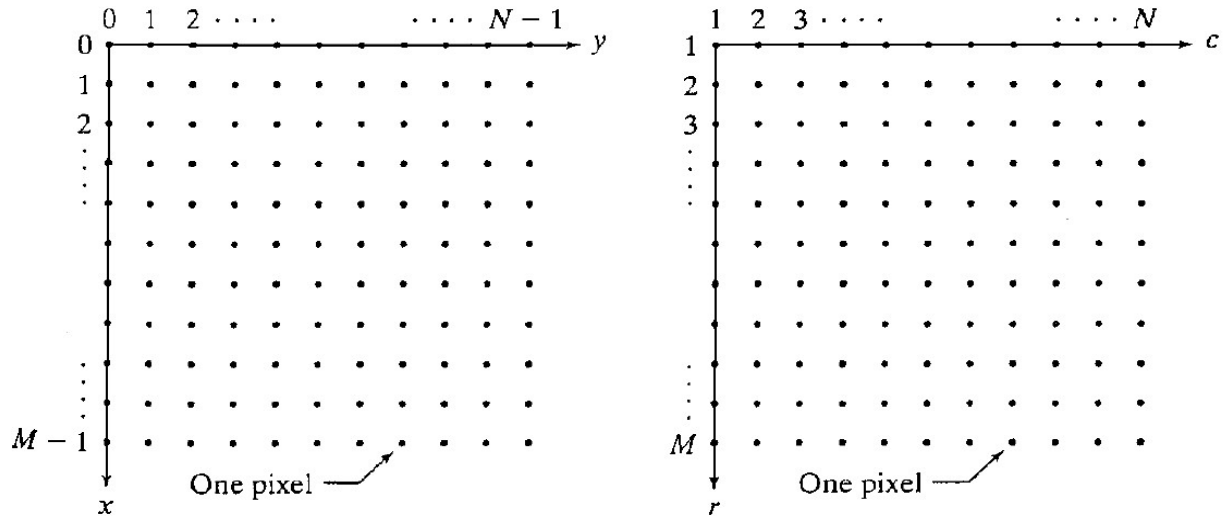
Digital image
$f[x, y]$

مفاهیم پایه

- نمایش تصویر دیجیتالی

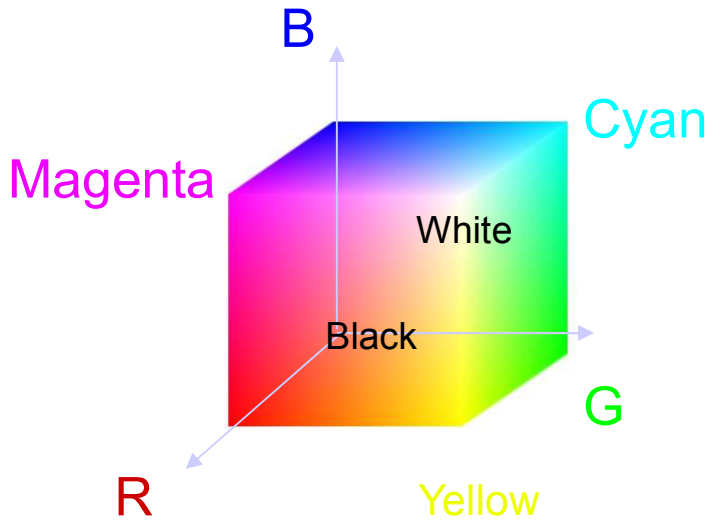
در بسیاری از کتابهای پردازش تصویر، مبدا تصویر به صورت زیر تعریف می شود: $(x,y)=(0,0)$. مولفه x شماره سطر و مولفه y شماره ستون یک پیکسل را نشان می دهند.

به دلیل اینکه در Matlab اندیس مولفه آرایه ها از 0 شروع نمی شود، در این محیط مبدا به صورت زیر تعریف می گردد: $(r,c)=(1,1)$



مفاهیم پایه

تصاویر رنگی، از ترکیب چند تصویر دوبعدی تشکیل شده اند. به طور مثال در سیستم رنگی RGB هر تصویر از سه مولفه تصویر قرمز، سبز و آبی تشکیل شده است.



مفاهیم پایه

تصویر به عنوان یک ماتریس: با نحوه نمایش تصویر معرفی شده، می توان هر تصویر دیجیتالی را به صورت ماتریسی به فرم زیر نمایش داد:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

هر مولفه ماتریس یک پیکسل را مشخص می کند.

این تصویر به صورت زیر به عنوان یک ماتریس در محیط Matlab تعریف می شود.

$$\mathtt{f} = \begin{bmatrix} \mathtt{f(1,1)} & \mathtt{f(1,2)} & \cdots & \mathtt{f(1,N)} \\ \mathtt{f(2,1)} & \mathtt{f(2,2)} & \cdots & \mathtt{f(2,N)} \\ \vdots & \vdots & & \vdots \\ \mathtt{f(M,1)} & \mathtt{f(M,2)} & \cdots & \mathtt{f(M,N)} \end{bmatrix}$$

مفاهیم پایه

•خواندن تصویر:

در محیط Matlab می توان یک فایل تصویری را با تابع imread با فرمت imread('filename') خوانده و به یک ماتریس تبدیل کرد.

مثال: f= imread('D:\myimages\chestxray.jpg');

اگر ; (سمی کولن) در آخر دستور قرار داده نشود، هنگام خواندن تصویر محتوای ماتریس تصویر یعنی f را نیز نمایش خواهد داد.

• اغلب فرمت های تصویر با دستور فوق قابل خواندن هستند. مانند:

TIFF, JPEG, GIF, BMP, PNG, PGM, ….

با تابع size می توان ابعاد تصویر خوانده شده، یعنی ماتریس تصویر را مشخص کرد:

```
>> size(f)
 ans=
        1024    1024
```

مفاهیم پایه

خواندن تصویر:

می توان تابع فوق را به فرم زیر نیز استفاده کرد.

```
[M,N]=size(f);
```

دستور فوق تعداد سطرها (M) و ستونهای تصویر (N) را مشخص می کند.

تابع who متغیرهای فعلی را نشان می دهد و تابع whos اطلاعات بیشتری در مورد متغیرها و آرایه ها ارائه می دهد.

```
>> whos f
```

Name	Size	Bytes	Class
f	1024*1024	1048576	uint8 array

; (semicolon) در انتهای تابع whos تاثیری نداشته و نوشته نمی شود.

مفاهیم پایه

نمایش تصاویر:

در محیط Matlab desktop می توان از تابع imshow برای نمایش تصاویر استفاده کرد.

imshow (f,G)

که f ماتریس تصویر خوانده شده و G تعداد سطوح خاکستری است. مقدار پیش فرض آن ۲۵۶ است.

با استفاده از فرمت زیر: imshow(f,[low high])

پیکسلهایی که مقادیر آنها کمتر یا مساوی low باشد سیاه و پیکسلهایی که مقدار آنها بزرگتر یا مساوی high باشند، سفید نشان داده می شوند.

با استفاده از فرمت: imshow(f,[ ]) مقدار low برابر مینیمم f و مقدار high برابر ماکسیمم f در نظر گرفته می شود. برای تصاویری که دارای محدوده دینامیکی کمی هستند، این روش مناسب تر است.

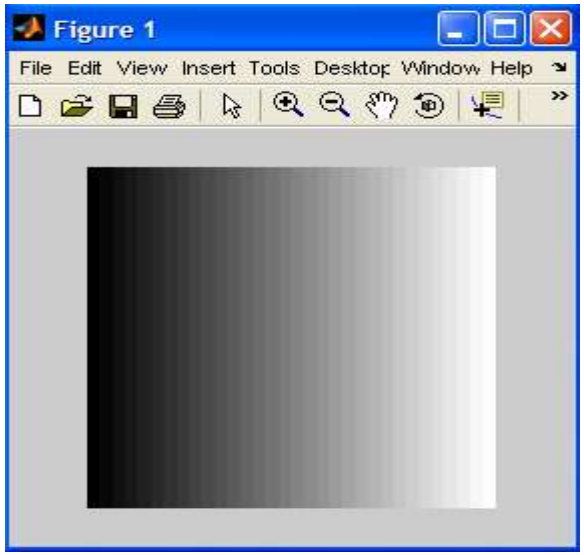
مفاهیم پایه

مثال:

```
>> f=imread('c:\matlab7\work\gradiant.bmp');
```

می توان دستور فوق را به صورت زیر نیز نوشت چرا که c:\matlab7\work\ دایرکتوری فعلی (Current Directory) محسوب شده ونیازی به دادن مسیر نیست.

```
>> f=imread('gradiant.bmp');
>> imshow (f,[ ])
>> size(f)
ans =
        512   512
```



تصویر نمایش داده شده را با استفاده از Export در منوی File، می توان با فرمت دلخواه ذخیره کرد.

مفاهیم پایه

اگر f و g دو آرایه تصویر خوانده شده باشند، با اجرای دستورات زیر:

```
>> imshow(f,[ ])
>> imshow(g,[ ])
```

ابتدا تصویر f و سپس g در پنجره figure 1 نمایش داده خواهند شد. برای نمایش همزمان هر دو تصویر، می توان نوشت:

```
>> figure(1); imshow(f,[ ])
>> figure(2); imshow(g,[ ])
```

مفاهیم پایه

نوشتن تصویر

یک ماتریس تصویر (f) با فرمت زیر می تواند به عنوان یک تصویر با فرمت دلخواه ذخیره گردد:

imwrite (f, ‘filename’)

فرمتهای متداول برای نوشتن تصویر : TIFF, JPEG, BMP, PNG, PGM,

مثال:

```
>> imwrite (f, ‘D:\gradiant2.tif’)
>> imwrite (f, ‘D:\gradiant2’, ‘tif’)
```

یا

مفاهیم پایه

با تابع imfinfo می توان جزئیات فایل ذخیره شده را مشاهده کرد:

imfinfo filename

```
>> imfinfo D:\gradiant2.tif
```

می توان این تابع را به فرم زیر استفاده کرد:

```
>> k= imfinfo (‘D:\gradiant2.tif ‘);
```

در این حالت متغیر به صورت structure تعریف شده و مشخصه های فایل به صورت فیلدهایی قابل دسترسی خواهند بود :

```
k.Height    k.Width   k.Format  k.FileSize
```

مفاهیم پایه

کلاس داده ها (Data Classes): گر چه در در نمایش تصویر، مختصات مکانی با اعداد صحیح نمایش داده می شوند، ولی مقدار پیکسلها محدود به مقادیر صحیح نیستند.

نام	توصیف	کلاس
double	اعداد ممیز شناور با دقت مضاعف (۸ بایت)	عددی (Numeric)
uint8	اعداد صحیح بدون علامت (یک بایت)	عددی (Numeric)
uint16	اعداد صحیح بدون علامت (دو بایت)	عددی (Numeric)
uint32	اعداد صحیح بدون علامت (چهار بایت)	عددی (Numeric)
int8	اعداد صحیح علامت دار (یک بایت)	عددی (Numeric)
int16	اعداد صحیح علامت دار (دو بایت)	عددی (Numeric)
int32	اعداد صحیح علامت دار (چهار بایت)	عددی (Numeric)
single	اعداد ممیز شناور (٤ بایت)	عددی (Numeric)
char	کاراکترها (دو بایت)	کاراکتر (Character)
logical	مقادیر 0 و 1 (یک بایت)	منطقی (Logical)

مفاهیم پایه

تمام محاسبات عددی در Matlab با مقادیر Double انجام می گیرد. بنا بر این، این نوع کلاس یکی از رایج ترین کلاس داده در پردازش تصویر محسوب می شود.

کلاس uint8 نیز یکی از متداول ترین نوع کلاس داه ها به خصوص در خواندن تصاویر ذخیره شده محسوب می شود. این کلاس متداول ترین کلاس داده برای تصاویر با پیکسلهای ۸ بیتی است.

کلاس character نمایش کاراکترها را به صورت Unicode نگه می دارد.

مفاهیم پایه

انواع تصاویر (Image Types):

جعبه ابزار پردازش تصویر چهار نوع تصویر را پشتیبانی می کند:

Intensity Image

Binary Images

Indexed Images

RGB Images

پردازشهای سیاه و سفید اغلب روی تصاویر Binary و Intensity صورت می گیرد و تاکید بیشتری روی این نوع تصاویر خواهیم داشت. با انتخاب یک آستانه، می توان تصاویر Intensity را به Binary تبدیل کرد.

تصاویر RGB و Indexed در بحث پردازش تصاویر رنگی استفاده می گردند.

مفاهیم پایه

اگر A یک آرایه عددی شامل 0 و 1 باشد، با دستور زیر این آرایه به یک آرایه منطقی تبدیل می شود.

B= logical(A)

اگر A شامل مولفه هایی غیر از 0 و 1 باشد، تابع فوق مقادیر غیر صفر را به 1 و مقادیر 0 را به 0 منطقی تبدیل می کند.

برای بررسی اینکه آیا یک آرایه logical است یا نه، از تابع زیر استفاده می گردد.

islogical(B)

اگرB منطقی باشد، تابع مقدار یک و در غیر این صورت مقدار صفر بر می گرداند.

یک تصویر با کلاس داده (Data class)و نوع تصویر (Image type) مشخص می گردد. مثال: uint8 intensity image

بعضی توابع در جعبه ابزار، از تمام کلاس داده ها پشتیبانی می کنند. ولی برخی از آنها فقط از کلاس خاصی پشتیبانی می کنند. به طور مثال، پیکسلهای یک تصویر باینری، فقط می توانند از نوع logical باشند.

مفاهیم پایه

تبدیلات کلاس داده ها :

فرمت کلی برای تبدیل کلاس داده ها به صورت زیر است:

B=data_class_name(A)

به طور مثال اگر A از نوع uint8 باشد، دستور زیر آنرا به نوع double تغییر می دهد:

B=double(A)

تابع فوق بسیار متداول است، چرا که Matlab انتظار دارد عملوندها در محاسبات عددی از نوع double باشند.

اگر C یک متغیر از نوع double باشد دستور زیر آنرا به uint8 تبدیل می کند:

D=uint8(C)

از آنجایی که محدوده uint8 برابر [0 255] است، دستور فوق برای مقادیر کوچکتر از 0 مقدار صفر و برای مقادیر بزرگتر از 255 مقدار 255 قرار می دهد. بنابراین تغییر مقیاس مناسب (proper scaling) قبل از عمل تبدیل ضروری است.

مفاهیم پایه

تبدیلات نوع تصاویر:

جدول زیر تبدیلات تصاویر را نشان می دهد:

نام تابع	نوع تصویر تبدیل شده	کلاس های ورودی معتبر
im2uint8	uint8	Logical,uint8,uint16, double
im2uint16	uint16	Logical,uint8,uint16, double
mat2gray	Double (in range [0,1])	double
im2double	double	Logical,uint8,uint16, double
im2bw	logical	uint8,uint16, double

مفاهیم پایه

تبدیلات نوع تصاویر:

تابع im2uint8 مقادیری از آرایه ورودی را که کمتر از 0 باشند برابر 0 قرار داده و مقادیری که بیشتر از 1 باشند، برابر 255 قرارمی دهد. مقادیر بین 0 و 1 را نیز به 255 ضرب کرده و به نزدیکترین عدد صحیح گرد می کند.

مثال

```
>> f=[-1, 0.25; 1.5  0]
f=
   -1.0000   0.2500
    1.5000        0
>> g=im2uint8(f)
g=
         0    64
       255     0
```

- تابع im2uint16 به طور مشابه عمل کرده و فقط ضرب را در 65535 انجام می دهد.

مفاهیم پایه

تبدیلات نوع تصاویر:

تابع mat2gray آرایه ورودی دلخواه از نوع double را دریافت و به آرایه double در بازه [0,1] تبدیل می کند. فرمت تابع به صورت زیر است:

g=mat2gray(f, [fmin,fmax])

که g دارای مقادیر 0(سیاه) و 1 (سفید) است. مقادیر کوچکتر از fmin به صفر و مقادیر بزرگتر از fmax به یک نگاشت شده و مقادیر بین آنها با رابطه زیر تبدیل می شود:g(f)=(f-fmin)/(fmax-fmin)

اگر مقادیر fmin,fmax در تابع ذکر نشود، این مقادیر برابر ماکسیمم و مینیم تابع f در نظر گرفته می شوند.

مثال

```
>> f=[-1, 0.25; 1.5  0]
f=
   -1.0000   0.2500
    1.5000        0
>> g=mat2gray(f)
g =
         0    0.5000
    1.0000    0.4000
```

مفاهیم پایه

تابع im2double آرایه ورودی را به بازه [0,1] تبدیل می کند.اگر ورودی از نوع uint8 باشد، مقادیر ورودی را به 255 و اگر از نوع uint16 باشد به 65535 تقسیم می کند.

مثال

```
>> f=uint8([10 60;128 255])
f =
   10   60
  128  255
>> g=im2double(f)
g =
    0.0392    0.2353
    0.5020    1.0000
```

مفاهیم پایه

تابع im2bw آرایه ورودی را به تصویر باینری تبدیل می کند:

g=im2bw(f,T)

T مقدار آستانه و در بازه [0,1] قرار دارد. بنا بر این ابتدا آرایه ورودی باید در بازه [0,1] قرار گیرد. مقدار خروجی برای مقادیر کوچکتر از آستانه 0 و برای مقادیر بزرگتر از آستانه 1 خواهد بود.پیش فرض برای آستانه، 0.5 است.

```
>> f=uint8([10 60;128 255])
f =
   10   60
  128  255
>> g1=im2double(f)
g1 =
    0.0392    0.2353
    0.5020    1.0000
>> g=im2bw(g1,0.6)
g =
     0     0
     0     1
```

مفاهیم پایه

آرایه ها:

بردار سطری (Row vector) :

```
>> v=[5 6 7 8 9];
      >> v(1)
ans=
5
>> v(3)
ans=
7
```

اندیس آرایه ها از یک شروع می شود

بردار سطری با گرفتن ترانسپوز به بردار ستونی (column vector)تبدیل می شود:

```
>> w=v';
```

مفاهیم پایه

v=[5 6 7 8 9]

```
>> v(2:4)
ans=
6    7   8
>> v(3:end)
ans=
7    8   9
>> v(1:2:end)
ans=
5    7   9
>>v(end:-2:1)
ans=
9    7    5
```

بردار ستونی

```
>> v(:)
ans=
  5
  6
  7
  8
  9
```

```
>> v([1 3 5])
ans =
   5    7    9
```

مفاهیم پایه

تابع linspace:

x=linspace(a,b,n)

این تابع بردار x را به n مولفه که با فواصل خطی بین a و b قرار می گیرند تولید می کند.

```
>> x=linspace(1,21,11)
x =
1   3   5  7  9  11  13  15  17  19  21
```

مفاهیم پایه

ماتریس ها:

```
>> A=[1 2 3; 4 5 6; 7 8 9]
A =
   1    2    3
   4    5    6
   7    8    9

>> A(3,2)
ans =
        8
>> C2=A(:,2)
C2 =
     2
     5
     8
```

```
>> R3=A(3,:)
R3 =
      7    8    9
>> A1=A(1:3,2:3)
A1 =
      2    3
      5    6
      8    9

>> A(end,end-1)
ans =
        8
```

مفاهیم پایه

مثال: یک فایل تصویری را به صورت ماتریس f خوانده و اعمال زیر را روی آن انجام داده ونتیجه را توجیه کنید:

```
f=imread('cameraman.tif');
f1= f(end:-1:1,:);
f2= f(:,end:-1:1);
f3= f(1:2:end,1:2:end);
figure(1);imshow(f,[])
figure(2);imshow(f1,[])   % Vertical Flip
figure(3);imshow(f2,[])   % Horizontal Flip
figure(4);imshow(f3,[])
```

Fig. 1

Fig. 2

Fig. 3

Fig. 4

مفاهیم پایه

ماتریس ها:

دستور A([a b],[c d]) یک ابزار قدرتمند برای انتخاب اعضای ماتریس است. این دستور عضوهای ماتریس را به صورت زیر انتخاب می کند:

(سطر a، ستون c) ، (سطر a، ستون d)، (سطر b، ستون c)، (سطر b، ستون d)

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

>> A2=A([1 3],[2 3])
A2 =
     2    3
     8    9
```

❑ روش آدرس دهی:

```
>> D=logical([1 0 0; 0 0 1; 0 1 0])
D =
     1    0    0
     0    0    1
     0    1    0
>> A(D)
ans =
         1
         8
         6
```

مفاهیم پایه

دستور A(:) ، ماتریس را به یک بردار ستونی تبدیل می کند.

تابع sum(v) تمام موالفه های بردار v را با هم جمع می کند:

```
>> sum(A)
```

این دستور عناصر در هر ستون را جمع می زند

```
ans =
   12    15    18
>> sum(sum(A))
ans =
   45
>> S= sum(A(:))
S=
    45
```

- تابع ndims ابعاد آرایه ها را مشخص می کند:

```
>> d=ndims(A)
   d=
       2
```

مفاهیم پایه

تعدادی از توابع مهم برای تولید آرایه های خاص:

zeros(M,N) یک ماتریس 0 با ابعاد M*N و کلاس double تولید می کند.

ones(M,N) یک ماتریس 1 با ابعاد M*N و کلاس double تولید می کند.

true(M,N) یک ماتریس منطقی از 1 ها به ابعاد M*N

false(M,N) یک ماتریس منطقی از 0 ها به ابعاد M*N می سازد.

rand(M,N) یک ماتریس به ابعاد M*N که موالفه های آن اعداد تصادفی با توزیع یکنواخت در بازه [0,1] هستند.

randn(M,N) یک ماتریس به ابعاد M*N که موالفه های آن اعداد تصادفی با توزیع گوسی با میانگین صفر و واریانس یک هستند.

magic(M) یک ماتریس مربعی به ابعاد M*M تولید می کند به نحوی که جمع هر سطر و ستون و قطر اصلی آن با هم برابر هستند.

اگر فقط یک مقدار در توابع فوق تعریف شود، ماتریس ها مربعی خواهند بود.

مفاهیم پایه

```
>> magic(3)
ans =
     8     1     6
     3     5     7
     4     9     2
```

```
>> B=rand(3,2)
B =
    0.9501    0.4860
    0.2311    0.8913
    0.6068    0.7621
```

```
>> C=4*ones(3,4)
C =
     4     4     4     4
     4     4     4     4
     4     4     4     4
```

```
>> D=4*ones(2)
D =
     4     4
     4     4
```

مفاهیم پایه

- M-File
- مجموعه ای از دستورات Matlab را می توان با M-File اجرا کرد. همچنین می توان توابعی تعریف کرد که با دریافت آرگومان، خروجی مورد نظر را تولید کند.
- معمولا در ابتدای M-File توضیحاتی نوشته می شود. این توضیحات با ٪ مشخص می شوند.

```
% this M-file computes ...
```

- با دستور help filename توضیحات نوشته شده در ابتدای برنامه در Command Window ظاهر می شوند(H1 line).
- با دستور edit filename می توان M-File را ویرایش کرد. برای ویرایش M-File می توان از هر ویرایشگر متنی استفاده کرد. کافی است فایل با پسوند .m ذخیره گردد.

مفاهیم پایه

❑ عملگرها (Operators)

عملگرهای Matlab به سه دسته تقسیم می شوند:

✓ عملگرهای محاسباتی (Arithmetic operators)

✓عملگرهای رابطه ای (Relational operators)

✓عملگرهای منطقی (Logical operators)

مفاهیم پایه

```
>> A=[1 2;3 4]
A =
     1     2
     3     4
>> B=[5 6;7 8]
B =
     5     6
     7     8
>> A*B
ans =
    19    22
    43    50
>> A.*B
ans =
     5    12
          21    32
```

```
>> inv(B)
ans =
   -4.0000    3.0000
    3.5000   -2.5000
>> A/B
ans =
    3.0000   -2.0000
    2.0000   -1.0000
 >> A*inv(B)
ans =
    3.0000   -2.0000
    2.0000   -1.0000
```

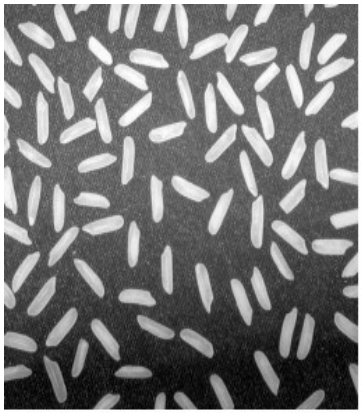
```
>> A./B
ans =
    0.2000    0.3333
    0.4286    0.5000

>> A'
ans =
     1     3
     2     4
```

مفاهیم پایه

در جعبه ابزار پردازش تصویر، دستوراتی برای انجام عملیات محاسباتی روی تصاویر وجود دارد. مانند:
imadd, imsubtract, immultiply, imdivide

مثال:

```
I= imread('rice.png');
J= imread('cameraman.tif');
K= imadd(I,J,'uint16');
figure(1);imshow(I,[])
figure(2);imshow(J,[])
figure(3);imshow(K,[])
```



مفاهیم پایه

عملگرهای رابطه ای

این عملگرها برای مقایسه مولفه های دو ماتریس می توانند استفاده شوند و عبارتند از:

> ، >= ، < ، =< ، == و ~=

```
>> A=[1  3 0; 5 2 7; 1 4 5]
A =
    1    3    0
    5    2    7
    1    4    5

>> B=[4  3 0; 7 2 1; 4 0 8]
B =
    4    3    0
    7    2    1
    4    0    8
```

```
>> A>B
ans =
    0    0    0
    0    0    1
    0    1    0

>> A<B
ans =
    1    0    0
    1    0    0
    1    0    1
```

```
>> A==B
ans =
    0    1    1
    0    1    0
    0    0    0

>> A~=B
ans =
    1    0    0
    1    0    1
    1    1    1
```

مفاهیم پایه

❑ عملگرهای منطقی

AND: با فرمت and(A,B) یا A&B

ماتریسی با ابعاد ورودی و مولفه های 1و 0 تولید می کند. اگرمقدار مولفه ها غیر صفر باشد در محل مولفه مقدار یک و در غیر این صورت صفر قرار می دهد.

OR: با فرمت or(A,B) یا A|B

ماتریسی با ابعاد ورودی و مولفه های 1و 0 تولید می کند. اگر حداقل یک مولفه غیر صفر باشد در محل مولفه مقدار یک و در غیر این صورت صفر قرار می دهد.

NOT: با فرمت not(A) یا ~A

ماتریسی با ابعاد ورودی تولید می کند که مولفه های صفرماتریس ورودی را با یک و مولفه های غیر صفررا با صفر پر می کند.

XOR : با فرمت xor(A,B)

اگر هر دو مولفه صفر یا غیر صفر باشند مقدار صفر و اگر یکی صفر و دیگری غیر صفر باشد مقدار یک بر میگرداند.

مفاهیم پایه

- عملگرهای منطقی

```
>> A=[1  3 0; 5 2 7; 1 4 5]
A =
    1    3    0
    5    2    7
    1    4    5
>> B=[4  3 0; 7 2 1; 4 0 8]
B =
    4    3    0
    7    2    1
    4    0    8
>> and(A,B)
ans =
    1    1    0
    1    1    1
    1    0    1
```

```
>> or(A,B)
ans =
    1    1    0
    1    1    1
    1    1    1

>> not(A)
ans =
    0    0    1
    0    0    0
    0    0    0

>> xor(A,B)
ans =
    0    0    0
    0    0    0
    0    1    0
```

مفاهیم پایه

- عملگرهای منطقی

تابع all: اگر تمام مولفه های بردار غیر صفر باشند مقدار یک و در غیر این صورت مقدار صفر بر می گرداند.

تابع any: اگر تمام مولفه های یک بردار صفر باشند مقدار صفر و در غیر این صورت یک بر می گرداند.

این دو تابع برای ماتریس به صورت ستونی عمل می کنند (بردارهای ستونی).

```
>> A=[0 1 2; 0 0 3]
A =
     0     1     2
     0     0     3
```

```
>> all(A)
ans =
     0     0     1

>> any(A)
ans =
     0     1     1
```

مفاهیم پایه

دستورات کنترل جریان (Flow Control statements)

دستور شرطی IF:

فرمت:

```
if expression1
   statements 1
elseif expression2
   statements 2
else
   statements 3
end
```

مثال:

```
if a==1
b=1
else if a==2
b=2
else
b=3
end
```

مفاهیم پایه

حلقه FOR:

```
for index=start:increment:end
   statements
end
```

مثال:

```
a=1;
for k=0:0.1:1
  a=a+1;
end
```

حلقه WHILE:

تا وقتی که شرط مقابل while برقرار است، حلقه ادامه می یابد.

```
while expression
   statements
end
```

مفاهیم پایه

دستور SWITCH

```
switch  switch_expression
   case case_expression1
       statement(s)
    case case_expression2
       statement(s)
              .
              .
     otherwise
      statement(s)
end
```

مثال:

```
count=1;
switch count
case count==1
fprintf('one')
case count==2
fprintf('two')
end
```

مفاهیم پایه

برخی ثابت ها:

```
pi=3.1415926535897....
eps=2.2204e-016
i = 0 + 1.0000i
```

دستورات tic و toc برای محاسبه زمان اجرای برنامه استفاده می شوند.

```
tic
for i=1:10              % Delay
end
toc

tic
for i=1:1000000         % Delay
end
toc
```

```
Elapsed time is 0.000000 seconds.
Elapsed time is 0.015000 seconds.
```

می توان نشان داد که زمان اجرای دستورات برداری (Vectorized Codes) حدوداً ۱۰۰۰ مرتبه سریعتر از اجرای حلقه for است.

مفاهیم پایه

ورود داده ها:

```
t=input('Enter your Data:')
```

دستور فوق ابتدا عبارت داخل ' ' را روی صفحه ظاهر کرده و منتظر ورودی می شود و آنرا وارد متغیر t می کند. ورودی می تواند یک عدد، یک بردار یا یک ماتریس باشد.

برای وارد کردن یک رشته کاراکتر می توان از فرمت زیر استفاده کرد:

```
t=input('Enter your Data:' , 's')
```

با دستور strread می توان رشته خوانده شده را به فرمتهای مختلف تبدیل کرد

مفاهیم پایه

مثال:

```
>> t=input('Enter your data: ','s')
Enter your data: 8250361 Ali Ahmadi
t =
8250361 Ali Ahmadi
>> [student_ID First_Name Last_Name]=strread(t,'%f%q%q')
student_ID =
     8250361
First_Name =
    'Ali'
Last_Name =
    'Ahmadi'
```

مفاهیم پایه

فرمت های قابل استفاده در دستور strread

%n - read a number - float or integer (returns double array)
%5n reads up to 5 digits or until next delimiter
%d - read a signed integer value (returns double array)
%5d reads up to 5 digits or until next delimiter
%u - read an integer value (returns double array)
%5u reads up to 5 digits or until next delimiter
%f - read a floating point value (returns double array)
%5f reads up to 5 digits or until next delimiter
%s - read a whitespace separated string (returns cellstr)
%5s reads up to 5 characters or until whitespace
%q - read a (possibly double quoted) string (returns cellstr)
%5q reads up to 5 non-quote characters or until whitespace
%c - read character or whitespace (returns char array)
%5c reads up to 5 characters including whitespace


دیگران را بخشید
به خاطر اینکه لایق بودند
به خاطر اینکه لایق بودند