

## فصل اول

# آشنایی

هدف: بررسی چند مثال کاربردی در متلب

## مثال ۱ - خواندن و ذخیره تصویر

### ۱- خواندن تصویر

بهتر است ابتدا متغیرهای فضای کاری را پاک کنید:

```
clear all;
```

و همچنین پنجره هایی که تاکنون باز مانده اند را نیز ببندید:

```
close all;
```

حال تصویر مورد نظر را به کمک دستور `imread` خوانده و در آرایه‌ای به نام `I` قرار می‌دهیم:

```
I = imread('pout.tif');
```

برای دیدن لیستی از فایل‌های مورد پشتیبانی در MATLAB (از جمله فایل‌های تصویری) دستور زیر را وارد کنید:

```
>>help fileformats
```

حال می‌توانید تصویر خوانده شده را نمایش دهید. برای این کار دو تابع متداول است یکی `imshow` و دیگری `imtool` که البته غالباً از اولی استفاده میشود زیرا فقط کار نمایش را انجام میدهد اما دومی علاوه بر نمایش اطلاعات بیشتری در اختیار قرار داده و حتی می‌توانیم به کمک آن برخی پردازشها را نیز انجام دهیم.

استفاده از دستور `imshow`:

```
imshow (I)
```

نتیجه:



**Grayscale Image pout.tif**

### ۲- کسب اطلاعات در مورد نحوه تغییر فضای کاری

به پنجره فضای کاری (`workspace`) خود نگاه کنید و ببینید که در اثر اجرای دستور خواندن تصویر، چه تغییری به فضای کاری اضافه شده است. البته می‌توانید از دستور `whos` نیز در پنجره فرمان استفاده کنید:

```
>>whos
```

نتیجه:

Name	Size	Bytes	Class	Attributes
I	291x240	69840	uint8	

در MATLAB، تصاویر به صورت (یا نوع) uint8، uint16، و double قابل ذخیره می باشند (توضیح هر کدام؟).

### ۳- بهبود تبیان (یا کانتراست)<sup>1</sup> تصویر

#### تعریف کانتراست؟

کانتراست تصویر مورد بررسی در این مثال کم (یا پایین) است. برای بررسی صحت این جمله به صورت علمی باید هیستوگرام تصویر مذکور را بررسی کرد.

#### تعریف هیستوگرام؟

برای نمایش نحوه توزیع شدت روشنایی (یا همان هیستوگرام) یک تصویر میتوان از دستور imhist استفاده کرد. ابتدا از دستور figure برای ایجاد یک پنجره جدید استفاده میکنیم (چرا؟).

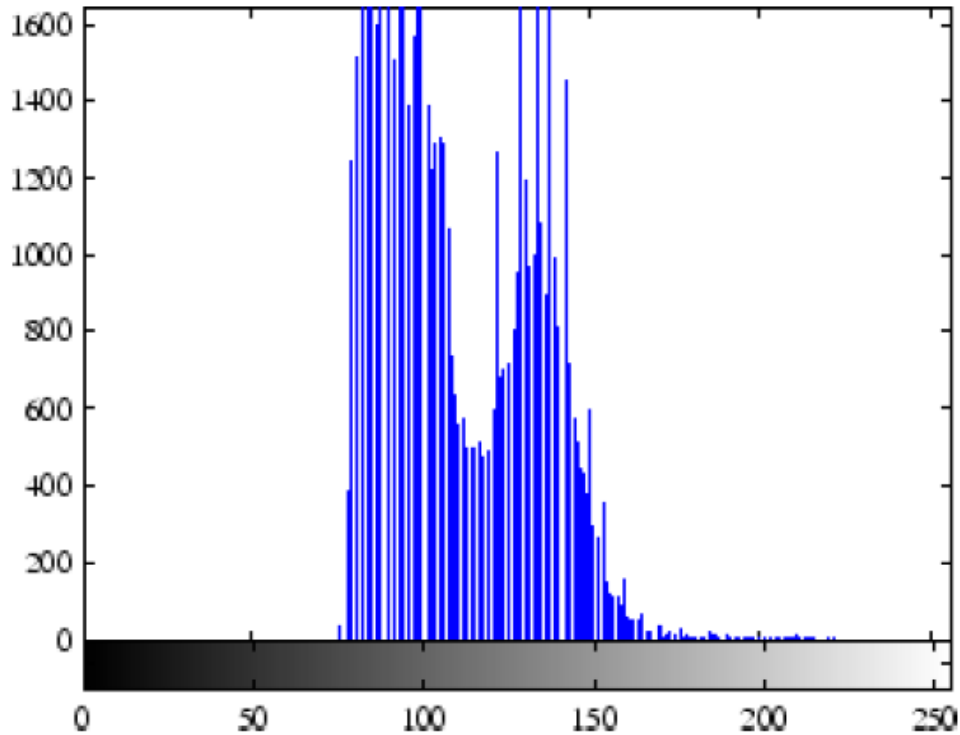
```
>>figure
```

حال دستور imhist را استفاده میکنیم:

```
>>imhist(I)
```

نتیجه :

<sup>1</sup> Contrast



حال، به کمک شکل فوق، محدوده تغییرات شدت روشنایی (یا محدوده دینامیکی<sup>۲</sup>) را ارزیابی کنید و برای خود استدلال کنید که چرا کانتراست تصویر I پایین است.

چرا به دنبال افزایش کانتراست باید بود؟

برای افزایش کانتراست تصویر در MATLAB راههای مختلفی وجود دارد از جمله استفاده از دستور `histeq` که از فرآیند تعدیل‌سازی هیستوگرام<sup>۳</sup> استفاده می‌کند. نتیجه اجرای این فرآیند را در تصویر دیگری مانند I2 ذخیره می‌کنیم:

```
I2 = histeq(I);
```

حال نتیجه را در پنجره جدیدی نمایش می‌دهیم:

```
figure, imshow(I2)
```

<sup>2</sup> Dynamic Range

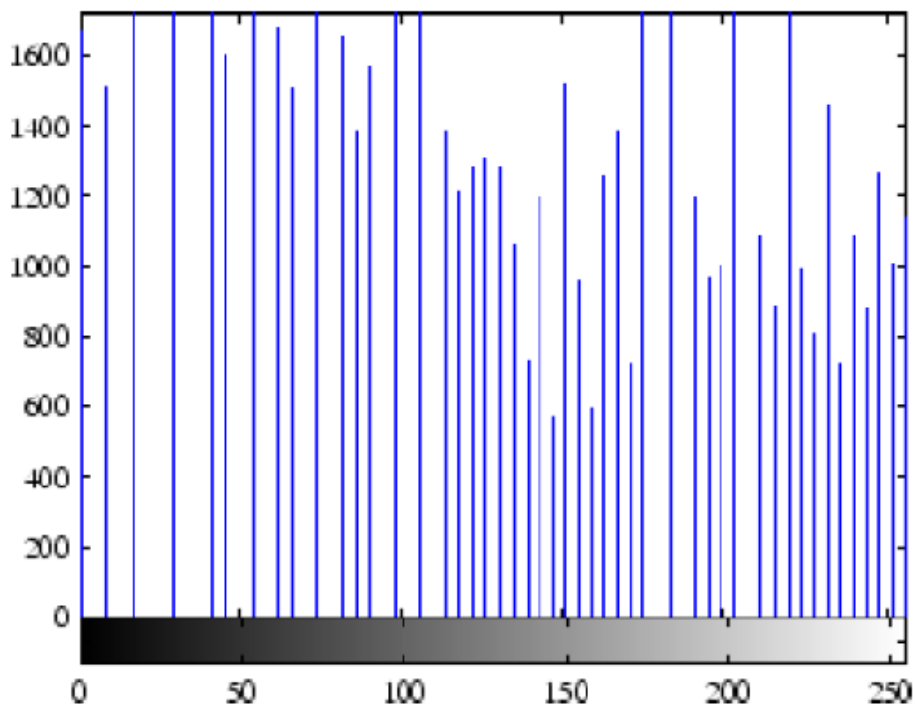
<sup>3</sup> Histogram Equalization



### Equalized Version of pout.tif

حال برای اینکه اثر دستور `histeq` را روی هیستوگرام تصویر `I` ببینید، مجدداً از دستور `imhist` استفاده کنید:

`figure, imhist(I2)`



(روی تصویر فوق فکر و استدلال کنید)

برخی دیگر امکانات موجود در متلب برای افزایش کانتراست یک تصویر عبارتند از:

۱- دستور `imadjust`

۲- دستور `adapthisteq`

۳- استفاده از یک ابزار تعاملی<sup>۴</sup> به کمک استفاده از دستور `imcontrast`

تمرین: در مورد هر یک از قابلیت‌های فوق به طور عملی تحقیق کنید. یعنی برنامه‌ای برای کار با هر یک از دو دستور اول نوشته و اثر آنها را بررسی کنید. همچنین نحوه‌ی استفاده از ابزار مذکور (شماره‌ی ۳) را بررسی کنید.

#### ۴- ذخیره‌ی یک تصویر در حافظه‌ی کامپیوتر

از دستور `imwrite` برای ذخیره‌ی اطلاعات تصویری موجود در یک آرایه به صورت یک فایل تصویری در حافظه‌ی کامپیوتر استفاده می‌شود. شما می‌توانید از هر یک از قالب‌های تصویری مورد پشتیبانی متلب استفاده کنید. مثلاً:

```
imwrite(I2, 'pout2.png');
```

#### ۵- محتویات فایل ذخیره شده را بررسی کنید

برای اینکه ببینید فایل ذخیره شده در مرحله‌ی قبل چه ویژگی‌هایی دارد، می‌توانید از دستور `imfinfo` استفاده کنید. این دستور اطلاعات مختلفی در مورد تصویر مورد نظرتان ارائه می‌دهد. برای استفاده، به صورت زیر عمل کنید:

```
imfinfo('pout2.png')
```

نتیجه:

<sup>4</sup> Interactive tool

```
ans =
```

```
    Filename: 'pout2.png'  
    FileModDate: '29-Dec-2005 09:34:39'  
    FileSize: 36938  
    Format: 'png'  
    FormatVersion: []  
    Width: 240  
    Height: 291  
    BitDepth: 8  
    ColorType: 'grayscale'  
    FormatSignature: [137 80 78 71 13 10 26 10]  
    Colormap: []  
    Histogram: []  
    InterlaceType: 'none'  
    Transparency: 'none'  
    SimpleTransparencyData: []  
    BackgroundColor: []  
    RenderingIntent: []  
    Chromaticities: []  
    Gamma: []  
    XResolution: []  
    YResolution: []  
    ResolutionUnit: []  
    XOffset: []  
    YOffset: []  
    OffsetUnit: []  
    SignificantBits: []  
    ImageModTime: '29 Dec 2005 14:34:39 +0000'  
    Title: []  
  
    Author: []  
    Description: []  
    Copyright: []  
    CreationTime: []  
    Software: []  
    Disclaimer: []  
    Warning: []  
    Source: []  
    Comment: []  
    OtherText: []
```

## مثال ۲- تحلیل تصویر

### ۱- خواندن تصویر

تعریف تصویر سطح خاکستری؟

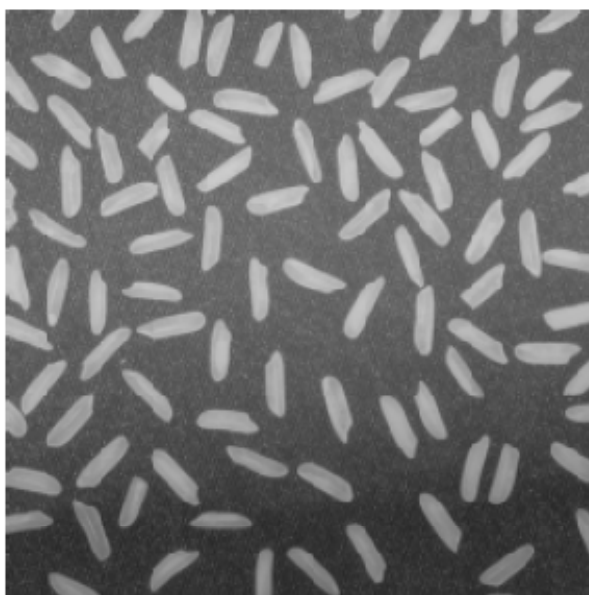
تصویر سطح خاکستری rice.png را خوانده و در یک آرایه به نام I ذخیره کنید:

```
I = imread('rice.png');
```

حال آن را نمایش دهید:

```
imshow(I)
```

نتیجه:



**Grayscale Image rice.png**

تعریف پس‌زمینه<sup>۵</sup>؟

تعریف پیش‌زمینه<sup>۶</sup>؟

ملاحظه میکنید که پس‌زمینه تصویر یکنواخت نیست بلکه در قسمتهای مرکزی روشنتر و در قسمتهای پایینی تیره‌تر است. عدم یکنواختی پس‌زمینه، اغلب باعث دشوارتر شدن کار جداکردن اشیاء از تصویر می‌شود.

### ۲- تخمین پس‌زمینه

<sup>5</sup> Background

<sup>6</sup> Foreground



از عملگر مورفولوژی بازکردن<sup>۷</sup> (شامل اعمال متوالی کاهش<sup>۸</sup> و سپس گسترش<sup>۹</sup> هر دو به کمک یک ماسک مشترک) برای حذف دانه‌های برنج استفاده می‌کنیم تا بتوانیم به پس‌زمینه دست پیدا کنیم. اثر نهایی عملگر بازکردن این است که اشیایی را که نمی‌توانند ماسک را در بر بگیرند، حذف میکند (اطلاعات بیشتر در مورد عملگرهای مورفولوژی در فصل ۱۰ آمده است).

برای استفاده از عملگر بازکردن از دستور `imopen` به صورت زیر استفاده می‌کنیم:

```
background = imopen(I, strel('disk', 15));
```

آرگومان اول، تصویر مورد نظرمان و آرگومان دوم ماسک مورد استفاده برای انجام عمل بازکردن است. در اینجا برای تشکیل ماسک از دستور `strel` استفاده شده است. دستور مذکور در این مثال برای ایجاد یک ماسک دایره‌ای به شعاع ۱۵ پیکسل مورد استفاده قرار گرفته است. اندازه‌ی ماسک را به گونه‌ای انتخاب کرده‌ایم که هیچ دانه‌ی برنجی نتواند ماسک را به طور کامل در بر بگیرد و در نتیجه از تصویر حذف شود. اگر تقریب فوق را به کمک دستور `imshow` نمایش دهیم نتیجه چنین خواهد بود:



<sup>7</sup> Opening

<sup>8</sup> Erosion

<sup>9</sup> Dilation

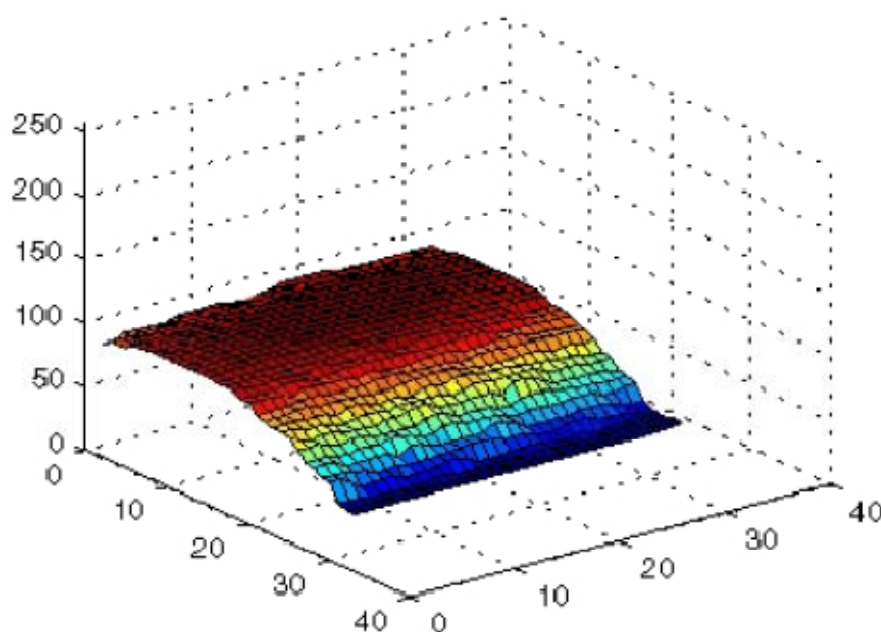
### ۳- مشاهده (و نمایش) تقریب پس‌زمینه به صورت یک سطح

برای نمایش پس‌زمینه به صورت یک سطح از دستور surf استفاده میکنیم؛ اما این دستور نیاز به آرگومان ورودی از نوع double دارد در حالیکه تصویر موجود در آرایه‌ی background از نوع uint8 است. بنابراین، در حین استفاده از دستور surf کار تبدیل نوع را هم انجام می‌دهیم:

```
figure, surf(double(background(1:8:end,1:8:end))),zlim([0 255]);
set(gca,'ydir','reverse');
```

(بررسی نکات موجود در دستور فوق)

نتیجه :



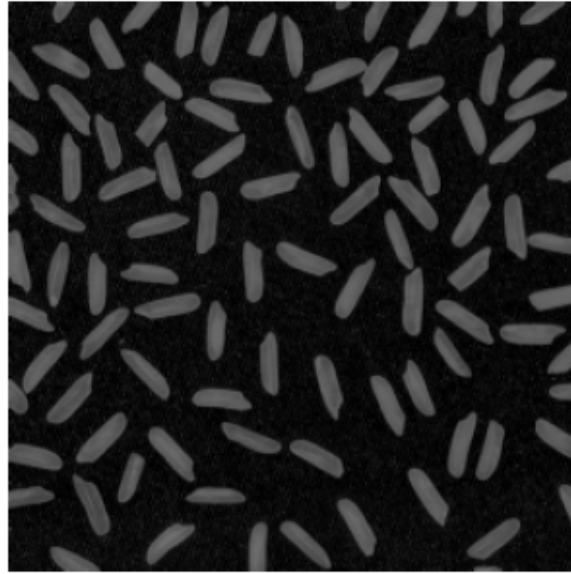
## Surface Plot

### ۴- تفریق پس‌زمینه از تصویر اصلی

برای ایجاد یک تصویر با پس‌زمینه یکنواخت، می‌توان تقریب بدست آمده از پس‌زمینه را از تصویر اصلی کم کرد و حاصل را نمایش داد:

```
I2 = I - background;
figure, imshow(I2)
```

نتیجه:



**Image with Uniform Background**

#### ۵- افزایش کانتراست تصویر

تصویری که تا بحال بدست آوردیم پس زمینه تقریباً یکنواختی دارد اما تا حدی تاریک است و باید کانتراست آن را افزایش دهیم. در اینجا میخواهیم از دستور `imadjust` برای انجام این کار استفاده کنیم. این دستور یک درصد از داده‌های بالا و پایین را اشباع می‌کند و بقیه‌ی مقادیر را طوری تغییر می‌دهد که از تمام بازه‌ی تغییرات ممکن (در اینجا یعنی از ۰ تا ۲۵۵ که مربوط به نوع داده‌ی `uint8` است) استفاده شود.

```
I3 = imadjust(I2);  
figure, imshow(I3);
```

نتیجه:



**Image After Intensity Adjustment**

## ۶- آستانه‌گیری تصویر

تعریف آستانه‌گیری<sup>۱۰</sup>؟

تعریف تصویر دودویی (یا باینری<sup>۱۱</sup>)؟

نحوه‌ی دودویی کردن، تعریف آستانه، اهمیت مقدار آستانه؟

برخی مزایای استفاده از تصاویر باینری (چرا به دنبال دودویی کردن باید باشیم؟):

الف - حجم پایین پردازشها (سرعت اجرای بالا)

ب- بدست آوردن اطلاعات مفید از تصویر (مانند تعداد اشیاء، مساحت هر کدام، موقعیت هر شیء)

برای تبدیل یک تصویر سطح خاکستری به تصویر باینری از دستور `im2bw` میتوان استفاده کرد. این دستور نیاز به مقدار یک آستانه دارد. برای انتخاب مناسب مقدار آستانه، میتوانیم از دستور `graythresh` استفاده کنیم:

```
level = graythresh(I3);
bw = im2bw(I3,level);
```

برای حذف نویز، میتوانیم از دستور `bwareaopen` استفاده کنیم. ایده‌ی اساسی این است که نویز عموماً شامل نقاط با مساحت کم است. بنابراین میتوان هر شیء با مساحتی کمتر از یک حد را نویز دانست. بنابراین به کمک عملگر مورفولوژی باز کردن میتوان تمام اشیایی که مساحتشان کمتر از یک مقدار مشخص (در اینجا ۵۰) باشد را حذف کرد.

**توجه:** مقدار ۵۰ در حقیقت تابعی از رزولوشن<sup>۱۲</sup> (یا درجه تفکیک) تصویر است. هر چه رزولوشن بیشتر باشد، باید مقدار این آستانه را نیز بزرگتر در نظر گرفت.

تعریف رزولوشن (در مکان و در مقدار)؟

تعریف dpi؟

```
bw = bwareaopen(bw, 50);
figure, imshow(bw)
```

نتیجه:

<sup>10</sup> Thresholding

<sup>11</sup> Binary

<sup>12</sup> Resolution



**Binary Version of the Image**

### ۷- شناسایی اشیاء در تصویر

تعریف جزء به هم پیوسته<sup>۱۳</sup>؟

برای استخراج اجزاء به هم پیوسته از یک تصویر باینری میتوان از هر یک از دستورات `bwconncomp`، `bwlabel` و `bwlabeln` استفاده کرد اما بهترین و جدیدترین آنها دستور `bwconncomp` است که حافظه کمتری مصرف کرده و گاهی سریعتر نیز می‌باشد.

The accuracy of your results depends on the size of the objects, the connectivity parameter (4, 8, or arbitrary), and whether or not any objects are touching (in which case they could be labeled as one object). Some of the rice grains in `bw` are touching.

مقایسه مختصر سه دستور فوق:

	Input Dim	Output Form	Memory Use	Connectivity
BWLABEL	2-D	Double-precision label matrix	High	4 or 8
BWLABELN	N-D	Double-precision label matrix	High	Any

<sup>13</sup> Connected Component

BWCONNCOMP N-D CC struct Low Any

(تمرین: جزئیات دستور اخیر را پیدا کرده و آنها را بررسی کنید.)

در پنجره فرمان دستورات زیر را وارد کنید:

```
cc = bwconncomp(bw, 4)
cc.NumObjects
```

نتیجه:

```
cc =

Connectivity: 4
ImageSize: [256 256]
NumObjects: 95
PixelIdxList: {1x95 cell}
```

```
ans =
```

```
95
```

تمرین ۱: پیدا کردن و پاک کردن بزرگترین شیء در یک تصویر باینری

```
CC = bwconncomp(BW);
numPixels = cellfun(@numel,CC.PixelIdxList);
[biggest,idx] = max(numPixels);
BW(CC.PixelIdxList{idx}) = 0;
```

توضیحات دستورات cellfun و numel

تمرین ۲: معرفی و استفاده از دستور regionprops

برای کسب برخی اطلاعات از یک تصویر باینری، ابتدا اطلاعات اجزاء به هم پیوسته‌ی آن را به کمک یکی از سه دستور مذکور (مانند bwconncomp) استخراج کرده و به دستور regionprops بدهید. این دستور برخی ویژگیها را محاسبه می‌کند که عبارتند از:

```
>>S = regionprops(CC,'all');
```

```
S =
```

88x1 struct array with fields:

```
Area
Centroid
BoundingBox
SubarrayIdx
MajorAxisLength
MinorAxisLength
Eccentricity
Orientation
ConvexHull
```

ConvexImage  
 ConvexArea  
 Image  
 FilledImage  
 FilledArea  
 EulerNumber  
 Extrema  
 EquivDiameter  
 Solidity  
 Extent  
 PixelIdxList  
 PixelList  
 Perimeter

اگر میخواهید فقط برخی از ویژگیهای فوق محاسبه شوند (تا در زمان صرفه جویی کنید)، این ویژگیها را پشت سر هم لیست کنید. مانند:

```
>> S = regionprops(CC,'Centroid','Area');
```

تمرین ۳: استفاده از دستورات `bwlabel` و `bwlabeln` و نیز دستور `imtool`

```
BW = imread('text.png');
L = bwlabel(BW,4); % or use L = bwlabeln(BW,4);
[r,c] = find(L == 2)
imtool(L)
```

نکته ۱: شیء با شماره‌ی صفر همان پس‌زمینه است.

نکته ۲: خروجی دستور `bwconncomp` از نوع سلول است (نه یک ماتریس مانند `L` در مثال فوق). برای اینکه در این دستور هم بتوانیم یک خروجی ماتریسی مانند ماتریس `L` در مثال اخیر داشته باشیم از دستور `labelmatrix` میتوانیم استفاده کنیم:

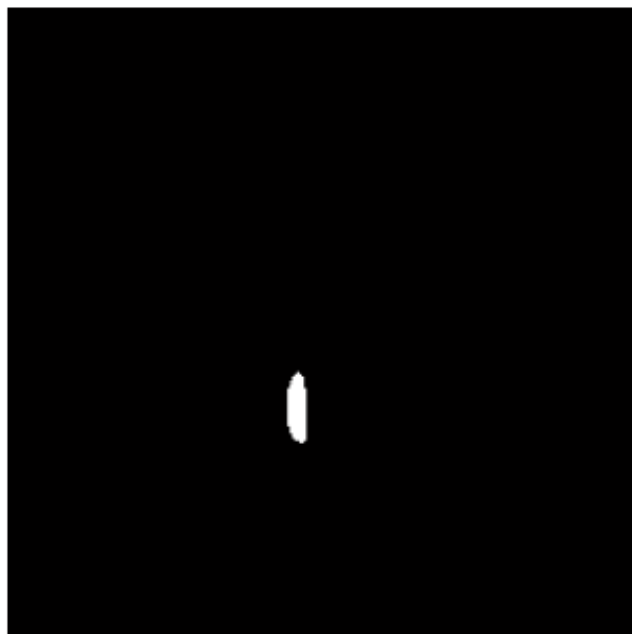
```
BW = imread('text.png');
CC = bwconncomp(BW);
L = labelmatrix(CC);
L2 = bwlabel(BW);
whos L L2
```

#### ۸- بررسی یک شیء

نشان دادن دانه برنج با شماره‌ی ۵۰ :

```
grain = false(size(bw));
grain(cc.PixelIdxList{50}) = true;
figure, imshow(grain);
```

نتیجه:



## The 50th Connected Component

۹- مشاهده‌ی تمام اشیاء

ابتدا از دستور `labelmatrix` برای تهیه ماتریس برچسب استفاده میکنیم:

```
labeled = labelmatrix(cc);
```

```
whos labeled
```

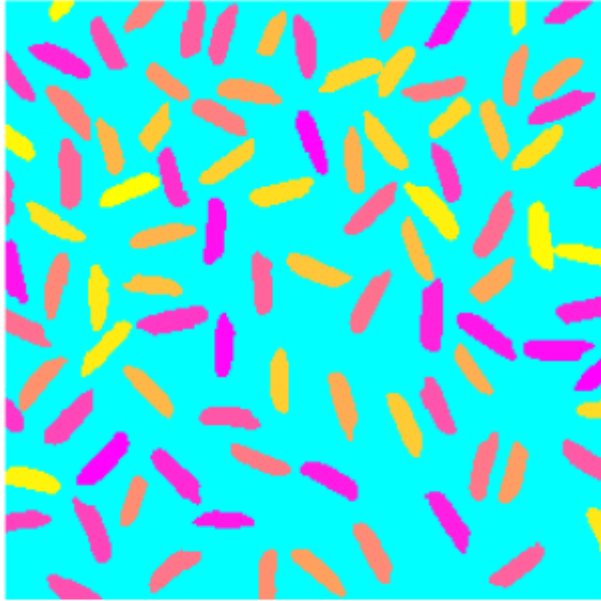
Name	Size	Bytes	Class	Attributes
labeled	256x256	65536	uint8	

حال برای اینکه هر دانه برنج را با یک رنگ نشان دهیم تا بتوانیم تصویری رنگی مشاهده کنیم، ماتریس به دست آمده را به کمک دستور `label2rgb` به یک تصویر رنگی تبدیل کرده و آن را نمایش می‌دهیم:

```
RGB_label = label2rgb(labeled, @spring, 'c', 'shuffle');  
figure, imshow(RGB_label)
```

نتیجه:





## Label Matrix Displayed as Pseudocolor Image

تمرین: از دستور `label2rgb` راهنما بگیرید و در مورد نحوه استفاده از رنگ‌آمیزیهای مختلف تحقیق کنید. در اینجا ممکن است به دستور `help graph3d` نیاز داشته باشید.

### ۱۰- محاسبه مساحت اشیاء مختلف

هر دانه برنج یک جزء به هم پیوسته است. به کمک دستور `regionprops` اطلاعاتی را میتوان در مورد هر دانه برنج کسب کرد:

```
graindata = regionprops(cc, 'basic')
```

نتیجه:

```
graindata =
```

```
95x1 struct array with fields:
```

```
Area
```

```
Centroid
```

```
BoundingBox
```

برای دسترسی به مساحت برنج ۵۰ ام:

```
graindata(50).Area
```

```
ans =
```

```
194
```

## ۱۱- محاسبه‌ی برخی ویژگیها از روی مساحت

ابتدا اطلاعات مساحت دانه‌های برنج را در یک آرایه جداگانه ذخیره کرده و سپس، کوچکترین دانه برنج را پیدا کنید:

```
grain_areas = [graindata.Area];
[min_area, idx] = min(grain_areas)
grain = false(size(bw));
grain(cc.PixelIdxList{idx}) = true;
figure, imshow(grain);
```

نتیجه (مقدار مساحت و شماره‌ی کوچکترین دانه برنج):

```
min_area =
```

```
61
```

```
idx =
```

```
16
```



**Smallest Grain**

## ۱۲- تولید هیستوگرام برای مساحت‌های دانه‌های برنج

چه اطلاعاتی از هیستوگرام میتوان به دست آورد؟

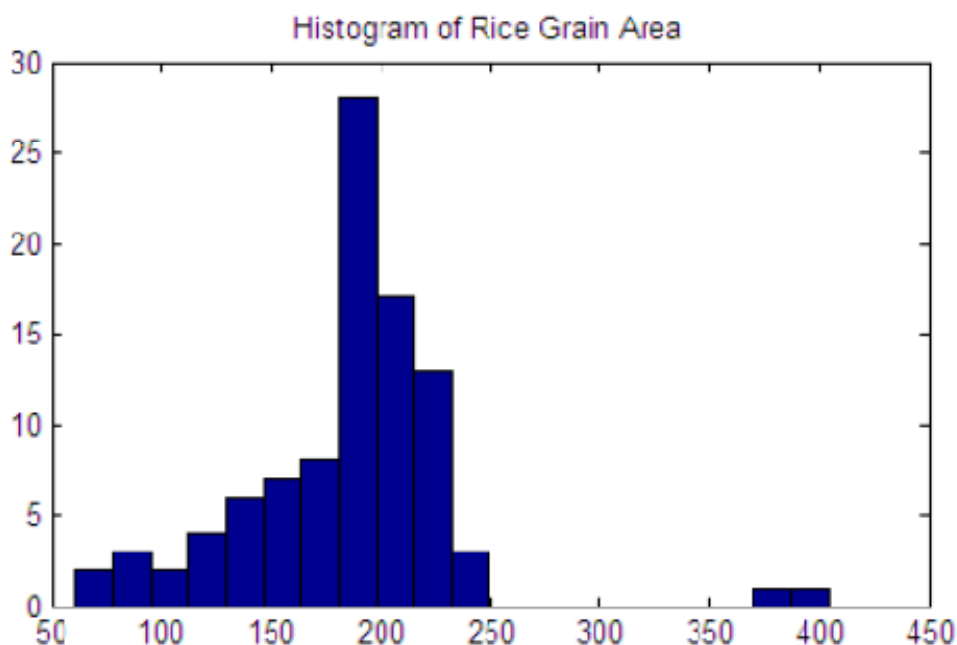
الف- عمده دانه-برنجها چه مساحتی دارند؟

ب- چقدر دانه-برنجها یکدست هستند؟

از دستور hist برای تولید هیستوگرام برداری از داده‌ها استفاده می‌کنیم. میتوانیم تعیین کنیم که محدوده تغییرات داده‌ها در بردار ورودی به چند بازه تقسیم شود.

```
nbins = 20;  
figure, hist(grain_areas, nbins)  
title('Histogram of Rice Grain Area');
```

نتیجه:



**توجه:** برای دیدن نمایشی از قابلیت‌های پردازش تصویر در متلب دستور زیر را وارد کنید تا صفحه مربوطه برای شما نمایش داده شود.

```
>> iptdemos
```

## فصل دوم

# مقدمه

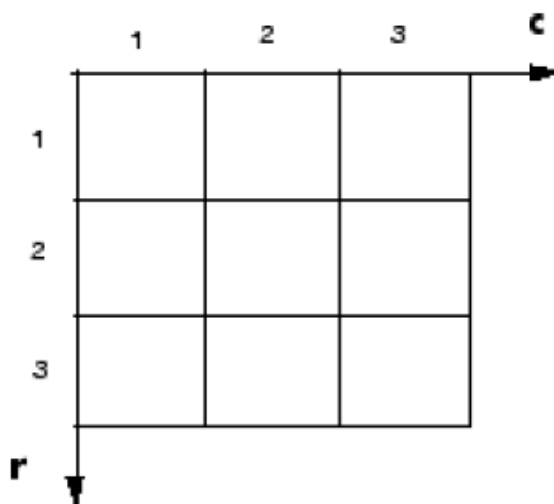
هدف: آشنایی با اصول اولیه و محاسبات ریاضی تصاویر در متلب

در متلب اغلب تصاویر به صورت یک آرایه‌ی دو بعدی ذخیره میشوند که به آنها ماتریس می‌گوییم. هر عنصر از ماتریس متناظر با یک پیکسل<sup>۱۴</sup> از تصویر مربوطه است. برای نمایش برخی تصاویر مانند تصاویر رنگی، نیاز به آرایه سه بعدی داریم که در آن، اولین صفحه نمایشگر شدت مولفه قرمز، دومین صفحه نمایشگر شدت مولفه سبز، و سومین صفحه نمایشگر شدت مولفه آبی می‌باشد (به اصطلاح تصویر RGB هم گفته میشوند).

## سیستم‌های مختصات دهی تصاویر

### ۱- سیستم مختصات دهی پیکسلی

متداولترین روش برای مختصات دهی است. در این روش، تصویر به عنوان شبکه‌ای از عناصر گسسته در نظر گرفته میشود که مطابق شکل زیر، به ترتیب از بالا به پایین، و از چپ به راست چیده شده‌اند.



### The Pixel Coordinate System

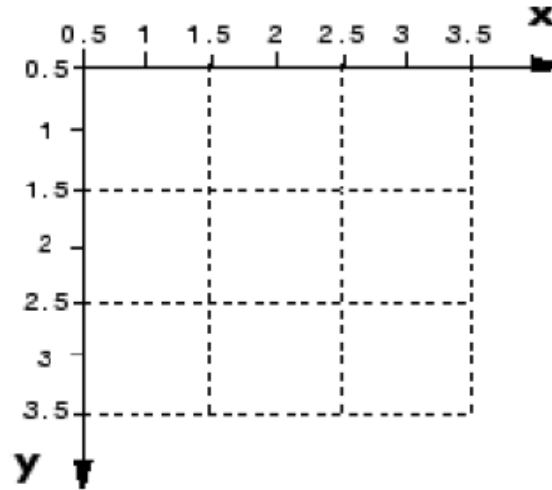
اولین مولفه،  $r$  (سطر) بوده و به سمت پایین افزایش می‌یابد. دومین مولفه  $c$  (ستون) بوده و به سمت راست افزایش می‌یابد. مختصات پیکسلها اعداد صحیحی هستند که بین ۱ تا ماکزیم تعداد سطرها/ستونها تغییر می‌کنند. بین مختصات سطر و ستونی پیکسلها و مختصات نقاط متناظر در ماتریس تصویر تناظر یک به یک وجود دارد. برای مثال عضوی از ماتریس  $I$  به مختصات  $I(2,15)$  همان پیکسل واقع در سطر دوم و ستون پانزدهم است.

### ۲- مختصات مکانی

در این روش به جای اینکه پیکسل را به عنوان یک نقطه گسسته در نظر بگیریم، آن را به عنوان یک مربع کوچک در نظر می‌گیریم. بنابراین، در این روش مختصات  $(2,2)$  معنا دار خواهد بود حال آنکه در

<sup>14</sup> Pixel : Picture Element

مختصات پیکسلی فاقد معنا است. در روش مختصات مکانی، نقاط مختلف تصویر در واقع مکانهایی در صفحه  $x-y$  بوده و با همین اعداد (یعنی  $x$  و  $y$ ) توصیف میشوند نه  $c$  و  $r$ . شکل زیر این مطلب را نشان می دهد.



**The Spatial Coordinate System**

بین دو سیستم مختصات دهی فوق رابطه‌ای نیز وجود دارد و آن این است که مختصات نقاط واقع بر مقادیر صحیح  $x$  و  $y$  - یعنی مختصات مراکز پیکسلها - (در مختصات دهی مکانی) همان مختصات پیکسل واقع بر سطر  $r$  و ستون  $c$  (با  $r=y$  و  $c=x$ ) می باشد. یکی از موارد اختلاف بین دو سیستم فوق این است که در مختصات دهی پیکسلی مختصات اولین نقطه برابر  $(1,1)$  است اما در مختصات دهی مکانی این مختصات برابر  $(0,0)$  است زیرا سیستم مختصات دهی پیکسلی، گسسته و سیستم مختصات دهی مکانی، پیوسته است. دومین اختلاف مهم بین دو سیستم فوق ترتیب نوشتن مختصات است. در مختصات دهی پیکسلی به ترتیب  $(r,c)$  اما در مختصات دهی مکانی به ترتیب  $(x,y)$  عمل می شود. در صفحات مرجع و راهنما هر گاه از نمادهای  $r$  و  $c$  برای مختصات پیکسلها استفاده شده باشد به معنای سیستم مختصات دهی پیکسلی و هر گاه از  $x$  و  $y$  استفاده شود به معنای سیستم مختصات دهی مکانی است.

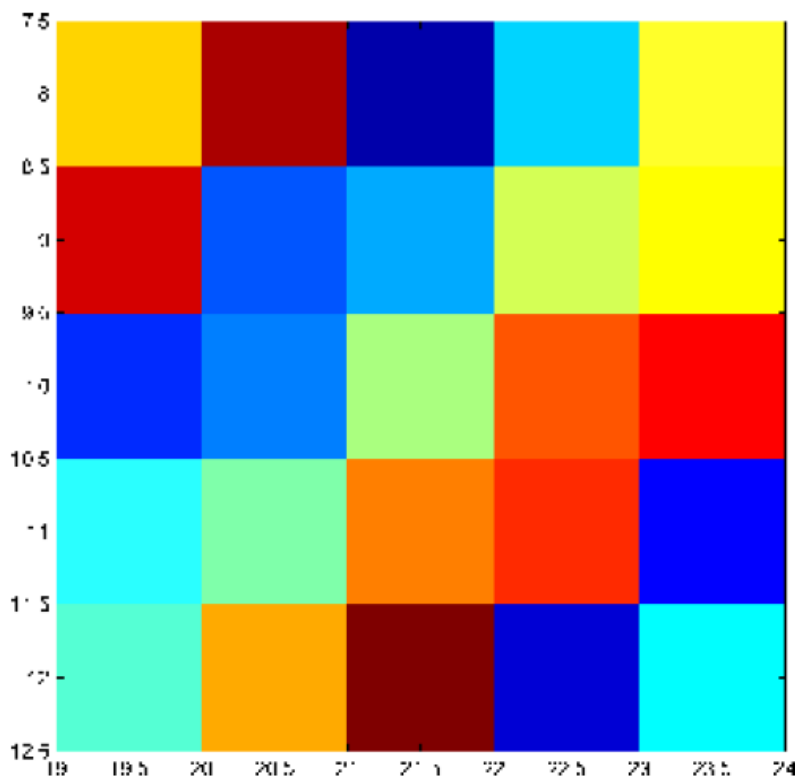
### ۳- تغییر محدوده مختصات مکانی

محدوده تغییرات مختصات مراکز پیکسلها در ویژگیهای  $XData$  و  $YData$  قرار دارد. اگر  $A$  تصویری دارای  $100$  سطر و  $200$  ستون باشد، مقدار پیش فرض  $YData$  برابر  $[1,100]$  و مقدار پیش فرض  $XData$  برابر  $[1,200]$  است بنابراین، مختصات  $x$  نقاط این تصویر در سیستم مختصات دهی مکانی، در حالت پیش فرض، در محدوده  $[0.5,200.5]$  و مختصات  $y$  نیز در محدوده  $[0.5,100.5]$  تغییر میکنند. برای تغییر محدوده مختصات در سیستم مختصات دهی مکانی میتوانید ویژگیهای  $XData$  و  $YData$  را در حین نمایش تصویر تغییر دهید. برای مثال:

```
A = magic(5);
x = [19.5 23.5];
```

```
y = [8.0 12.0];
image(A,'XData',x,'YData',y), axis image, colormap(jet(25))
```

نتیجه:



### انواع داده‌ای تصاویر

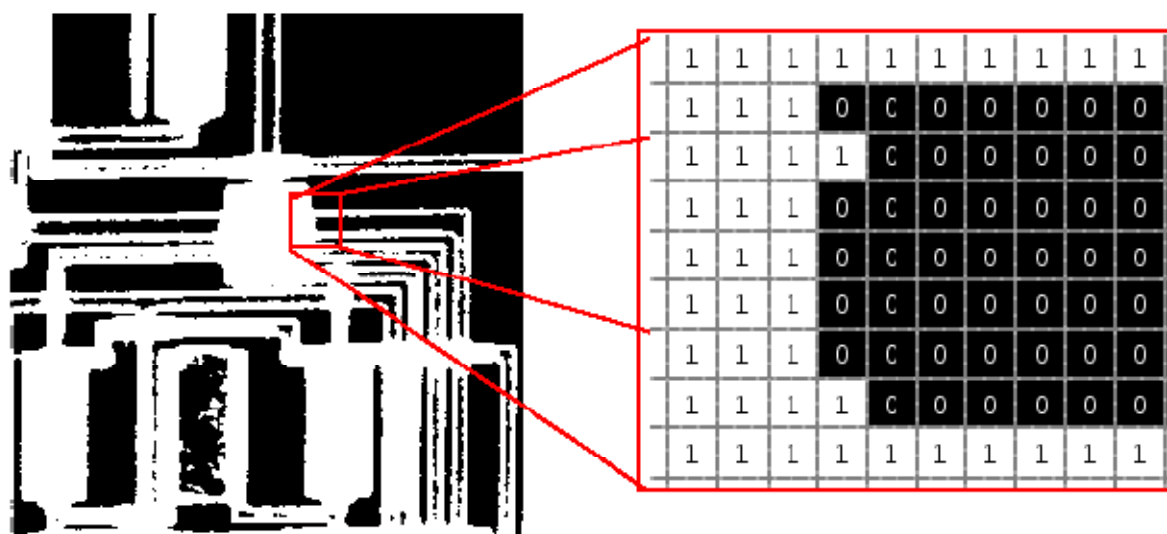
منظور از انواع داده‌ای تصاویر، نحوه‌ی برخورد و تفسیر متلب با داده‌های یک تصویر است. چهار نوع اساسی در متلب وجود دارد:

Image Type	Interpretation
Binary (Also known as a <i>bilevel image</i> )	Logical array containing only 0s and 1s, interpreted as black and white, respectively. See “Binary Images” on page 2-8 for more information.
Indexed (Also known as a <i>pseudocolor image</i> )	<p>Array of class <code>logical</code>, <code>uint8</code>, <code>uint16</code>, <code>single</code>, or <code>double</code> whose pixel values are direct indices into a colormap. The colormap is an <math>m</math>-by-3 array of class <code>double</code>.</p> <p>For <code>single</code> or <code>double</code> arrays, integer values range from <math>[1, p]</math>. For <code>logical</code>, <code>uint8</code>, or <code>uint16</code> arrays, values range from <math>[0, p-1]</math>. See “Indexed Images” on page 2-9 for more information.</p>

Grayscale (Also known as an <i>intensity</i> , <i>gray scale</i> , or <i>gray level</i> image)	Array of class uint8, uint16, int16, single, or double whose pixel values specify intensity values.  For single or double arrays, values range from [0, 1]. For uint8, values range from [0,255]. For uint16, values range from [0, 65535]. For int16, values range from [-32768, 32767]. See “Grayscale Images” on page 2-11 for more information.
Truecolor (Also known as an <i>RGB</i> image )	<i>m</i> -by- <i>n</i> -by-3 array of class uint8, uint16, single, or double whose pixel values specify intensity values.  For single or double arrays, values range from [0, 1]. For uint8, values range from [0, 255]. For uint16, values range from [0, 65535]. See “Truecolor Images” on page 2-12 for more information.

### ۱- تصاویر باینری<sup>۱۵</sup> (یا دودویی)

یک تصویر باینری به صورت یک آرایه منطقی (یا logical) ذخیره می‌شود. معمولاً در این کتابچه از نماد bw برای اشاره به تصاویر باینری استفاده می‌شود.



**Pixel Values in a Binary Image**

### ۲- تصاویر اندیس گذاری شده<sup>۱۶</sup>

در این روش از یک آرایه و یک ماتریس رنگ (به نام ماتریس نقشه رنگ<sup>۱۷</sup>) استفاده می‌شود. ابعاد آرایه برابر ابعاد تصویر اصلی است. مقادیر هر عنصر از آرایه در حقیقت اشاره‌گری به درون ماتریس رنگ است تا رنگ

<sup>15</sup> Binary

<sup>16</sup> Indexed



(در حالت تصاویر رنگی) و یا شدت روشنایی (در حالت تصاویر سطح خاکستری) پیکسل مربوطه را تعیین کند. ابعاد ماتریس رنگ برابر  $m \times 3$  و داده‌های داخل آن از نوع double (یعنی اعشاری) و بین [0,1] می‌باشند. هر سطر این ماتریس بیانگر ترکیبی خاص از رنگهای قرمز، سبز، و آبی بوده و در واقع، ضرایب این ترکیب محسوب میشوند.

در این کتابچه به طور پیش فرض معمولاً از نماد X برای آرایه‌ی مذکور و از نماد map برای ماتریس رنگ مذکور استفاده می‌شود.

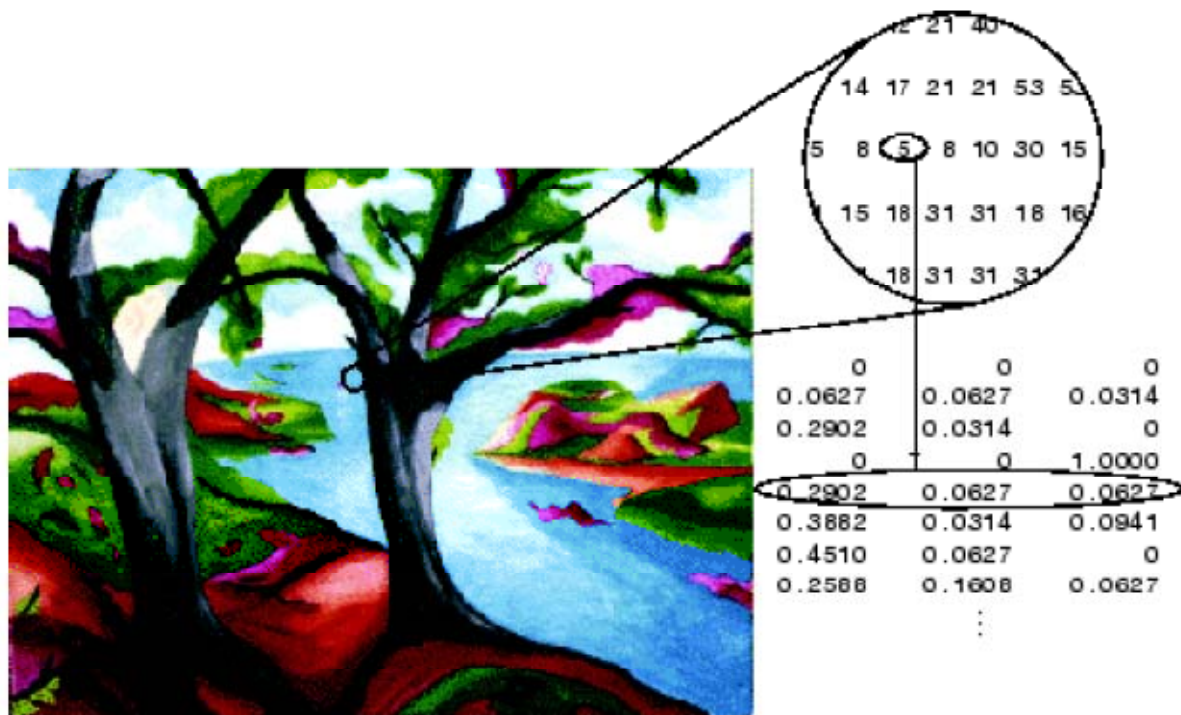


Image Courtesy of Susan Cohen

## Pixel Values Index to Colormap Entries in Indexed Images

معمولاً همراه هر تصویر اندیس‌گذاری شده، ماتریس رنگ مربوطه نیز ذخیره می‌شود و اگر به کمک دستور imread شما تصویری را بخوانید، ماتریس رنگ نیز به طور خودکار خوانده می‌شود. بنابراین، در فضای کاری خود باید دو متغیر مجزا مشاهده کنید.

تعداد نسبتاً زیادی ماتریس رنگ از قبل مشخص وجود دارد که شما می‌توانید هر کدام را که مایل بودید استفاده کنید (حتی می‌توانید خودتان یک ماتریس رنگ ایجاد کرده و از آن استفاده کنید).

بین مقدار عناصر آرایه X و تعداد سطرهای ماتریس رنگ map همواره رابطه‌ای وجود دارد و نوع این رابطه به نوع اعداد آرایه (به اصطلاح کلاس<sup>17</sup> تصویر) بستگی دارد. اگر نوع تصویر برابر single و یا double باشد، آرایه X شامل اعدادی بین ۱ تا p است که p برابر طول (یا تعداد سطرهای) ماتریس رنگ است. مقدار ۱ در

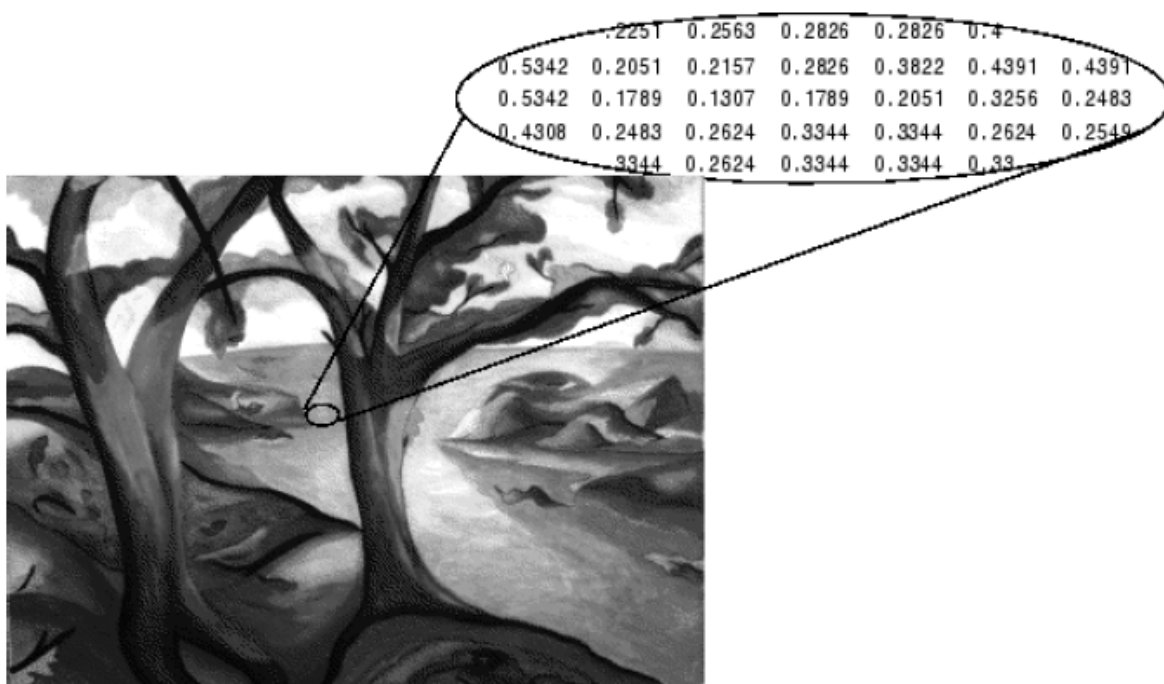
<sup>17</sup> Colormap

<sup>18</sup> Class

آرایه X به اولین سطر (یا رنگ) ماتریس رنگ، مقدار ۲ به دومین سطر، ... ، و مقدار p به p امین سطر ماتریس رنگ اشاره می‌کند. اگر نوع تصویر برابر logical، uint8، و یا uint16 باشد، مقدار صفر در آرایه‌ی X به اولین سطراز ماتریس رنگ، مقدار ۱ به دومین سطر، (و همین طور الی آخر) اشاره می‌کند.

### ۳- تصاویر سطح خاکستری

یک تصویر سطح خاکستری<sup>۱۹</sup> یک ماتریس دو بعدی است که عناصر آن معرف شدت روشنایی پیکسل مربوطه در محدوده‌ای مشخص می‌باشند. به طور پیش فرض در این کتابچه از نماد I برای اشاره به تصاویر سطح خاکستری استفاده شده است.



### Pixel Values in a Grayscale Image Define Gray Levels

نوع (یا کلاس) یک تصویر سطح خاکستری میتواند برابر single، double، uint8، uint16، و یا int16 باشد. متلب معمولاً برای نمایش تصاویر سطح خاکستری نیز از ماتریس رنگ (مشابه با تصاویر اندیس گذاری شده) استفاده می‌کند. اما برای ذخیره‌ی آنها معمولاً از ماتریس رنگ استفاده نمی‌کند. اگر نوع تصویر برابر single و یا double باشد، طبق ماتریس رنگ پیش‌فرض، مقدار صفر متناظر با سیاه کامل و مقدار ۱ متناظر با رنگ سفید کامل است. برای تصاویری از نوع uint8، uint16، و یا int16 مقدار intmin(class(I)) متناظر با رنگ سیاه کامل و مقدار intmax(class(I)) متناظر با رنگ سفید کامل است.

<sup>19</sup> Grayscale (Also : gray scale, gray-scale, gray level)

## ۴- تصاویر رنگی

یک تصویر رنگی<sup>۲۰</sup>، تصویری است که هر پیکسل آن با سه عدد مشخص می‌شود که هر عدد متناظر با شدت یکی از رنگهای قرمز، سبز، و آبی است. (در حقیقت، میدانیم که هر رنگ را میتوان با ترکیبی از این رنگهای اصلی به دست آورد). در متلب هر تصویر رنگی به صورت یک آرایه‌ی  $m \times n \times 3$  (یعنی یک آرایه سه بعدی) ذخیره می‌شود. این آرایه، در حقیقت، شامل سه ماتریس رنگ است که هر ماتریس مشخص کننده‌ی شدت یکی از رنگها برای تمام پیکسلهای مختلف می‌باشد. در تصاویر رنگی، دیگر از ماتریس رنگ استفاده نمی‌شود.

0.2235	0.1294	<b>Blue</b>	0.4198			
0.5804	0.2902	<b>0.0627</b>	0.2902	0.2902	0.4824	0.2235
0.5804	0.0627	0.0627	0.0627	0.2235	0.2588	
0.5176	0.1922	0.0627	<b>Green</b>	0.1922	0.2588	0.2588
0.5176	0.1294	<b>0.1608</b>	0.1294	0.1294	0.2588	0.2588
0.5176	0.1608	0.0627	0.1608	0.1922	0.2588	0.2588
0.5490	0.2235	0.5490	<b>Red</b>	0.7412	0.7765	0.7765
0.5490	0.3882	<b>0.5176</b>	0.5804	0.5804	0.7765	0.7765
0.5490	0.2588	0.2902	0.2588	0.2235	0.4824	0.2235
0.5490	0.2235	0.1608	0.2588	0.2588	0.1608	0.2588
0.5490	0.2588	0.1608	0.2588	0.2588	0.2588	0.2588



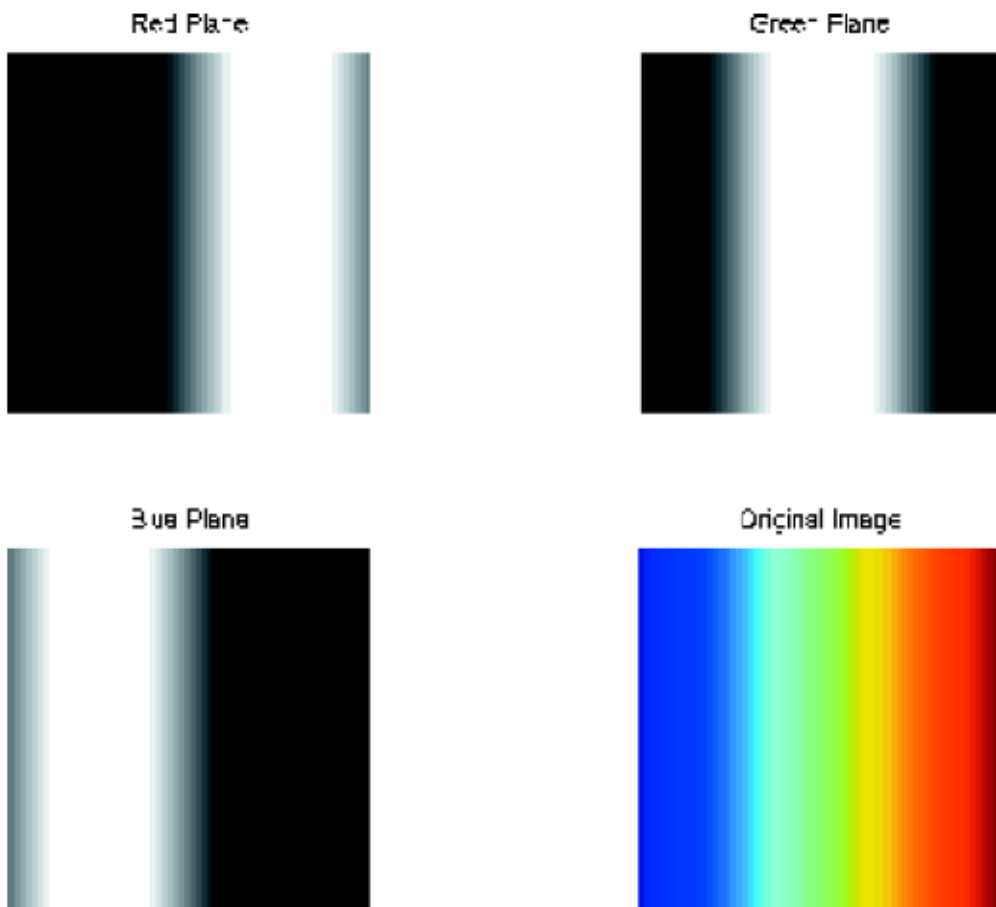
The Color Planes of a Tricolor Image

هر تصویر رنگی میتواند از یکی از انواع `uint8`، `uint16`، `single`، و یا `double` باشد. در حالت `single` و `double` هر مولفه رنگ عددی بین صفر تا ۱ است. بنابراین مولفه‌ی  $(0,0,0)$  به معنای سیاه کامل و مولفه‌ی  $(1,1,1)$  به معنای سفید کامل است. مولفه‌های رنگ هر پیکسل واقع در مختصات  $(x,y)$  در بعد سوم از آرایه‌ی سه بعدی متناظر با تصویر قرار داده میشوند. مثلاً مولفه‌های قرمز، سبز، و آبی پیکسل واقع در مختصات  $(4,15)$  از تصویر RGB در محلهای به ترتیب `RGB(4,15,1)`، `RGB(4,15,2)` و `RGB(4,15,3)` قرار دارند.

مثال زیر یک نمونه از ترکیب رنگها را نشان میدهد:

```
RGB=reshape(ones(64,1)*reshape(jet(64),1,192),[64,64,3]);
R=RGB(:,:,1);
G=RGB(:,:,2);
B=RGB(:,:,3);
imshow(R)
figure, imshow(G)
figure, imshow(B)
figure, imshow(RGB)
```

نتیجه:



**The Separated Color Planes of an RGB Image**

## تبدیل بین انواع مختلف تصویری

برای تبدیل نوع سطح خاکستری به نوع رنگی کافی است که تصویر را سه بار کپی کرده و در یکی از سه بعد تصویر رنگی مربوطه قرار دهیم. اگر I یک تصویر سطح خاکستری باشد و بخواهیم از روی آن تصویر رنگی RGB بسازیم به طریق زیر میتوانیم عمل کنیم:

```
RGB = cat(3,I,I,I);
```

تمرین: در مورد دستور cat راهنما بگیرید و نحوه کار آن را توضیح دهید.

توابع مربوط به تبدیل انواع:

Function	Description
demosaic	Convert Bayer pattern encoded image to truecolor (RGB) image.
dither	Use dithering to convert a grayscale image to a binary image or to convert a truecolor image to an indexed image.
gray2ind	Convert a grayscale image to an indexed image.
grayslice	Convert a grayscale image to an indexed image by using multilevel thresholding.
im2bw	Convert a grayscale image, indexed image, or truecolor image, to a binary image, based on a luminance threshold.
ind2gray	Convert an indexed image to a grayscale image.
ind2rgb	Convert an indexed image to a truecolor image.
mat2gray	Convert a data matrix to a grayscale image, by scaling the data.
rgb2gray	Convert a truecolor image to a grayscale image.  Note: To work with images that use other color spaces, such as HSV, first convert the image to RGB, process the image, and then convert it back to the original color space. For more information about color space conversion routines, see Chapter 14, "Color".
rgb2ind	Convert a truecolor image to an indexed image.

تمرین: در مورد توابع فوق (به جز دو مورد اول) تحقیق کنید.

## تبدیل بین انواع کلاسهای مختلف تصویری

برای تبدیل نوعهای uint8 و uint16 به نوع double از دستور double استفاده کنید اما توجه داشته باشید که گاهی اوقات لازم است در ادامه اعداد را نرمالیزه و یا بایاس هم کنید. برای اینکه مطمئن شوید که

کار نرمالیزه کردن و یا بایاس کردن به درستی انجام میشود میتوانید از توابع زیر برای تبدیلات مختلف خود استفاده کنید:

`im2uint16, im2int16, im2uint8, im2single, im2double`

برای مثال اگر تصویری از نوع `double` دارید (و بنابراین مقادیر عددی آن بین صفر تا ۱ می باشد)، به کمک دستور زیر آن را به نوع `uint8` تبدیل کنید (بنابراین در تصویر خروجی، مقادیر بین صفر تا ۲۵۵ می باشند):

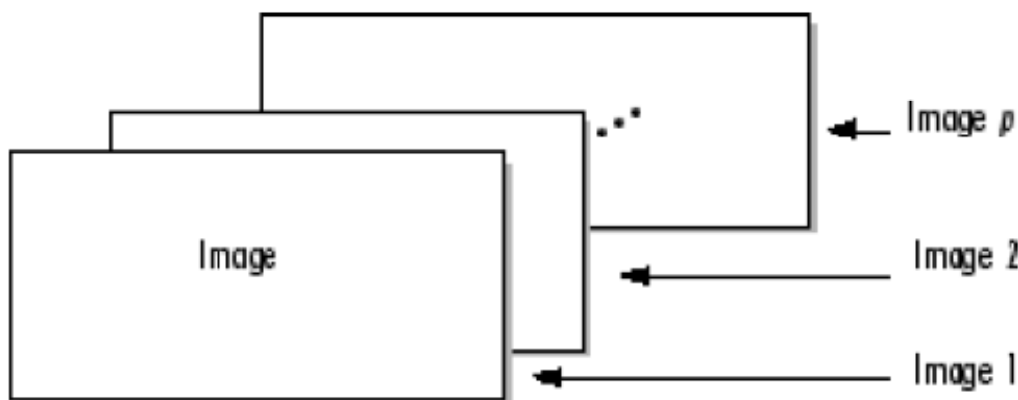
`RGB2 = im2uint8(RGB1);`

توجه کنید که در انجام برخی تبدیلات مقداری از اطلاعات را ممکن است از دست دهید. برای مثال، اگر تصویری از نوع `uint16` را بخواهید به نوع `uint8` تبدیل کنید، تعداد سطوح از ۶۵۵۳۶ به ۲۵۶ کاهش می یابد و بنابراین بدیهی است که مقداری از اطلاعات را از دست می دهید.

نکته دیگر در مورد تبدیل تصاویر اندیس گذاری شده است. توجه کنید که همیشه و در هر حالتی نمیتوانید یک تصویر اندیس گذاری شده را به نوع دیگری مانند `uint8` تبدیل کنید. برای مثال اگر تصویر اندیس گذاری شده دارای ۳۰۰ رنگ در ماتریس رنگ خود باشد، از آنجا که نوع `uint8` فقط قادر به نمایش ۲۵۶ سطح مختلف است، باید ابتدا تعداد رنگهای ماتریس رنگ را به کمک تابع `imapprox` کاهش داده و سپس از تبدیل مورد نظر استفاده کنید.

### کار با دنباله های تصاویر

برخی تصاویر در حوزه زمان (مانند فریمهای ویدیویی) یا مکان (مانند تصاویر MRI) به هم مرتبط هستند. این گونه تصاویر را دنباله های تصویری<sup>۲۱</sup> یا انباشته های تصویری<sup>۲۲</sup> می گویند. برای ذخیره و پردازش دنباله های تصویری باید یک آرایه سه بعدی یا چهار بعدی ایجاد کرد. برای ایجاد دنباله هایی از تصاویر سطح خاکستری و باینری نیاز به آرایه سه بعدی ( $m \times n \times p$ ) و برای ایجاد دنباله هایی از تصاویر رنگی نیاز به آرایه چهار بعدی ( $m \times n \times 3 \times p$ ) داریم.



### Multidimensional Array Containing an Image Sequence

<sup>21</sup> Image Sequences

<sup>22</sup> Image Stacks

برخی توابع در متلب آرایه‌های چند بعدی قبول میکنند اما لزوماً به چشم تصویر به آنها نگاه نمیکنند؛ بنابراین، در استفاده از آنها باید دقت کنید. جدول زیر این توابع را لیست کرده و به شما توضیح می‌دهد که چگونه از آنها برای دنباله‌های تصویری بهره ببرید (مثلاً به شما میگوید که برای یک دستور باید از کدام قالب آن استفاده کنید).

Function	Image Sequence Dimensions	Guideline When Used with an Image Sequence
<code>bwlabeln</code>	$m$ -by- $n$ -by- $p$ only	Must use the <code>bwlabeln(BW, conn)</code> syntax with a 2-D connectivity.
<code>deconvblind</code>	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	PSF argument can be either 1-D or 2-D.
<code>deconvlucy</code>	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	PSF argument can be either 1-D or 2-D.
<code>edgetaper</code>	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	PSF argument can be either 1-D or 2-D.
<code>entropyfilt</code>	$m$ -by- $n$ -by- $p$ only	<code>nhood</code> argument must be 2-D.
<code>imabsdiff</code>	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	Image sequences must be the same size.
<code>imadd</code>	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	Image sequences must be the same size. Cannot add scalar to image sequence.
<code>imbothat</code>	$m$ -by- $n$ -by- $p$ only	SE argument must be 2-D.
<code>imclose</code>	$m$ -by- $n$ -by- $p$ only	SE argument must be 2-D.
<code>imdilate</code>	$m$ -by- $n$ -by- $p$ only	SE argument must be 2-D.
<code>imdivide</code>	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	Image sequences must be the same size.
<code>imerode</code>	$m$ -by- $n$ -by- $p$ only	SE argument must be 2-D.
<code>imextendedmax</code>	$m$ -by- $n$ -by- $p$ only	Must use the <code>imextendedmax(I, h, conn)</code> syntax with a 2-D connectivity.

<b>Function</b>	<b>Image Sequence Dimensions</b>	<b>Guideline When Used with an Image Sequence</b>
imextendedmin	$m$ -by- $n$ -by- $p$ only	Must use the <code>imextendedmin(I,h,conn)</code> syntax with a 2-D connectivity.
imfilter	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	With grayscale images, $h$ can be 2-D. With truecolor images (RGB), $h$ can be 2-D or 3-D.
imhmax	$m$ -by- $n$ -by- $p$ only	Must use the <code>imhmax(I,h,conn)</code> syntax with a 2-D connectivity.
imhmin	$m$ -by- $n$ -by- $p$ only	Must use the <code>imhmin(I,h,conn)</code> syntax with a 2-D connectivity.
imlincomb	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	Image sequences must be the same size.
immultiply	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	Image sequences must be the same size.
imopen	$m$ -by- $n$ -by- $p$ only	SE argument must be 2-D.
imregionalmax	$m$ -by- $n$ -by- $p$ only	Must use the <code>imextendedmax(I,conn)</code> syntax with a 2-D connectivity.
imregionalmin	$m$ -by- $n$ -by- $p$ only	Must use the <code>imextendedmin(I,conn)</code> syntax with a 2-D connectivity.
imtransform	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	TFORM argument must be 2-D.
imsubtract	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	Image sequences must be the same size.
imtophat	$m$ -by- $n$ -by- $p$ only	SE argument must be 2-D.
padarray	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	PADSIZE argument must be a two-element vector.
rangefilt	$m$ -by- $n$ -by- $p$ only	NHOOD argument must be 2-D.
stdfilt	$m$ -by- $n$ -by- $p$ only	NHOOD argument must be 2-D.



Function	Image Sequence Dimensions	Guideline When Used with an Image Sequence
tformarray	$m$ -by- $n$ -by- $p$ or $m$ -by- $n$ -by-3-by- $p$	T must be 2-D to 2-D (compatible with imtransform). R must be 2-D. TDIMS_A and TDIMS_B must be 2-D, i.e., [2 1] or [1 2] TSIZE_B must be a two-element array [D1 D2], where D1 and D2 are the first and second transform dimensions of the output space. TMAP_B must be [TSIZE_B 2] F can be a scalar or a $p$ -by-1 array for $m$ -by- $n$ -by- $p$ arrays, or it can be a scalar, 1-by- $p$ array, 3-by-1 array, or 3-by- $p$ array, for $m$ -by- $n$ -by-3-by- $p$ arrays.
watershed	$m$ -by- $n$ -by- $p$ only	Must use watershed(I, conn) syntax with a 2-D connectivity.

### مثال: پردازش یک دنباله از تصاویر

در این مثال، یک سری از تصاویر از یک دایرکتوری خاص خوانده شده و به درون فضای کاری متلب منتقل می‌شوند. در این مثال، تصاویر به صورت یک آرایه سه بعدی  $m \times n \times p$  خوانده و در فضای کاری متلب ذخیره می‌شوند. سپس تابع stdfilt روی همه تصاویر اعمال می‌شود. این تابع کار فیلتر کردن مبتنی بر انحراف معیار را روی تک تک تصاویر انجام می‌دهد. با توجه به توضیحات جدول اخیر در مورد تابع stdfilt ملاحظه میکنید که باید از آرگومان nhood استفاده کنیم. بنابراین، ابتدا از تابع stdfilt راهنما بگیرید:

```
>> help stdfilt
```

STDFILT Local standard deviation of image.

J = STDFILT(I) returns the array J, where each output pixel contains the standard deviation value of the 3-by-3 neighborhood around the corresponding pixel in the input image I. I can have any dimension. The output image J is the same size as the input image I.

For pixels on the borders of I, STDFILT uses symmetric padding. In symmetric padding, the values of padding pixels are a mirror reflection of the border pixels in I.

J = STDFILT(I,NHOOD) performs standard deviation filtering of the input image I where you specify the neighborhood in NHOOD. NHOOD is a multidimensional array of zeros and ones where the nonzero elements specify the neighbors. NHOOD's size must be odd in each dimension.

By default, STDFILT uses the neighborhood ones(3). STDFILT determines the center element of the neighborhood by  $\text{FLOOR}((\text{SIZE}(\text{NHOOD}) + 1)/2)$ . For information about specifying neighborhoods, see Notes.

### Class Support

I can be logical or numeric and must be real and nonsparse. NHOOD can be logical or numeric and must contain zeros and/or ones. I and NHOOD can have any dimension. J is double.

### Notes

To specify the neighborhoods of various shapes, such as a disk, use the STREL function to create a structuring element object and then use the GETNHOOD function to extract the neighborhood from the structuring element object.

### Examples

```
I = imread('circuit.tif');
J = stdfilt(I);
imshow(I);
figure, imshow(J,[]);
```

See also std2, rangefilt, entropyfilt, strel, GETNHOOD.

ملاحظه می‌کنید که nhood تعیین کننده پیکسل‌هایی است که باید در کار محاسبه‌ی انحراف معیار محلی از آنها استفاده کنیم. در مثال فعلی، از ones(3) برای این آرگومان استفاده شده است. یعنی یک همسایگی دو بعدی تعریف شده است.

برنامه‌ی مثال:

```
% Create an array of filenames that make up the image sequence
fileFolder = fullfile(matlabroot,'toolbox','images','imdemos');
dirOutput = dir(fullfile(fileFolder,'AT3_1m4_*.tif'));
fileNames = {dirOutput.name}';
numFrames = numel(fileNames);
I = imread(fileNames{1});
% Preallocate the array
sequence = zeros([size(I) numFrames],class(I));
sequence(:,1) = I;
% Create image sequence array
for p = 2:numFrames
    sequence(:,p) = imread(fileNames{p});
end
```

```
% Process sequence
sequenceNew = stdfilt(sequence,ones(3));
% View results
figure;
for k = 1:numFrames
    imshow(sequence(:,:,k));
    title(sprintf('Original Image # %d',k));
    pause(1);
    imshow(sequenceNew(:,:,k),[]);
    title(sprintf('Processed Image # %d',k));
    pause(1);
end
```

برای توضیحات این برنامه از برخی توابعی که احتمالاً تاکنون آنها را ندیده‌اید، راهنما بگیرید. مثلاً برای تابع `fullfile`

```
>> help fullfile
```

FULLFILE Build full filename from parts.

FULLFILE(D1,D2, ... ,FILE) builds a full file name from the directories D1,D2, etc and filename FILE specified. This is conceptually equivalent to

$$F = [D1 \text{ filesep } D2 \text{ filesep } \dots \text{ filesep } FILE]$$

except that care is taken to handle the cases where the directory parts D1, D2, etc. may begin or end in a filesep. Specify FILE = "" to build a pathname from parts.

#### Examples

To build platform dependent paths to files:

```
fullfile(matlabroot,'toolbox','matlab','general','Contents.m')
```

To build platform dependent paths to a directory:

```
addpath(fullfile(matlabroot,'toolbox','matlab',''))
```

See also filesep, pathsep, fileparts.

برای دستور `dir` :

```
>> help dir
```

DIR List directory.

DIR directory\_name lists the files in a directory. Pathnames and wildcards may be used. For example, DIR \*.m lists all the M-files in the current directory.

D = DIR('directory\_name') returns the results in an M-by-1

structure with the fields:

name -- Filename

date -- Modification date

bytes -- Number of bytes allocated to the file

isdir -- 1 if name is a directory and 0 if not

datenum -- Modification date as a MATLAB serial date number.

This value is locale-dependent.

See also what, cd, type, delete, ls, rmdir, mkdir, datenum.

## فصل سوم

# خواندن و نوشتن تصاویر

هدف: آشنایی با نحوه خواندن، نوشتن، و کسب اطلاعات در مورد تصاویر

برای کسب اطلاعات در مورد یک فایل تصویری ذخیره شده در حافظه کامپیوتر از دستور `imfinfo` استفاده کنید. برای دیدن لیست قالبهای تصویری مورد پشتیبانی در متلب، در پنجره فرمان `imformats` را وارد کنید.

مثالی از خواندن یک فایل تصویری:

```
RGB = imread('football.jpg');
```

برخی قالبهای تصویری مانند `jpg` برای هر پیکسل ۸ بیت تخصیص می‌دهند. بنابراین، متلب این نوع تصاویر را به صورت نوع `uint8` نمایش می‌دهد. اما برخی قالبهای تصویری مانند `TIFF` و `PNG` می‌توانند برای هر پیکسل ۱۶ بیت تخصیص دهند، بنابراین دستور `imread` باعث تولید آرایه‌ای از نوع `uint16` می‌گردد. برای خواندن یک تصویر به صورت یک تصویر اندیس گذاری شده، متلب از دو متغیر یکی برای ماتریس رنگ و دیگری برای ماتریس اشاره‌گرها. دستور `imread` همواره اطلاعات ماتریس رنگ را در ماتریسی از نوع `double` قرار میدهد گرچه ماتریس اشاره‌گرها خود از نوع `uint8` و یا `uint16` است. مثالی از خواندن یک تصویر به صورت یک تصویر اندیس گذاری شده:

```
[X,map] = imread('trees.tif');
```

برخی قالبهای تصویری مانند `tiff` قادر به ذخیره بیش از یک تصویر در خود هستند. در اینگونه واقع، دستور `imread` به طور پیش فرض اولین تصویر (یا فریم) را می‌خواند مگر اینکه از قالب دستور العمل مناسب برای خواندن بقیه‌ی فریمها استفاده کنیم. در مثال زیر، ۲۷ تصویر از یک فایل با قالب `tiff` خوانده شده و در یک آرایه چهار بعدی قرار داده می‌شوند. البته جلوتر می‌توانید از دستور `imfinfo` کمک بگیرید تا ببینید چند فریم در فایل ذخیره شده است.

```
mri = zeros([128 128 1 27],'uint8'); % preallocate 4-D array
```

```
for frame=1:27
```

```
[mri(:,:,,frame),map] = imread('mri.tif',frame);
```

```
end
```

اگر میخواهید فایل بزرگی را بخوانید برای اینکه مشکل کمبود حافظه پیش نیاید، یک راه این است که از پردازش بلوکی (فصل ۱۵) استفاده کنید.

برای ذخیره داده‌های یک تصویر به صورت یک فایل تصویری روی حافظه کامپیوتر از دستور `imwrite` می‌توانید استفاده کنید. در مثال زیر ابتدا یک تصویر اندیس گذاری شده که در فایل `mat` با قالب `mat` ذخیره شده است خوانده شده و به فضای کاری متلب منتقل می‌شود. سپس این تصویر به صورت یک فایل تصویری با قالب `bmp` در حافظه کامپیوتر ذخیره می‌شود.

```
>>load clown
```

```
>>whos
```

Name	Size	Bytes	Class
X	200x320	512000	double array
caption	2x1	4	char array
map	81x3	1944	double array

Grand total is 64245 elements using 513948 bytes

```
>>imwrite(X,map,'clown.bmp')
```

برخی قالبهای تصویری یک سری پارامترهای خاص خود دارند که در دستور `imwrite` می‌توانید آنها را تعیین کنید. برای دیدن این پارامترهای خاص باید از دستور `imwrite` در محیط راهنمای متلب (کلید F1) راهنما بگیرید (در اینجا دستور `help` کمک چندانی نمی‌کند).  
برای مثال:

```
imwrite(I,'clown.png','BitDepth',4);
imwrite(A,'myfile.jpg','Quality',100);
```

**تمرین:** حتماً به راهنمای مذکور مراجعه کنید و چند قالب تصویری و پارامترهایشان را بررسی کنید.

برخی قالبها مانند `tiff` به شما اجازه می‌دهند که تصویرتان را به صورت ۱ بیت بر پیکسل (1 bpp) ذخیره کنید. در این حالت اصطلاحاً گفته می‌شود عمق بیت<sup>۲۳</sup> برابر ۱ بیت است. بنابراین کاملاً مناسب ذخیره تصاویر باینری است. از طرف دیگر، اگر چنین تصاویر ذخیره شده‌ای را با دستور `imread` بخوانید، ماتریسی از نوع `logical` (در فضای کاری متلب) خواهید داشت.  
مثال زیر یک تصویر باینری (با قالب `png`) را خوانده و آن را به صورت قالب `tiff` ذخیره میکند:

```
>>BW = imread('text.png');
>>imwrite(BW,'test.tif');
```

در مثال فوق اگر میخواهید مطمئن شوید که تصویر مورد بررسی به صورت ۱ بیت بر پیکسل است، از دستور `imfinfo` کمک گرفته و میدان `BitDepth` را بررسی کنید:

```
>>info = imfinfo('test.tif');
>>info.BitDepth
ans =
    1
```

اگر در فضای کاری متلب، تصویری از یک کلاس مشخص داشته باشید و بخواهید ببینید که کلاس تصویر حاصل از ذخیره به کمک دستور `imwrite` چه خواهد بود، از جدول زیر کمک بگیرید:

Storage Class of Image	Storage Class of Output Image File
logical	If the output image file format supports 1-bit images, <code>imwrite</code> creates a 1-bit image file.  If the output image file format specified does not support 1-bit images, <code>imwrite</code> exports the image data as a <code>uint8</code> grayscale image.
uint8	If the output image file format supports unsigned 8-bit images, <code>imwrite</code> creates an unsigned 8-bit image file.
uint16	If the output image file format supports unsigned 16-bit images (PNG or TIFF), <code>imwrite</code> creates an unsigned 16-bit image file.  If the output image file format does not support 16-bit images, <code>imwrite</code> scales the image data to class <code>uint8</code> and creates an 8-bit image file.
int16	Partially supported; depends on file format.
single	Partially supported; depends on file format.
double	MATLAB scales the image data to <code>uint8</code> and creates an 8-bit image file, because most image file formats use 8 bits.

### خواندن تصاویر با محدوده دینامیکی زیاد

تصاویر با محدوده دینامیکی زیاد معمولاً در فایل‌هایی با پسوند `hdr` ذخیره می‌شوند. برای خواندن این فایل‌ها از دستور `hdrread` استفاده کنید. برای مثال:

```
>>hdr_image = hdrread('office.hdr');
```

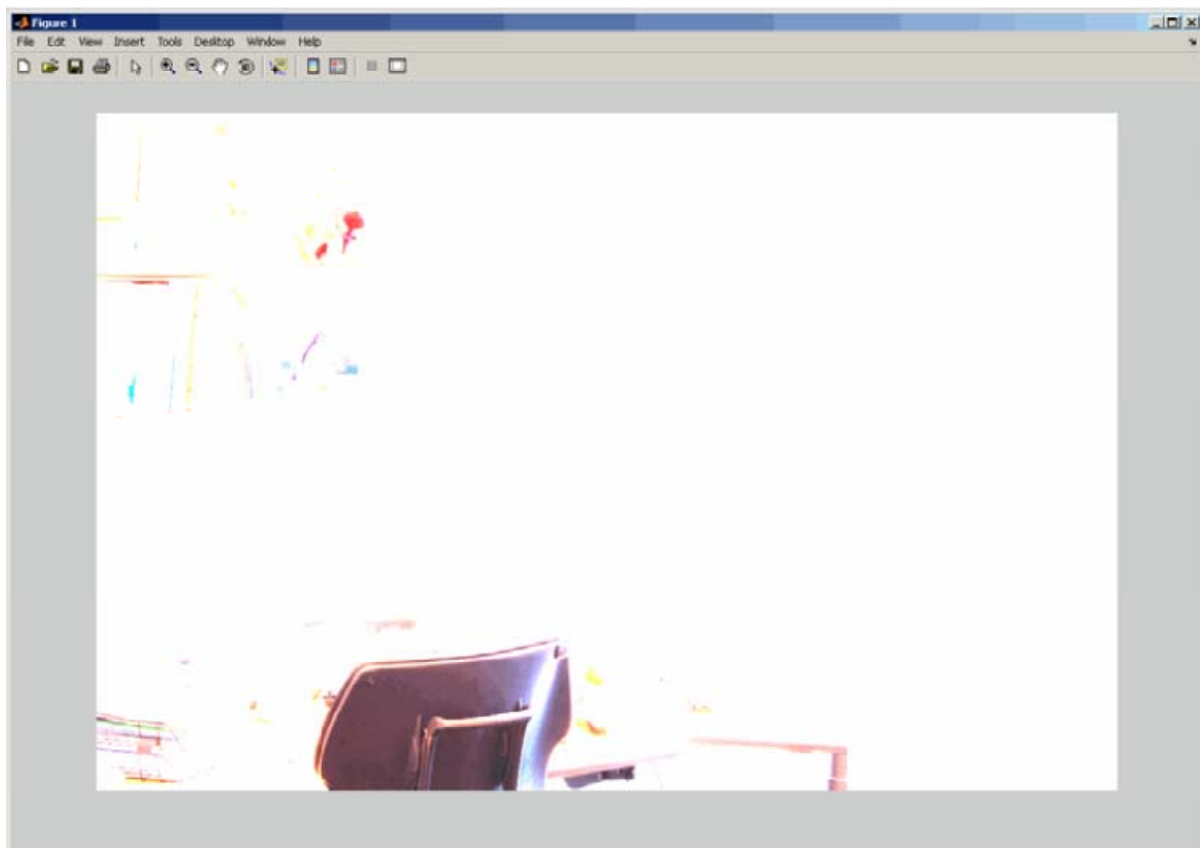
```
>>whos
```

```
Name          Size          Bytes          Class Attributes
hdr_image     665x1000x3   7980000        single
```

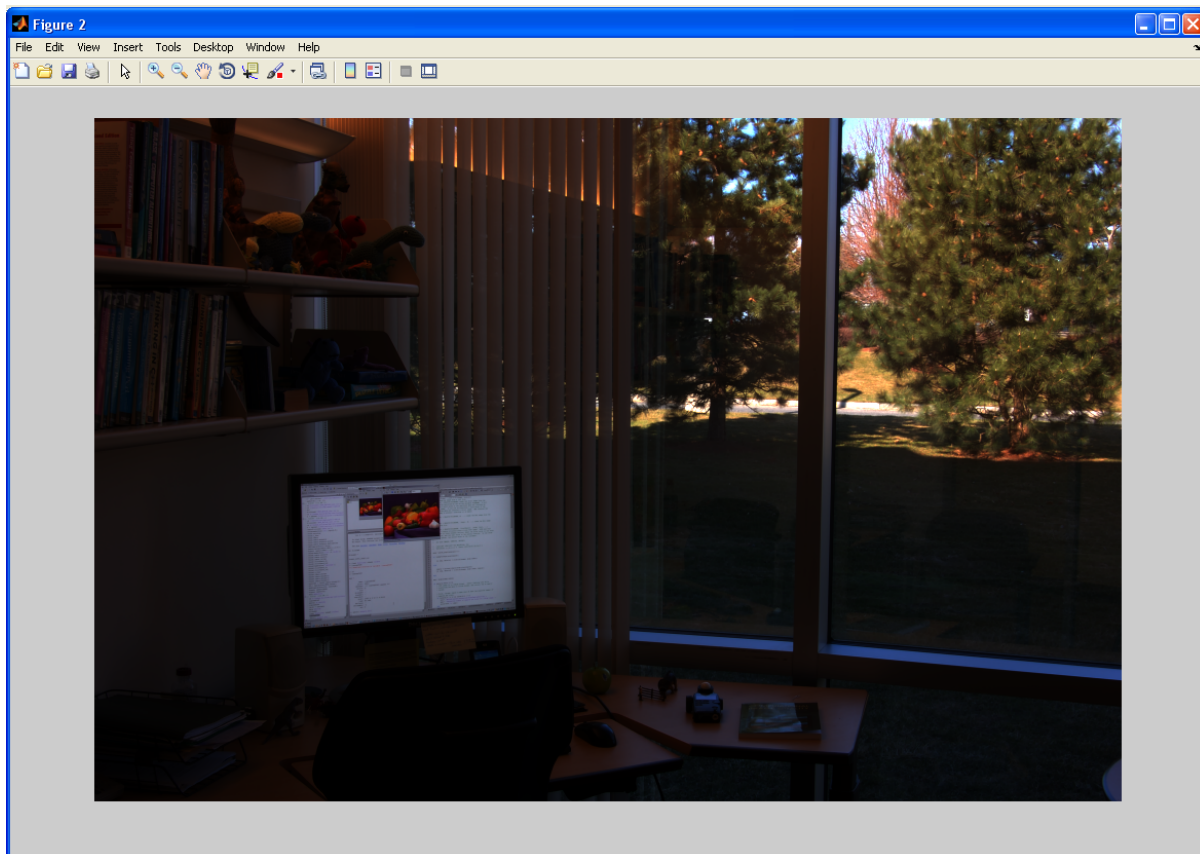
برای نمایش یک تصویر با محدوده دینامیکی زیاد باید ابتدا آن را متناسب با محدوده دینامیکی مانیتور تغییر دهید (اصلاح کنید). به این فرآیند تصحیح، نگاشت تن<sup>۲۴</sup> اطلاق می‌شود. اگر به سادگی (بدون انجام تصحیح مذکور)، از `imshow` برای نمایش این گونه تصاویر استفاده کنید، نمایش مناسبی به دست نمی‌آورد:

```
>>imshow(hdr_image);
```





البته ممکن است نمایش زیر را هم ببینید:



برای انجام نگاشت تن، از تابع tonemap استفاده کنید:

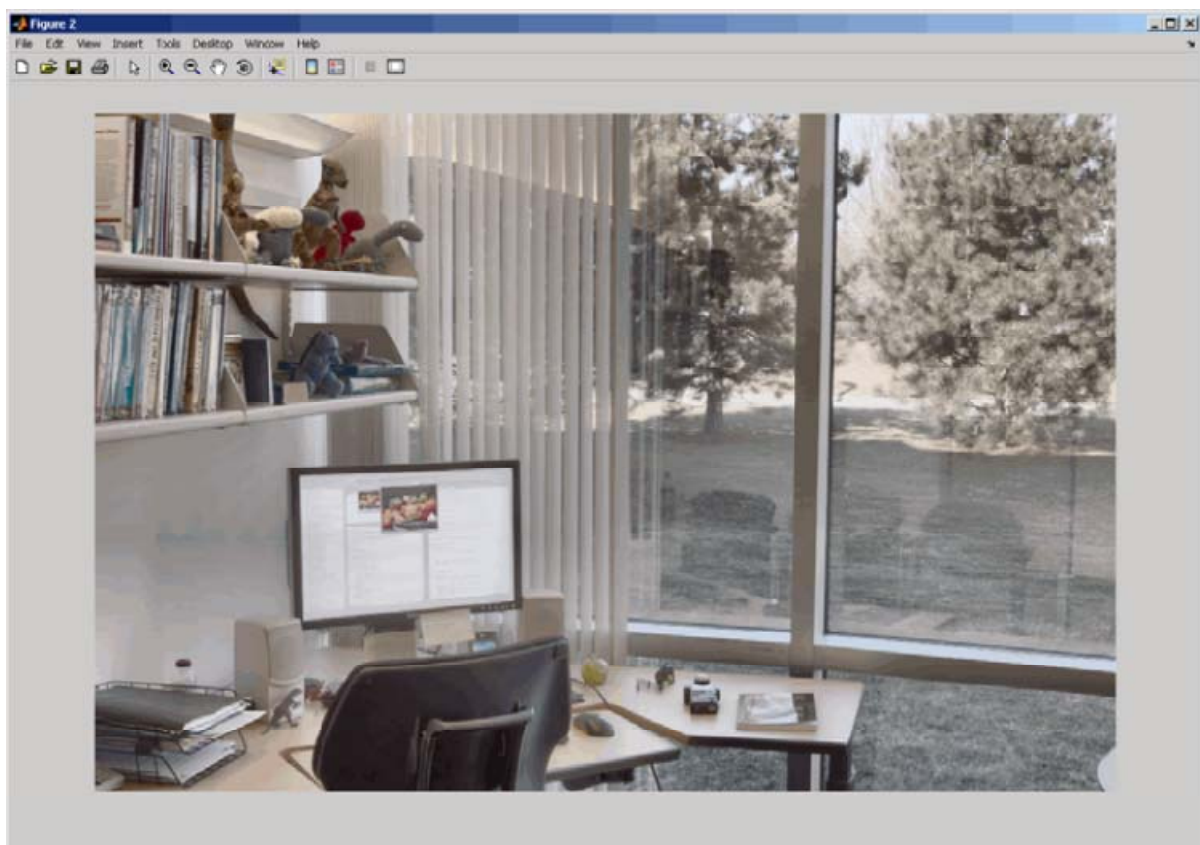
```
>>rgb = tonemap(hdr_image);
```

```
>>whos
```

Name	Size	Bytes	Class	Attributes
hdr_image	665x1000x3	7980000	single	
rgb	665x1000x3	1995000	uint8	

حال می‌توانید از دستور imshow استفاده کنید:

```
>>imshow(rgb);
```



برای ذخیره یک تصویر با محدوده دینامیکی زیاد روی حافظه کامپیوتر با قالب hdr می‌توانید از دستور  
hdrwrite استفاده کنید:

```
hdrwrite(hdr,'filename');
```

## فصل چهارم

# نمایش و واریسی تصاویر

هدف: آشنایی با نحوه نمایش، واریسی و تحلیل تصاویر

## دستور imshow

یک راه استفاده از دستور imshow : (ابتدا تصویر را بخوانید و سپس آن را نمای دهید)

```
moon = imread('moon.tif');
imshow(moon);
```

راه دیگر: نام فایل را مستقیماً ذکر کنیم :

```
imshow('moon.tif');
```

البته این راه باعث نمیشود که داده‌های عددی تصویر به فضای کاری متلب منتقل شود؛ برای این کار باید از دستور getimage استفاده کنید:

```
moon = getimage;
```

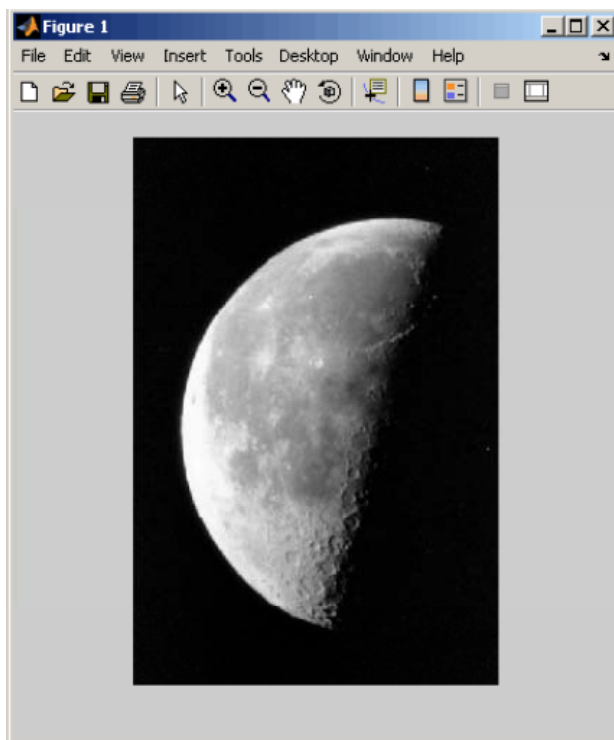
در حالت پیش فرض، دستور imshow یک تصویر با بزرگنمایی ۱۰۰٪ (یعنی به ازاء هر پیکسل از تصویر یک پیکسل از صفحه نمایش کامپیوتر متناظر می‌شود) نمایش می‌دهد. اگر بخواهید این پیش فرض را تغییر دهید (مثلاً با بزرگنمایی ۱۵۰٪):

```
pout = imread('pout.tif');
imshow(pout, 'InitialMagnification', 150)
```

اگر بخواهید تصویر متناسب با ابعاد فعلی پنجره موجود نمایش داده شود، از مقدار 'fit' به جای مقدار عددی بزرگنمایی استفاده کنید.

اگر بخواهید مقدار پیش فرض بزرگنمای را تغییر دهید ابتدا پنجره‌ی مکالمه‌ی جعبه ابزار پردازش تصویر را با اجرای دستور iptprefs فراخوانی کرده و سپس مقدار پارامتر ImshowInitialMagnification را تغییر دهید.

هنگام نمای یک تصویر، یک نوار خاکستری رنگ دور تصویر را احاطه کرده است:



'loose'

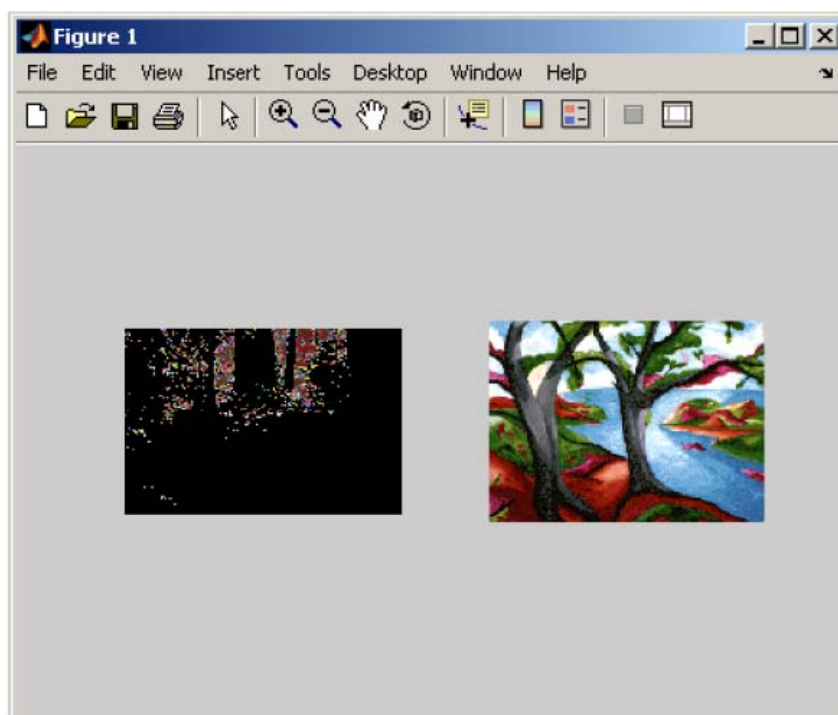
اگر بخواهید این نوار حذف شود، مقدار پارامتر Border در دستور imshow را برابر 'tight' قرار دهید:  
`imshow('moon.tif','Border','tight')`



اگر بخواهید همیشه و برای تمام نمایشها، نوار مذکور حذف شود، از همان پنجره مکالمه‌ی جعبه ابزار پردازش تصویر، مقدار پارامتر `imshowBorder` را برابر 'tight' قرار دهید.

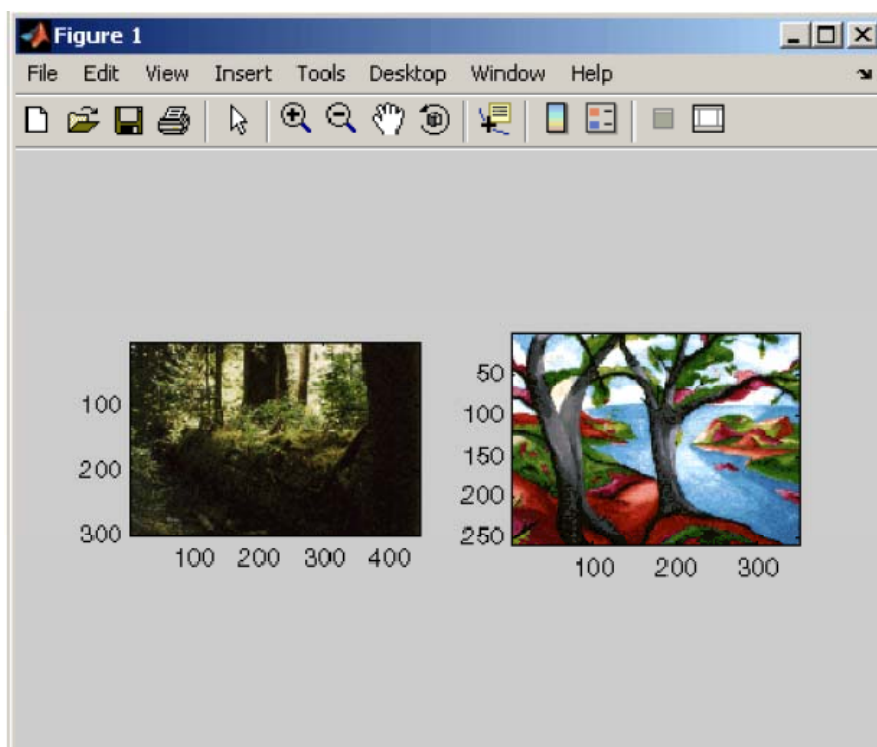
اگر بخواهید در یک پنجره چند تصویر نمایش دهید باید ابتدا به کمک دستور `subplot` پنجره را به چند زیرپنجره تقسیم کنید و سپس از دستور `imshow` و یا دستور `subimage` برای نمایش تصویر استفاده کنید. دستور `subimage` بهتر از `imshow` عمل می‌کند. توجه کنید:

```
[X1,map1]=imread('forest.tif');
[X2,map2]=imread('trees.tif');
subplot(1,2,1), imshow(X1,map1)
subplot(1,2,2), imshow(X2,map2)
```



**Two Images in Same Figure Using the Same Colormap**

```
[X1,map1]=imread('forest.tif');
[X2,map2]=imread('trees.tif');
subplot(1,2,1), subimage(X1,map1)
subplot(1,2,2), subimage(X2,map2)
```



**Two Images in Same Figure Using Separate Colormaps**

## دستور imshow

سه راه برای خواندن یک تصویر به کمک دستور imshow :

**راه اول:** ابتدا دستور imshow را به تنهایی اجرا کرد و در پنجره ظاهر شده، از منوی File گزینه‌ی Open یا گزینه‌ی Import from Workspace را انتخاب کنید (بستگی به محل تصویر مورد نظرتان).  
**راه دوم:** آن را به همراه نام تصویر فراخوانی کنید:

```
moon = imread('moon.tif');
imshow(moon)
```

**راه سوم:** مشابه راه دوم است:

```
imshow('moon.tif');
```

برای تغییر بزرگنمایی از مقدار پیش فرض ۱۰۰٪ :

```
pout = imread('pout.tif');
imshow(pout, 'InitialMagnification', 150)
```

در اینجا هم می‌توانید از مقدار 'fit' استفاده کنید.

## دنباله‌های تصویری

برای پخش دنباله‌های تصویری می‌توان از Movie Player استفاده کرد.

**مثال:**

۱- بارگذاری دنباله‌ی تصویری به فضای کاری متلب

```
>>load mrystack
```

در فضای کاری متغیری به نام mrystack ظاهر می‌شود که آرایه‌ای شامل ۲۱ فریم ۲۵۶ در ۲۵۶ و از نوع uint8 است.

```
mrystack      256x256x21      1276256      uint8
```

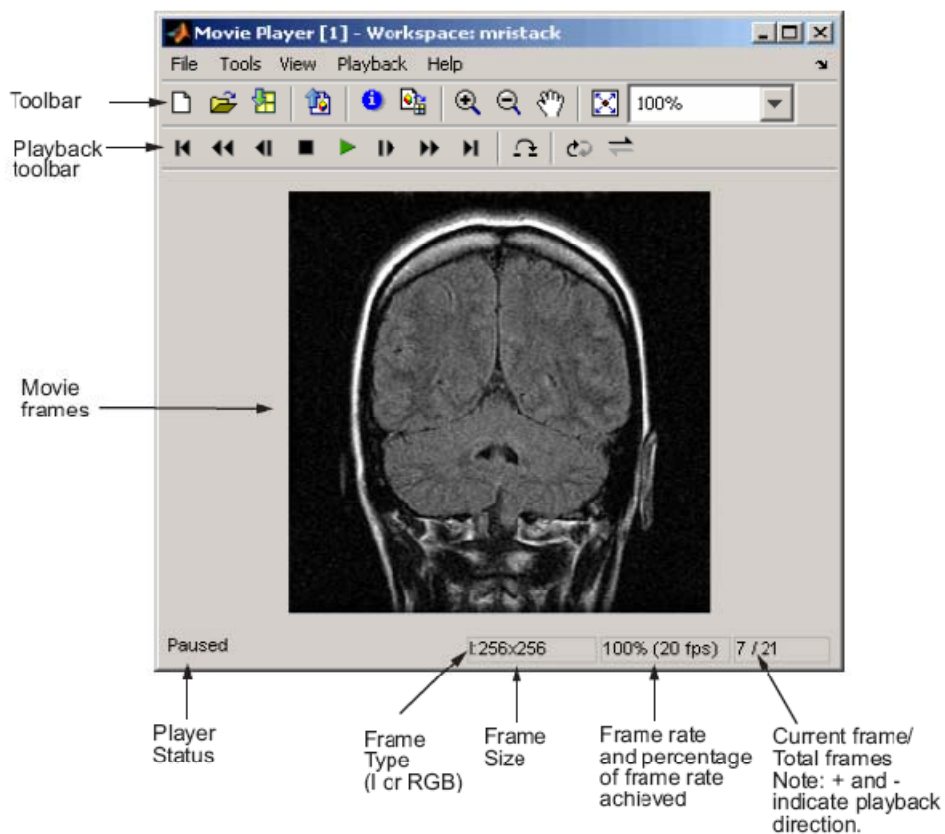
۲- پخش دنباله تصویری در Movie Player به کمک دستور imshow : (نام دنباله را به عنوان آرگومان

ورودی می‌دهیم)

```
imshow(mrystack)
```












حال با ابزارهای پخش موجود در پنجره ظاهر شده می‌توانید فریم به فریم یا به صورت انیمیشن، دنباله را ببینید.


همچنین به اطلاعاتی که در اطراف پنجره مهیا شده است توجه کنید.

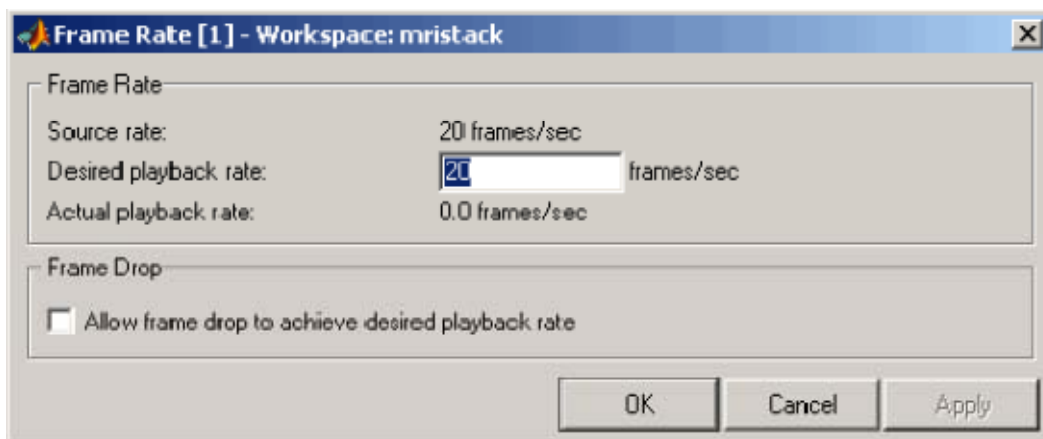


تمرین: قابلیت‌های موجود در پنجره فوق را بررسی کنید.  
جدول زیر حالت‌های مختلف کاری و نحوه انجام هر حالت را نشان می‌دهد.

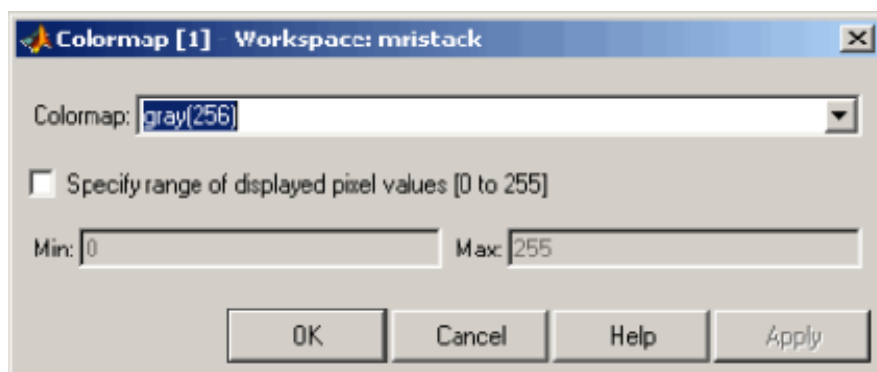


Viewing Option	Playback Control	Keyboard Shortcut
Specify the direction in which to play the image sequence.	Click the Playback mode button  in the Playback toolbar or select <b>Playback Modes</b> from the Playback menu. You can select forward, backward, or autoreverse. As you click the playback mode button, it cycles through these options and the appearance changes to indicate the current selection.	A
View the sequence repeatedly.	Click the Repeat button  in the Playback toolbar or select <b>Playback Modes &gt; Repeat</b> from the Playback menu. You toggle this option on or off.	R
Jump to a specific frame in the sequence.	Click the Jump to button  in the Playback toolbar or select <b>Jump to</b> from the Playback menu. This option opens a dialog box in which you can specify the number of the frame.	J
Stop the sequence.	Click the Stop button  in the Playback toolbar or select <b>Stop</b> from the Playback menu. This button is only enabled when an image sequence is playing.	S
Step through the sequence, one frame at a time, or jump to the beginning or end of the sequence (rewind).	Click one of the navigation buttons  in the Playback toolbar, in the desired direction, or select an option, such as <b>Fast Forward</b> or <b>Rewind</b> from the Playback menu.	Arrow keys Page Up/Page Down L (last frame) F (first frame)
Viewing Option	Playback Control	
Zoom in or out on the image, and pan to change the view.	Click one of the zoom buttons   in the toolbar or select <b>Zoom In</b> or <b>Zoom Out</b> from the Tools menu. Click the Pan button  in the toolbar or select <b>Pan</b> from the Tools menu. If you click Maintain fit to window button  in the toolbar or select <b>Maintain fit to window</b> or from the Tools menu, the zoom and pan buttons are disabled.	
Examine an area of the current frame in detail.	Click the Pixel region button  in the Playback toolbar or select <b>Pixel Region</b> from the Tools menu.	
Export frame to Image Tool	Click the Export to Image tool button  in the Playback toolbar or select <b>Export to Image Tool</b> from the File menu. The Movie Player opens an Image Tool containing the current frame.	

برای کسب اطلاعات در مورد فریمها روی آیکن اطلاعات ( یعنی  ) کلیک کنید.  
 برای تغییر نرخ نمایش فریم (تعداد فریمها در ثانیه<sup>۲۵</sup>) از منوی Playback گزینهی Frame Rate را انتخاب کنید یا اینکه کلید T را فشار دهید:



برای اعمال یک ماتریس رنگ به تصاویر سطح خاکستری از منوی Tools گزینهی Colormap را انتخاب کنید یا اینکه دکمه‌ی C را فشار دهید.



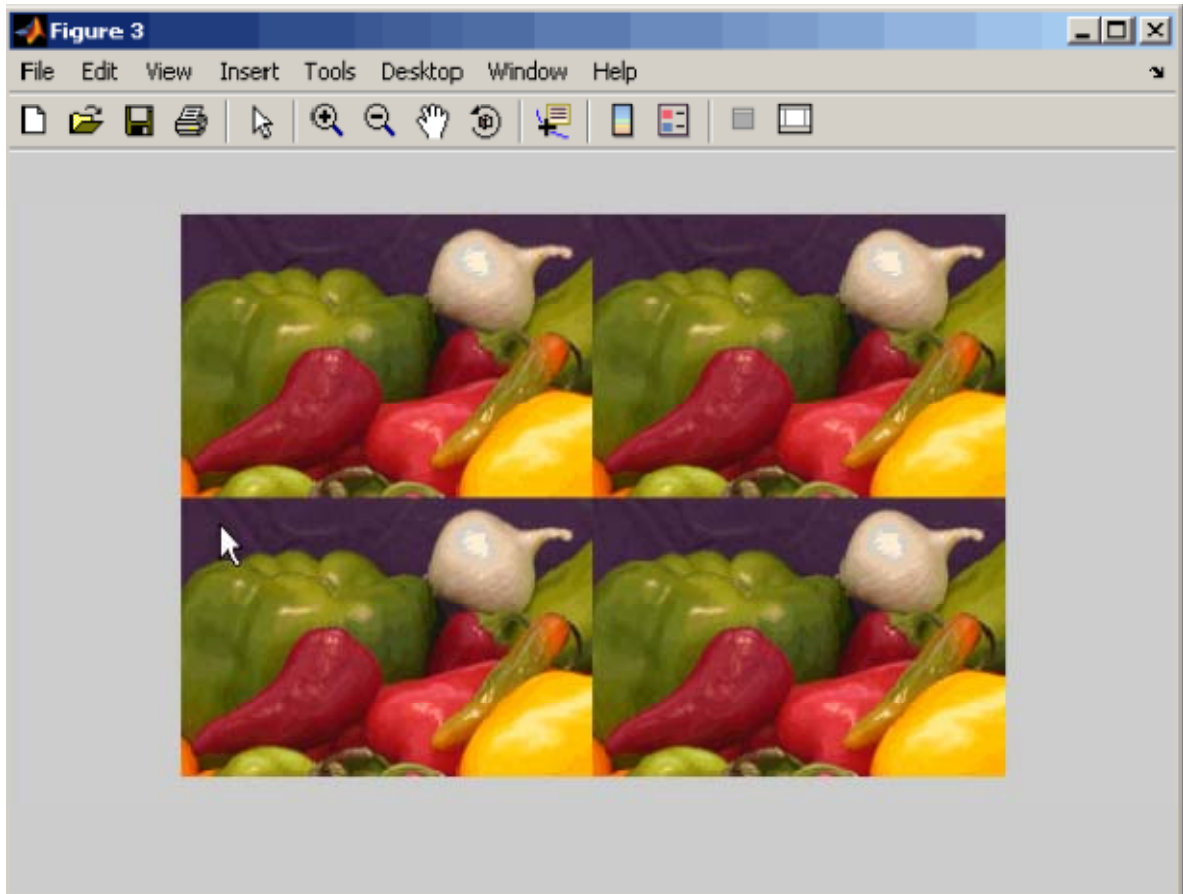
هرگاه بخواهید تصاویر موجود در یک آرایه چند بعدی را یکجا کنار هم داخل یک پنجره ببینید از دستور montage استفاده کنید.

```
onion = imread('onion.png');
```

```
onionArray = repmat(onion, [ 1 1 1 4 ]);
```

```
montage(onionArray);
```

دستور `repmat(A,m,n,p,q)` از روی ماتریس A یک ماتریس بزرگتر می‌سازد که در جهت بعد اول، A را m بار کپی می‌کند، در جهت بعد دوم، A را n بار کپی می‌کند، در جهت بعد سوم، A را p بار کپی می‌کند، و بالاخره در جهت بعد چهارم، A را q بار کپی می‌کند. تعداد پارامترهای m و n و... دلخواه است. دستور `repmat` در مثال فوق، یک آرایه‌ی چهار بعدی می‌سازد (یعنی یک آرایه چند بعدی) تا به عنوان دنباله‌ای از تصاویر محسوب شود.



### تبدیل یک تصویر چندفریمی به فیلم

از تابع `immovie` میتوان برای تبدیل یک تصویر چند فریمی به فیلم استفاده کرد. مثلاً برای تبدیل یک تصویر چندفریمی اندیس گذاری شده به فیلم به صورت زیر عمل کنید:

```
mov = immovie(X,map);
```

حال از دستور `implay` برای پخش فیلم استفاده کنید:

```
Implay(mov);
```

مثال زیر یک فایل `tif` به نام `mri.tif` را خوانده و در یک تصویر چندفریمی (همان آرایه چندبعدی) ذخیره می‌کند. سپس این تصویر چندفریمی به فیلم تبدیل شده و در انتها پخش می‌شود:

```
mri = uint8(zeros(128,128,1,27));
for frame=1:27
[mri(:,:,,frame),map] = imread('mri.tif',frame);
end
mov = immovie(mri,map);
implay(mov);
```

برای تولید فیلمی که در خارج از متلب هم قابل پخش باشد باید فیلمی از نوع `avi` بسازید. دو راه برای این کار وجود دارد:

۱- از دستورات `avifile` و `addframe` استفاده کنید.

۲- یا اینکه از دستور `movie2avi` استفاده کنید.

توجه کنید که می‌توانید به کمک تصاویر اندیس گذاری شده و یا RGB که از نوع uint8 و یا double باشند، یک فیلم avi بسازید.

### نمایش تصاویر اندیس گذاری شده

یک تصویر اندیس گذاری شده شامل یک ماتریس تصویر و یک ماتریس رنگ است. عناصر ماتریس تصویر در حقیقت اشاره گرهایی به درون ماتریس رنگ می‌باشند. برای نمایش چنین تصاویری باید هر دو ماتریس را در دستور imshow و یا imtool ذکر کنیم:

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

```
imtool(X,map)
```

اگر نوع عناصر ماتریس تصویر برابر double باشد، مقدار ۱ در ماتریس تصویر به اولین سطر ماتریس رنگ، مقدار ۲ به دومین سطر و همین طور الی آخر اشاره دارد. اما اگر نوع عناصر ماتریس تصویر برابر uint8 و یا uint16 باشد، مقدار ۰ به اولین سطر ماتریس رنگ، مقدار ۱ به دومین سطر و همین طور الی آخر اشاره دارد. اگر در ماتریس تصویر عنصری بزرگتر از تعداد سطرهای ماتریس رنگ وجود داشته باشد (یعنی سطر معادل آن وجود نداشته باشد)، از آخرین سطر ماتریس رنگ استفاده می‌شود.

### نمایش تصویری که مقادیر آن خارج از بازه معمول است

فرض کنید تصویری دارید که مقادیر آن خارج از بازه متداول است؛ یعنی:

۱- بازه [0,1] برای نوعهای single و double

۲- بازه [0,255] برای نوع uint8

۳- بازه [0,65535] برای نوع uint16

۴- بازه [-32768,32767] برای نوع int16

چنین تصویر می‌ممکن است برای مثال، حاصل فیلتر کردن یک تصویر نرمال (یعنی با مقادیر در بازه معمول) باشد. برای نمایش چنین تصاویری، بازه تغییرات را در دستور imshow و یا دستور imtool ذکر کنید. به صورت زیر:

```
imshow(I,'DisplayRange',[low high])
```

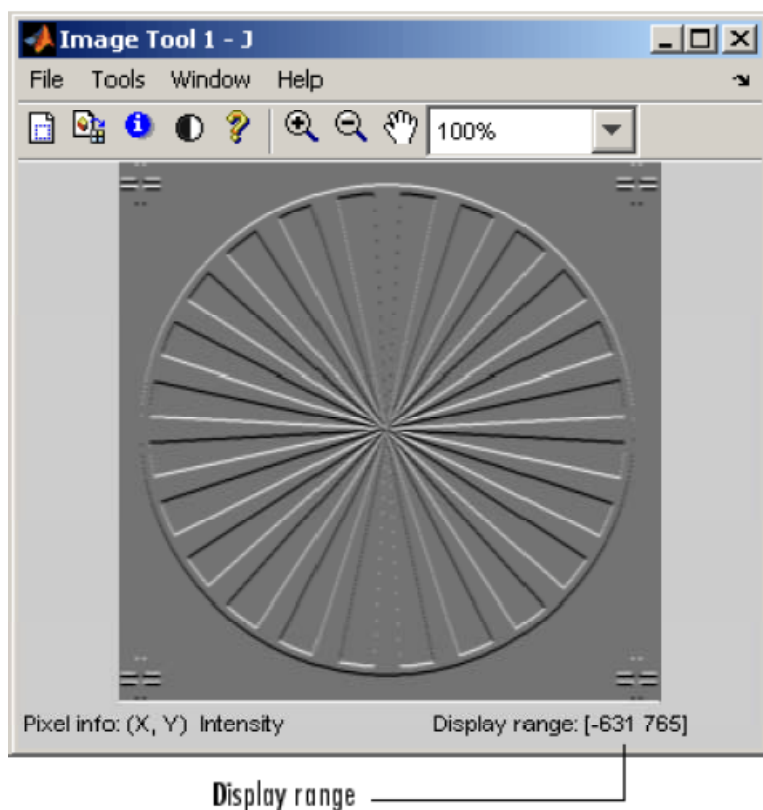
```
imtool(I,'DisplayRange',[low high])
```

اگر محدوده بالا و پایین را ذکر نکنید و فقط از [] استفاده کنید، محدوده مناسب به طور خودکار تعیین می‌شود. به مثال زیر توجه کنید (در شکا حاصل توجه کنید که محدوده تغییرات تصویر ذکر شده است):

```
I = imread('testpat1.png');
```

```
J = filter2([1 2;-1 -2],I);
```

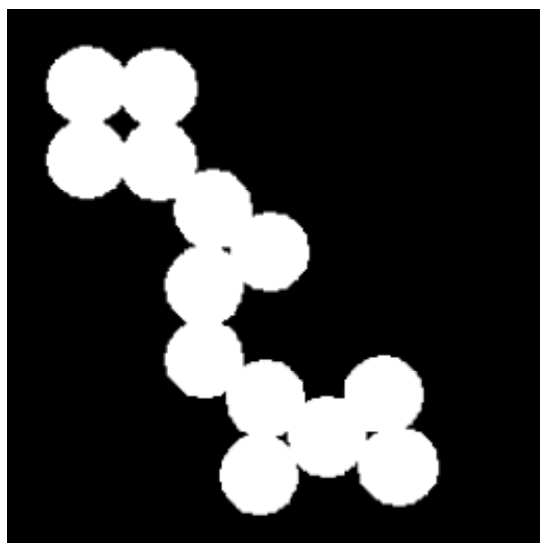
```
imtool(J,'DisplayRange',[]);
```



### نمایش تصاویر باینری

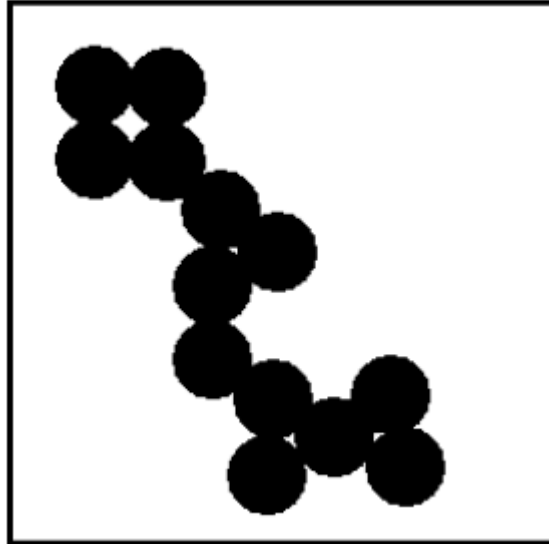
تصاویر باینری از نوع logical هستند. اگر می‌خواهید که تصویرتان به عنوان یک تصویر باینری توسط توابع جعبه ابزار پردازش تصویر تلقی شود، باید عناصر ماتریس تصویر از نوع logical تعریف شده باشند. مثالی از خواندن یک تصویر باینری:

```
BW = imread('circles.png');
imshow(BW)
or
imtool(BW)
```



اگر می‌خواهید جای رنگهای سیاه و سفید تصویر باینری نمای ش داده شده عوض شود، از علامت ~ قبل از نامه تصویر استفاده کنید. این علامت، همان علامت نقیض (یا متمم) است که مقدار ۰ را به ۱ و مقدار ۱ را به ۰ تبدیل می‌کند:

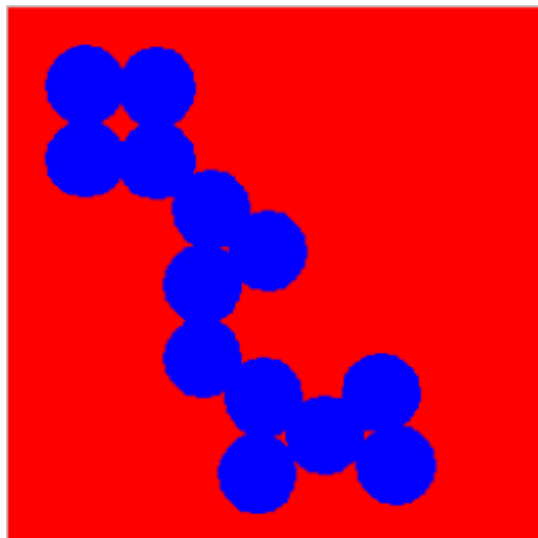
```
imshow(~BW)
or
imtool(~BW)
```



همچنین می‌توانید با آدرس‌دهی به شیوه‌ی تصاویر اندیس گذاری شده، یک تصویر باینری را به صورت رنگی نمایش دهید. برای مثال، دستور زیر برای رنگ سیاه از رنگ قرمز و به جای رنگ سفید از رنگ آبی استفاده می‌کند:

```
imshow(BW,[1 0 0; 0 0 1])
or
imtool(BW,[1 0 0; 0 0 1])
```

نتیجه:



## نمایش تصاویر رنگی

از دستور `imshow` استفاده می‌شود.

```
RGB = imread('peppers.png');
imshow(RGB)
or
imtool(RGB)
```



نکته: اگر یک تصویر رنگی را نمایش دادید اما به صورت سطح خاکستری نمایش داده شد، آن تصویر به صورت اندیس‌گذاری شده بوده و شما باید از ماتریس رنگ نیز در کنار ماتریس تصویر استفاده کنید.

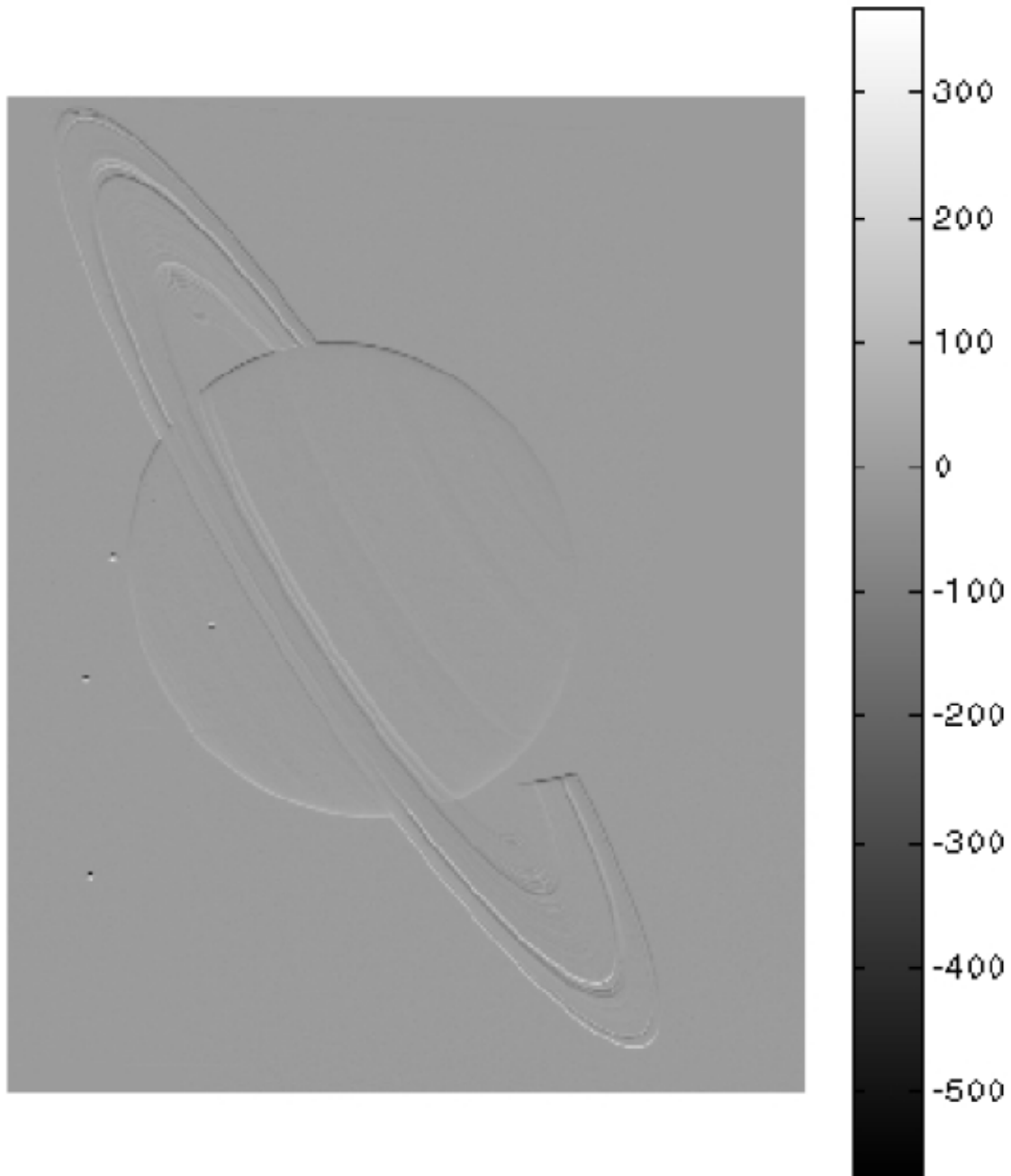
## نمایش میله رنگ

هدف از یک میله‌ی رنگ<sup>۲۶</sup> نشان دادن محدوده سطوح مختلف روشنایی و نیز رنگ متناظر با هر سطح روشنایی در نمایش تصویر است. برای اضافه کردن یک میله رنگ باید ابتدا از دستور `imshow` برای نمایش تصویر استفاده کنید و سپس از دستور `colorbar` استفاده کنید.

یکی از موارد استفاده از میله رنگ، زمانی است که تصویری در اختیار دارید که بنا به دلایل مختلف محدوده‌ی مقادیری خارج از محدوده معمول (مثلاً خارج از بازه صفر تا ۲۵۵) دارد. در مثال زیر، تصویر ورودی فیلتر شده و بنابراین، مقادیر پیکسل‌های تصویر خروجی خارج از بازه‌ی نوع `uint8` است.

```
RGB = imread('saturn.png');  
I = rgb2gray(RGB);  
h = [1 2 1; 0 0 0; -1 -2 -1];  
I2 = filter2(h,I);  
imshow(I2,'DisplayRange',[]), colorbar
```

نتیجه:



### چاپ تصویر

برای چاپ یک تصویر از دستور `print` می‌توانید استفاده کنید (علاوه بر استفاده از منوهای استاندارد موجود).