

图像的读取和展示

图像在MATLAB中的存储格式

读取和展示图像

图像的运算

图像的点运算

图像的四则运算

像素的统计分布

图像的二值化

图像的几何变换

使用MATLAB分析图像:目标计数

图像预处理

目标计数:标记连通区域

分析检测结果

学习一门技术最好的方式就是阅读官方文档,可以查看[MATLAB官方文档](#)

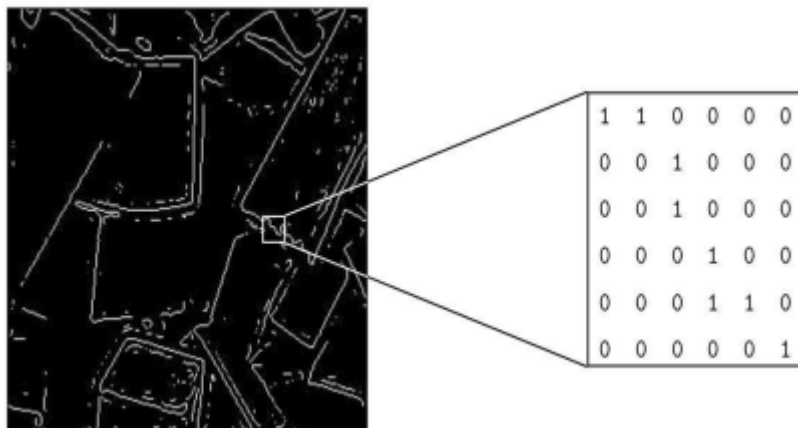
图像的读取和展示

图像在MATLAB中的存储格式

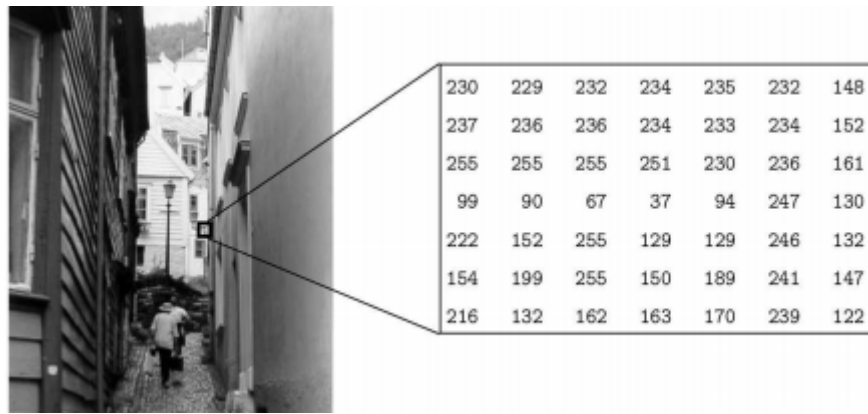
MATLAB能够处理的数字图像分为三种:二值图像,灰度图像,彩色图像.



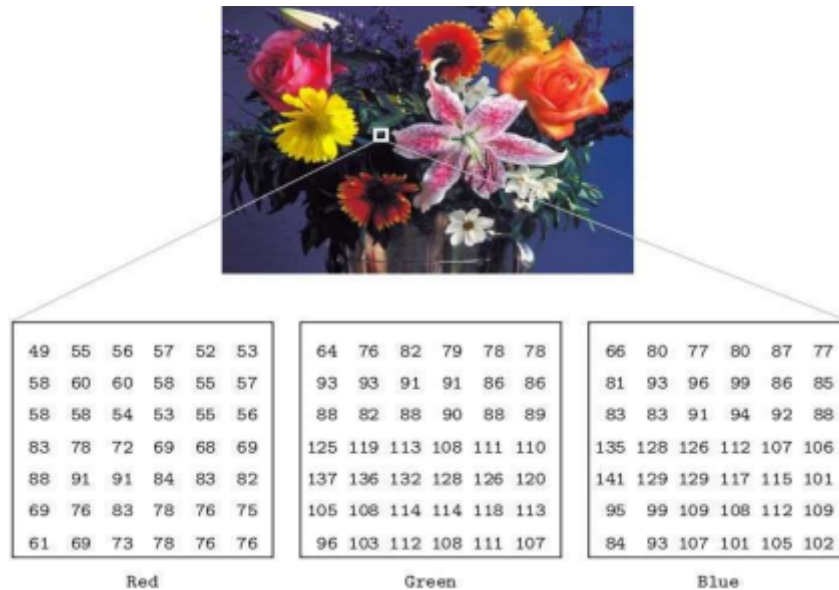
- **二值图像**在MATLAB中以一个矩阵存储,矩阵中元素的取值为0(表示白)或1(表示黑).



- **灰度图像**在MATLAB中以一个矩阵存储,矩阵中元素的取值介于0~255之间,表示灰度.



- 彩色图像在MATLAB中以三个矩阵存储,每个矩阵中元素的取值介于0~255之间,分别表示颜色R,G,B分量的浓度



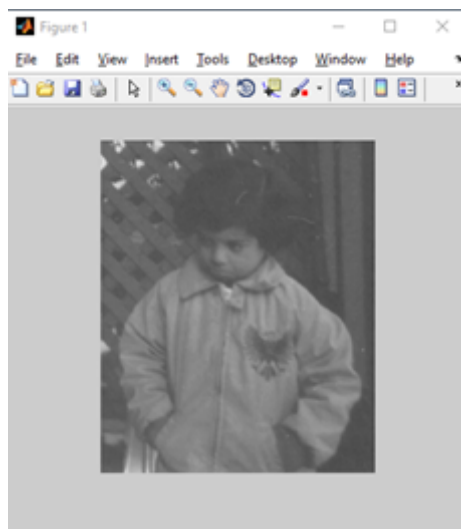
读取和展示图像

使用 `imread()` 函数将图像读取到内存中,使用 `imshow()` 函数展示图像,使用 `imwrite()` 函数将内存中的图像写进硬盘.

```

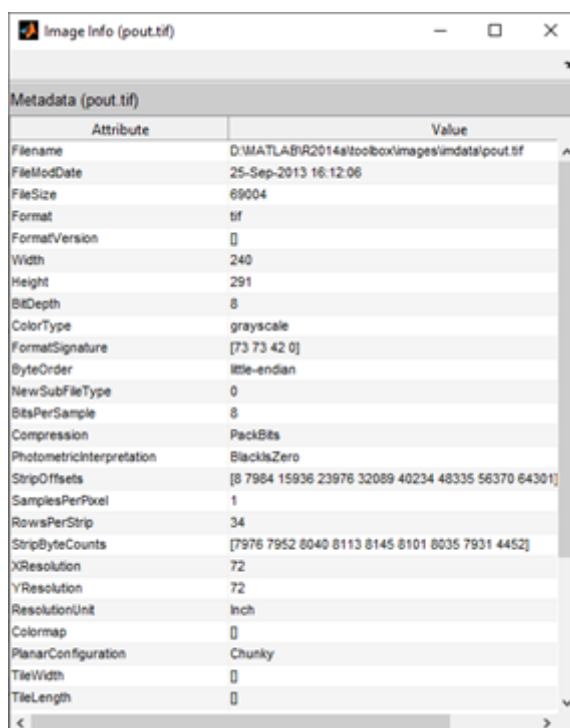
1 clear, close all
2 I = imread('pout.tif');      % 将MATLAB自带图像'pout.tif'读取到内存中
3 imshow(I);                  % 在图形窗口展示该图像
4 imwrite(I,'myimage.png');   % 将该图像存为png格式的文件

```



使用 `imageinfo()` 函数查看图片文件的详细信息.

```
1 | imageinfo('pout.tif')
```

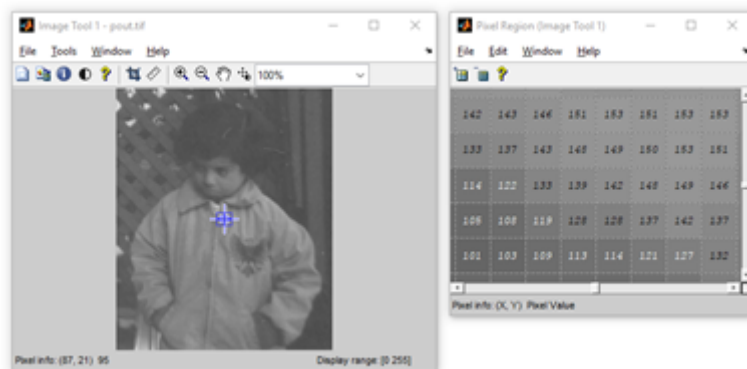


The screenshot shows a window titled 'Image Info (pout.tif)' with a table of metadata. The table has two columns: 'Attribute' and 'Value'.

Attribute	Value
Filename	D:\MATLAB\R2014a\toolbox\images\imdata\pout.tif
FileModDate	25-Sep-2013 16:12:06
FileSize	69004
Format	tif
FormatVersion	1
Width	240
Height	291
BitDepth	8
ColorType	grayscale
FormatSignature	[73 73 42 0]
ByteOrder	little-endian
NewSubFileType	0
BitsPerSample	8
Compression	PackBits
PhotometricInterpretation	BlackIsZero
StripOffsets	[8 7984 15936 23976 32089 40234 48335 56370 64301]
SamplesPerPixel	1
RowsPerStrip	34
StripByteCounts	[7976 7952 8040 8113 8145 8101 8035 7931 4452]
XResolution	72
YResolution	72
ResolutionUnit	Inch
Colormap	[]
PlanarConfiguration	Chunky
TileWidth	0
TileLength	0

使用 `imtool()` 函数可以打开图像处理工具.

```
1 | imtool('pout.tif')
```



图像的运算

图像的点运算

图像在内存中以矩阵的形式存储,因此我们可以像遍历矩阵那样遍历并编辑图片上的像素点.MATLAB也内置了一些函数用于进行图像运算.

图像的四则运算

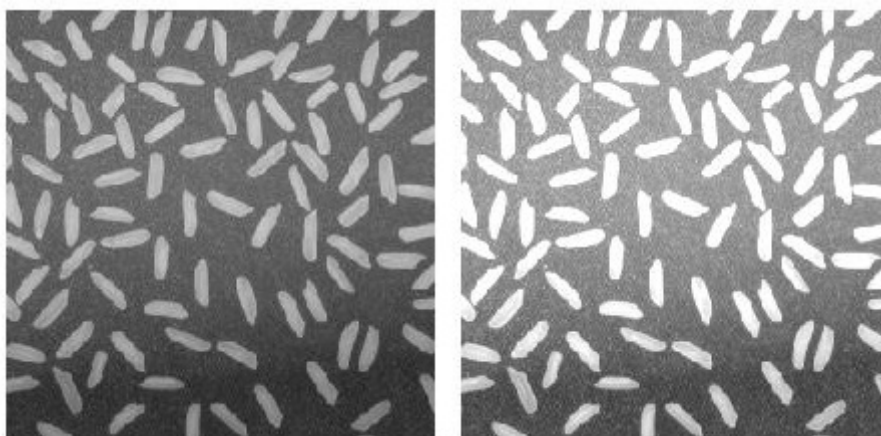
要对两个图像进行四则运算,要求这两个图像的尺寸相同.下面是常用的图像四则运算函数,具体细节请参考[官方文档](#).

函数	作用
<code>imabsdiff()</code>	两个图像求差值
<code>imadd()</code>	一个图像加上另一个图像或常数
<code>imsubtract()</code>	一个图像减去另一个图像或常数
<code>immultiply()</code>	一个图像乘以另一个图像或常数
<code>imdivide()</code>	一个图像除以另一个图像或常数
<code>imcomplement()</code>	对图像取反

```

1 I=imread('rice.png');
2 subplot(1,2,1); imshow(I);
3 J=immultiply(I, 1.5);
4 subplot(1,2,2); imshow(J);

```



```

1 I=imread('rice.png'); J=imread('cameraman.tif');
2 K=imadd(I,J);
3 subplot(1,3,1); imshow(I);
4 subplot(1,3,2); imshow(K);
5 subplot(1,3,3); imshow(J);

```



可以看到,进行加法操作后,得到的图像比原本的两个都亮,这是因为图像矩阵的数值整体上增加了。

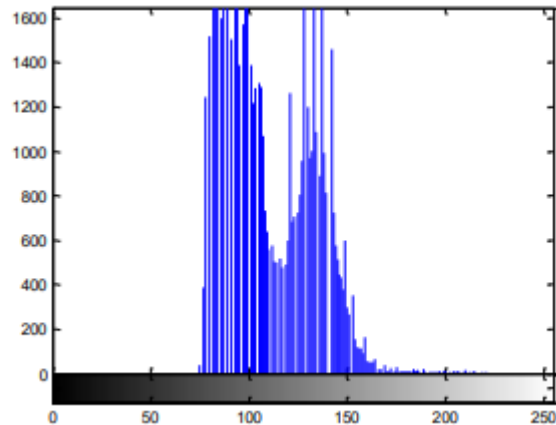
像素的统计分布

使用 `imhist()` 函数可以分析像素值的统计分布。

```

1 I = imread('pout.tif');
2 imhist(I)

```

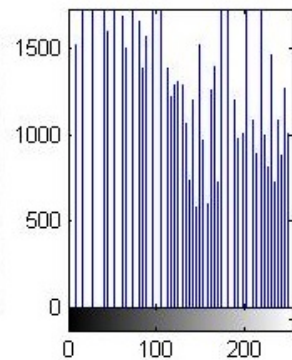
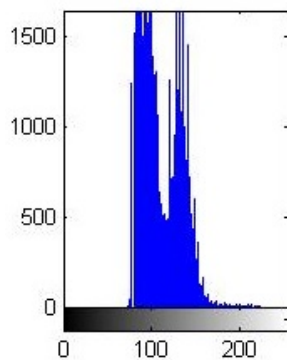


使用 `histeq()` 可以增大图像的对比度,这本质上做了直方图的均衡化(Histogram Equalization)操作.

```

1 I = imread('pout.tif'); I2 = histeq(I);
2 subplot(1,4,1); imhist(I);
3 subplot(1,4,2); imshow(I);
4 subplot(1,4,3); imshow(I2);
5 subplot(1,4,4); imhist(I2);

```



图像的二值化

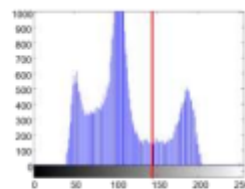
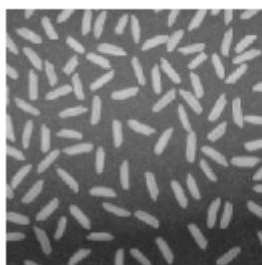
将灰度图像变为二值图像的过程被称为**二值化**,MATLAB内置了两个与二值化相关的函数.

- `graythresh()` 函数用于计算二值化变换过程中的最优阈值(threshold).灰度图像上超过该阈值的点将被赋值为1,低于该阈值的点将被赋值为0.
- `im2bw()` 用于进行二值化变换.

```

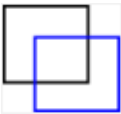
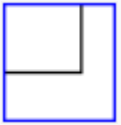
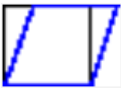
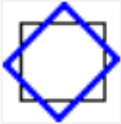
1 I = imread('rice.png');
2 level=graythresh(I); bw=im2bw(I, level);
3 subplot(1,2,1); imshow(I);
4 subplot (1,2,2); imshow(bw)

```

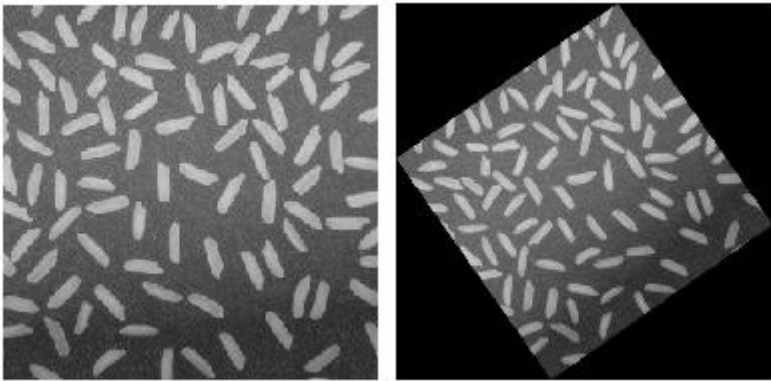


图像的几何变换

图像的几何变换本质上就是将图像乘以一个矩阵得到新图像的过程.

变换形式	图形示意	数学变换	MATLA命令
位移 (Translation)		$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$	<code>imtranslate()</code>
缩放(Scale)		$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$	<code>imresize()</code>
错切(Shear)		$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & h_x & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$	
旋转(Rotate)		$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$	<code>imrotate()</code>

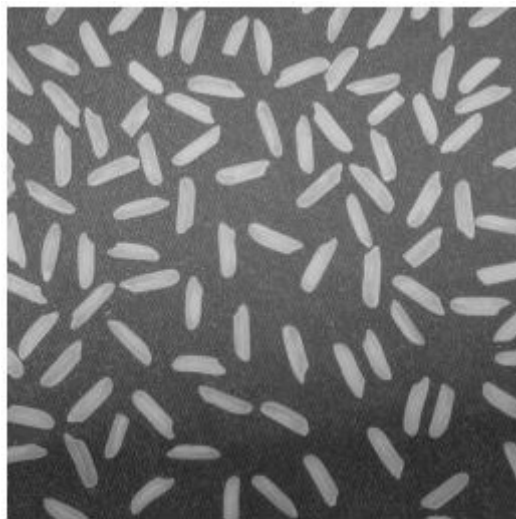
```
1 I = imread('rice.png'); J = imrotate(I, 35, 'bilinear');
2 subplot(1,2,1); imshow(I);
3 subplot(1,2,2); imshow(J);
4 size(I) % 得到 [256, 256]
5 size(J) % 得到 [357, 357]
```



可以看到,进行旋转变换后,图像的尺寸增加了.

使用MATLAB分析图像:目标计数

我们想要通过MATLAB分析 `rice.png` 图片中米粒的个数.



图像预处理

要分析图像中的米粒个数,我们需要对图像进行两步预处理:

1. 去除图像的背景:

```
1 I = imread('rice.png');
2 subplot(1,3,1); imshow(I);
3 BG = imopen(I, strel('disk', 15));
4 subplot(1,3,2); imshow(BG);
5 I2 = imsubtract(I, BG);
6 subplot(1,3,3); imshow(I2);
```

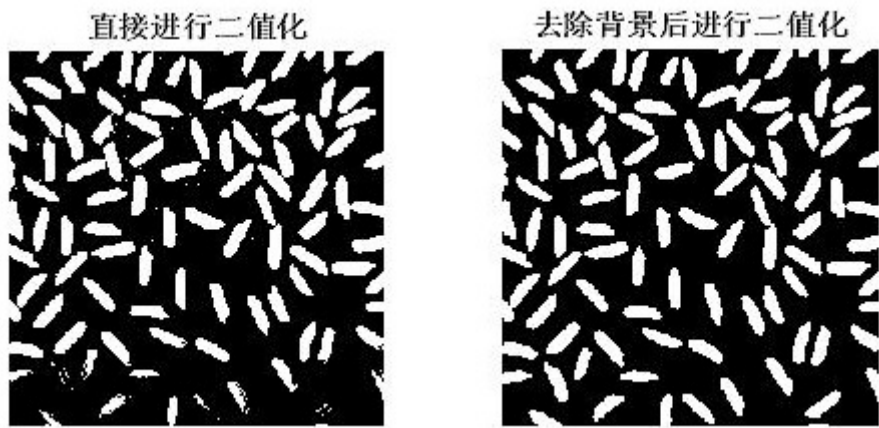


2. 对图像进行二值化:

```
1 I2 = imsubtract(I, BG); level=graythresh(I2);
2 bw2 = im2bw(I2, level);
```

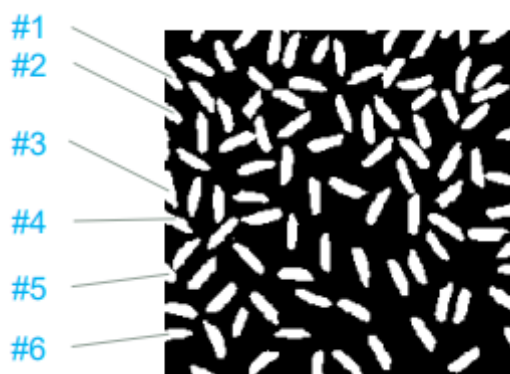
下面代码展示了是否去除背景对图像二值化结果的影响:

```
1 % 直接对图像进行二值化
2 I = imread('rice.png');
3 level=graythresh(I); bw = im2bw(I, level);
4 subplot(1,2,1); imshow(bw); title('直接进行二值化');
5
6 % 去除背景后对图像进行二值化
7 BG = imopen(I, strel('disk', 15)); I2 = imsubtract(I, BG);
8 level=graythresh(I2); bw2 = im2bw(I2, level);
9 subplot(1,2,2); imshow(bw2); title('去除背景后进行二值化');
```



目标计数:标记连通区域

识别米粒个数的关键在于识别连通区域.



在这里,我们使用MATLAB自带的 `bwlabel()` 函数计算连通区域,该函数使用了连通区域标记算法,将每个连通区域内的像素点赋值为同一个值.

Binary image								Label matrix							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	2	2	2	0	0
0	0	0	0	0	0	0	0	0	1	0	0	2	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```

1 I=imread('rice.png');
2 BG=imopen(I, strel('disk', 15));
3 I2=imsubtract(I, BG); level=graythresh(I2);
4 BW=im2bw(I2, level);
5 [labeled, numObjects]=bwlabel(BW, 8);

```

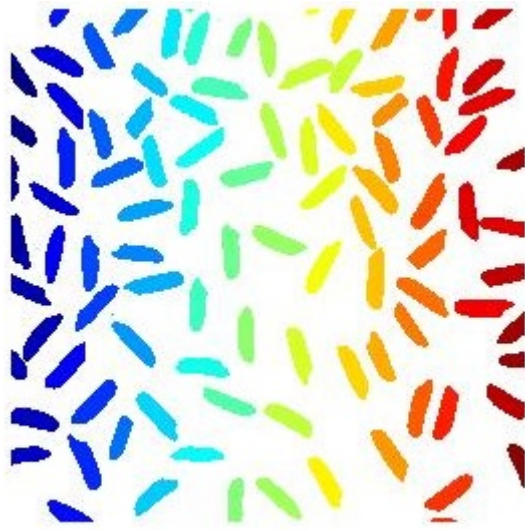
得到 `labeled` 为标记好的矩阵,其尺寸与原图片相同,每个连通区域都被赋值为一个相同的整数,其他区域被赋值为0. `numObjects` 为计算出的连通区域个数,为99.

使用 `label2rgb()` 函数可以将标记结果以彩色图片的形式展示

```

1 RGB_label=label2rgb(labeled);
2 imshow(RGB_label);

```

分析检测结果

使用 `regionprops()` 函数可以将检测结果封装成结构体数组。

```
1 graindata = regionprops(labeled, 'basic');  
2 graindata(51)
```

```
1         Area: 155  
2     Centroid: [112.4258 245.8645]  
3 BoundingBox: [108.5000 234.5000 8 22]
```

使用 `bwselect()` 函数可以交互式选择连通区域

```
1 ObjI = bwselect(BW);  
2 imshow(ObjI);
```

