MATLAB

Lab2

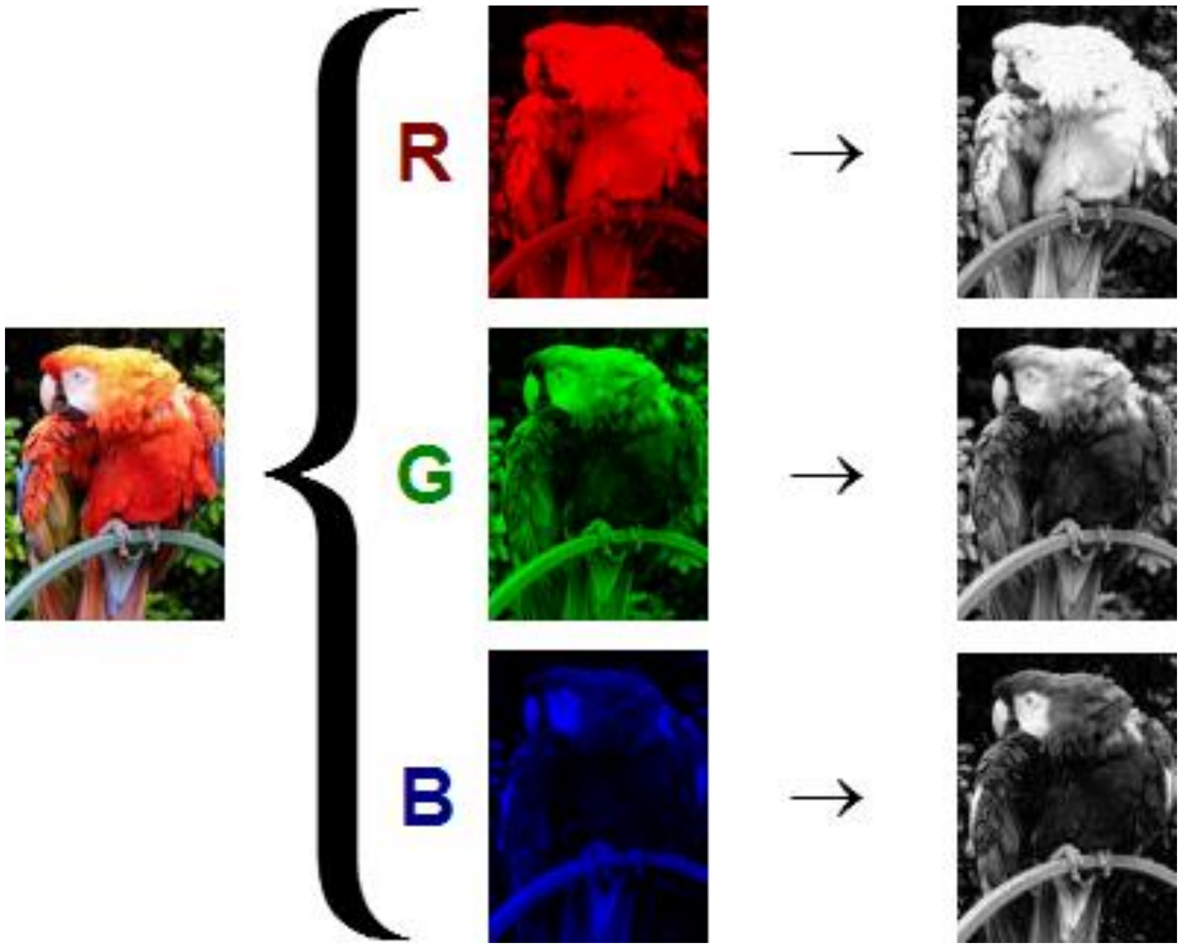
Blobing | Morphology | Filtering | Color space

Acquisition of digital image

Types of digital image

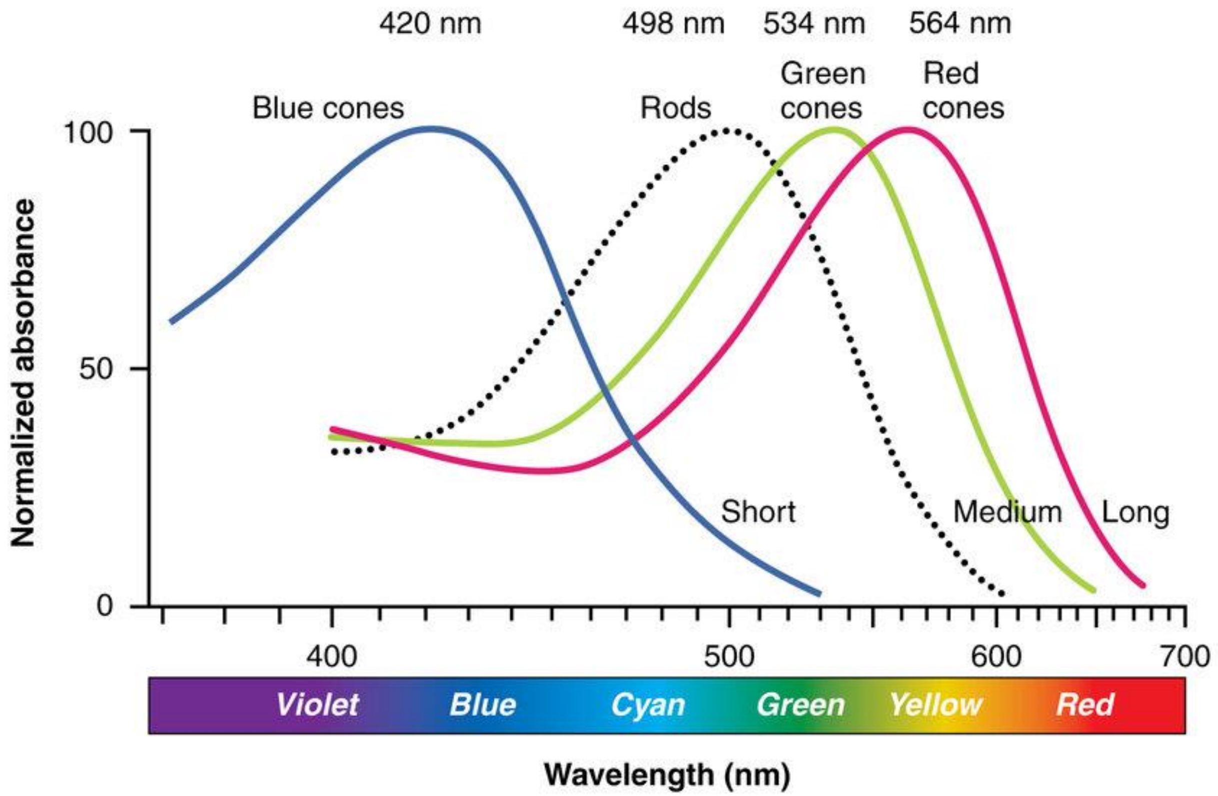
- Binary
  - Each pixel is just black or white
- Grayscale
  - Each pixel is a shade of gray, normally from 0 (black) to 255 (white)
- True color or RGB
  - Each pixel has a particular color described by the amount of red, green and blue in it

Typical RGB image

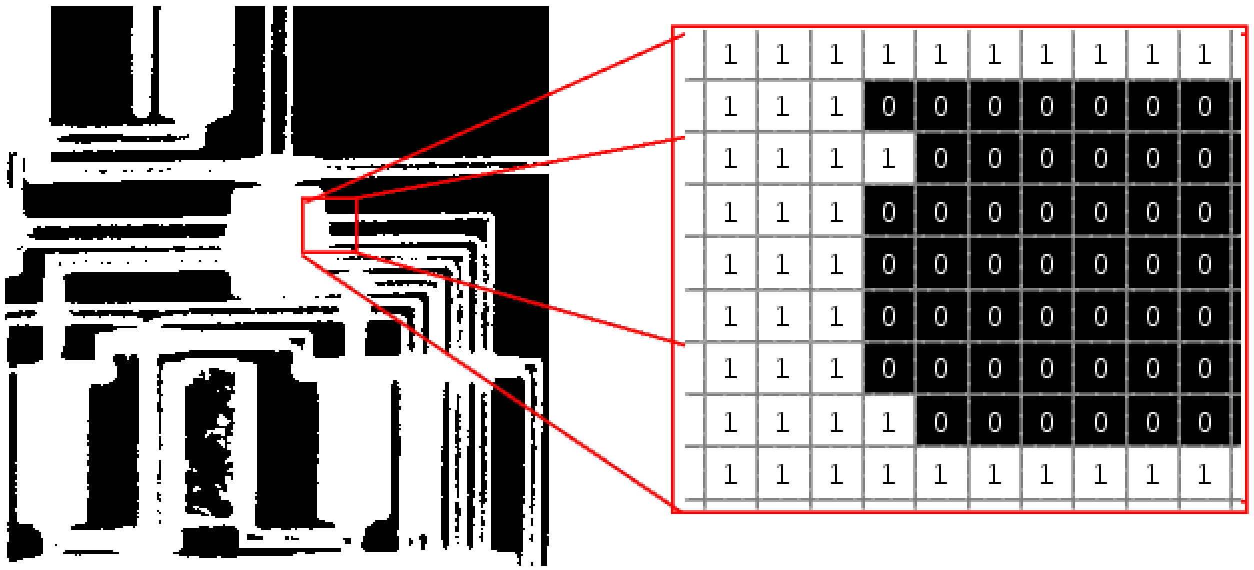


Why RGB?

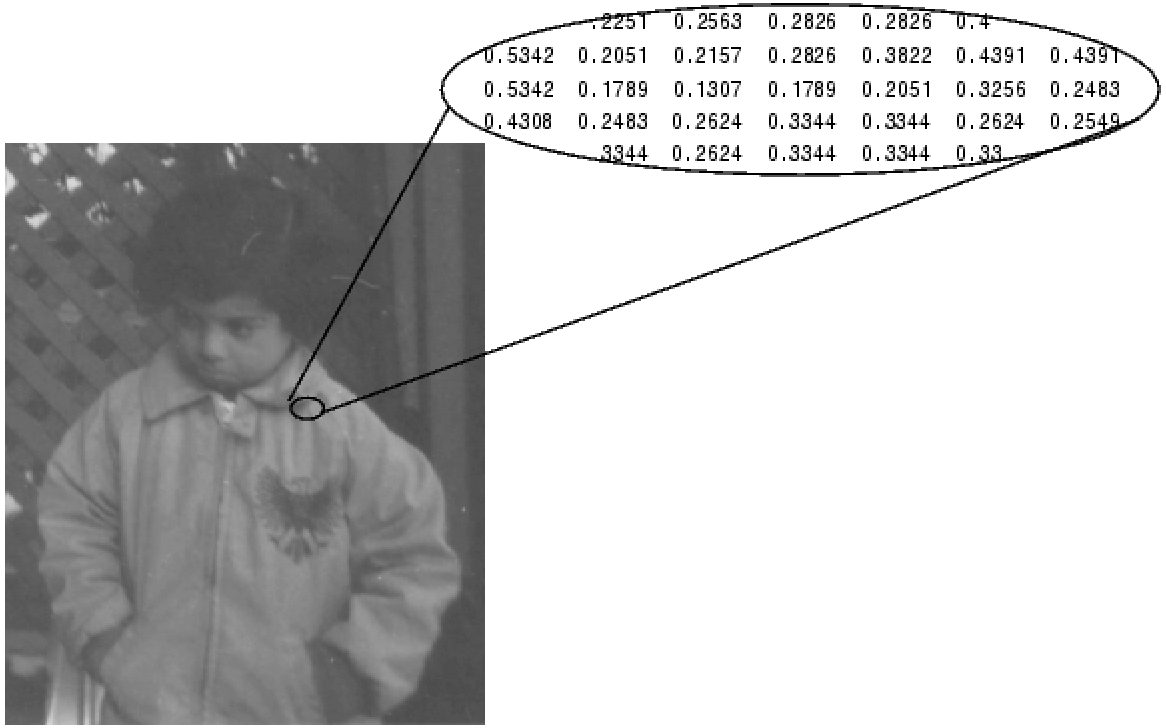
- Three kinds of light-sensitive photoreceptor cells in the human eye (i.e., cone cells) respond most to red, green and blue



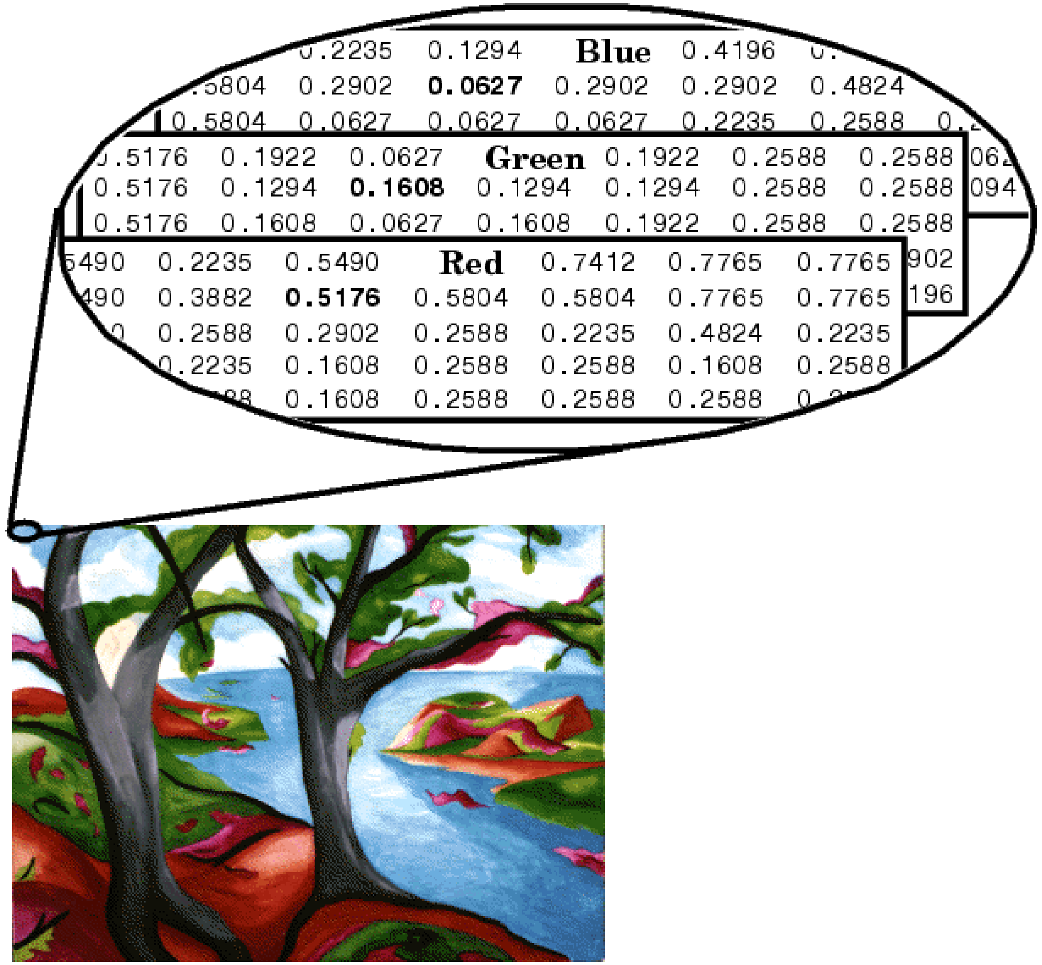
Binary image



Greyscale image



Color image



Blue
0.2235 0.1294 0.4196
0.5804 0.2902 0.0627 0.2902 0.2902 0.4824
0.5804 0.0627 0.0627 0.0627 0.2235 0.2588
Green
0.5176 0.1922 0.0627 0.1922 0.2588 0.2588
0.5176 0.1294 0.1608 0.1294 0.1294 0.2588 0.2588
0.5176 0.1608 0.0627 0.1608 0.1922 0.2588 0.2588
Red
0.2235 0.5490 0.7412 0.7765 0.7765
0.3882 0.5176 0.5804 0.5804 0.7765 0.7765
0.2588 0.2902 0.2588 0.2235 0.4824 0.2235
0.2235 0.1608 0.2588 0.2588 0.1608 0.2588
0.1608 0.2588 0.2588 0.2588

Read and show an image

- Read an image: `imread()`
- Show an image: `imshow()`
- Example:

```
clear, close all
I = imread('pout.tif'); % read
imshow(I); % show
```

Image variable in workspace

```
whos()
```

```
Name          Size      Bytes   Class
   I       291x240      69840   uint8
```

- Image matrix

291x240 uint8

	1	2	3	4	5
1	107	108	107	106	99
2	109	106	108	107	103
3	107	106	110	110	106
4	106	107	108	108	108
5	105	108	109	109	108
6	105	108	109	110	108
7	108	109	109	109	108
8	107	107	108	108	107
9	108	109	108	107	106
10	106	107	107	107	107
11	109	109	107	108	108
12	123	109	108	108	107
13	164	121	112	110	109

c

r

Image information

- Get image information

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

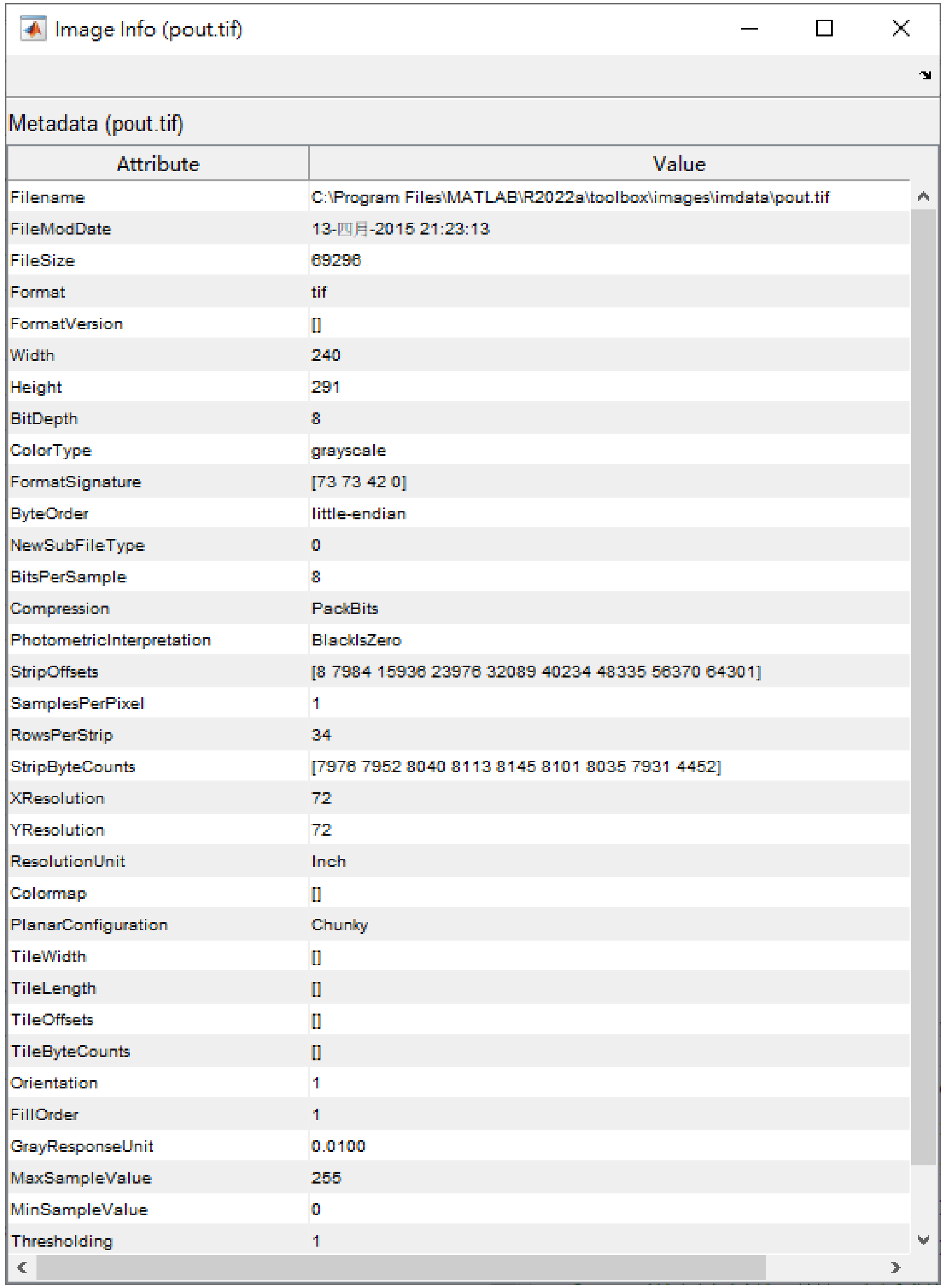


Image Info (pout.tif)

Metadata (pout.tif)

Attribute	Value
Filename	C:\Program Files\MATLAB\R2022a\toolbox\images\imdata\pout.tif
FileModDate	13-四月-2015 21:23:13
FileSize	69296
Format	tif
FormatVersion	[]
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	[]
TileLength	[]
TileOffsets	[]
TileByteCounts	[]
Orientation	1
FillOrder	1
GrayResponseUnit	0.0100
MaxSampleValue	255
MinSampleValue	0
Thresholding	1

Image viewer

- Get pixel information

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

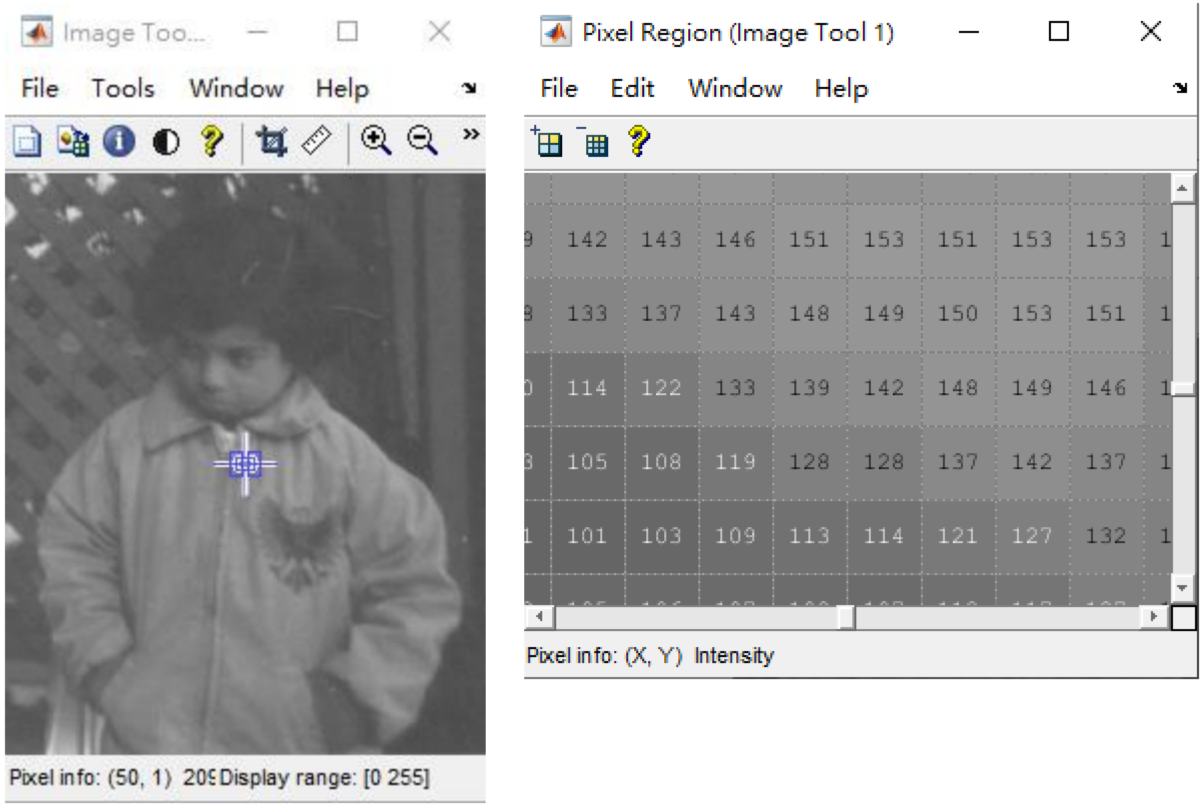


Image histogram

```
imhist(I)
```

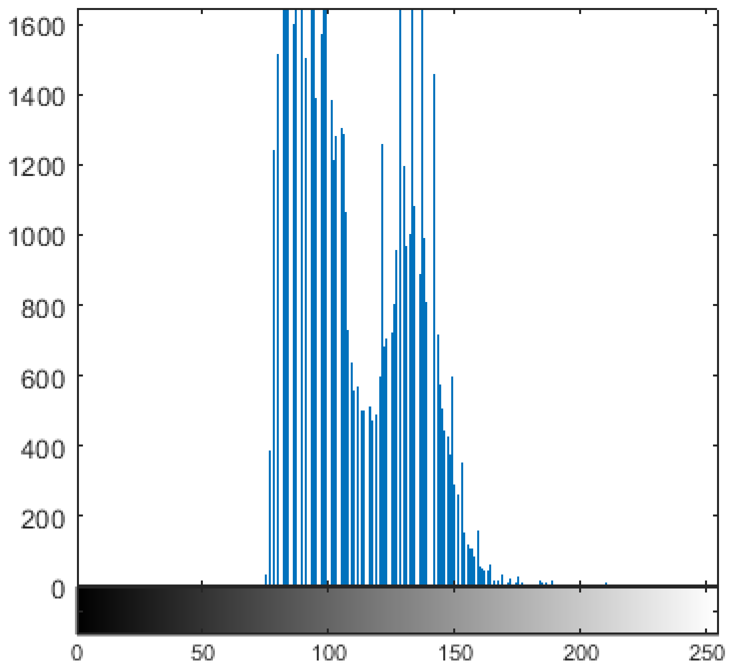


Image processing

- Any form of signal processing for which the input is an image

(a) PSNR-51.55, SSIM-0.9986, IEF-387 | (b) PSNR-40.04, SSIM-0.9847, IEF-283 | (c) PSNR-33.03, SSIM-0.9409, IEF-172

Histogram equalization

- Enhances the contrast of the image

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

Image arithmetic

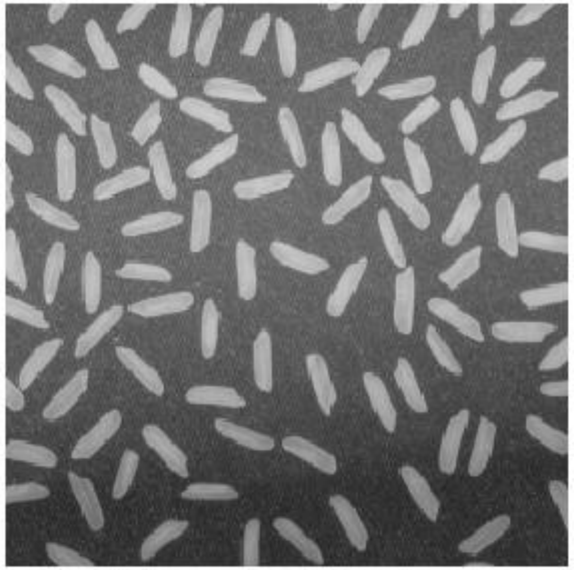
`imabsdiff`	Absolute difference of two images
`imadd`	Add two images or add constant to image
`imapplymatrix`	Linear combination of color channels
`imconplement`	Complement image
`imdevide`	Divide one image into another or divide image by constant
`imlincomb`	Linear combination of images
`immultiply`	Multiply two images or multiply image by constant
`imsubtract`	Subtract one image from another or subtract constant from image

Image addition

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

Image multiplication

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



Spatial transformations

- Translation

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

- Scale
  - `imresize()`

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Shear

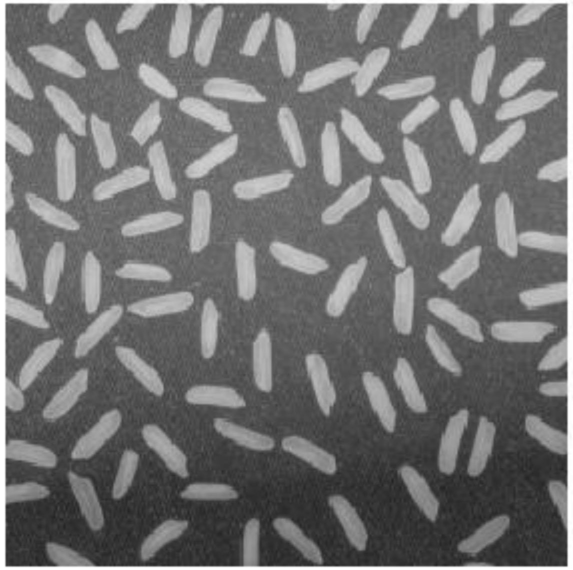
$$\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

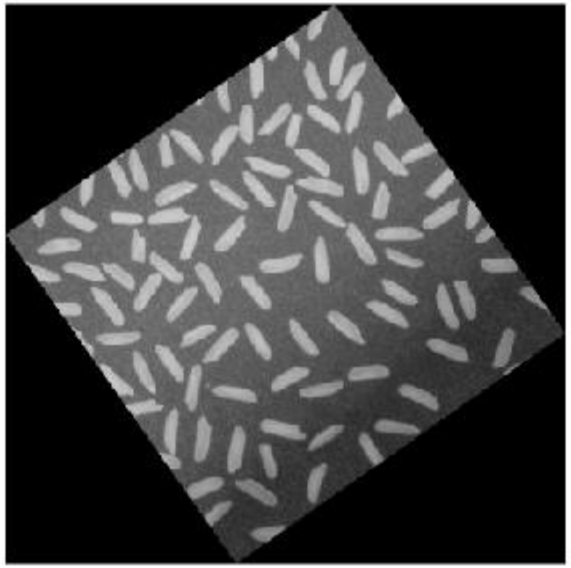
- Rotation
  - `imrotate()`

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Image rotation

```
I = imread('rice.png');
subplot(1,2,1); imshow(I);
J = imrotate(I, 35, 'bilinear');
subplot(1,2,2); imshow(J);
size(I)
size(J)
```





Write image

- Format supported:
  - 'bmp', 'gif', 'hdf', 'jpg', 'jpeg', 'jp2', 'jpx', 'pcx', 'pnm', 'ppm', 'ras', 'tif', 'tiff', 'xwd'

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

Problem setup

- Count the rice grains and identify their sizes in this image
- What are your strategies?

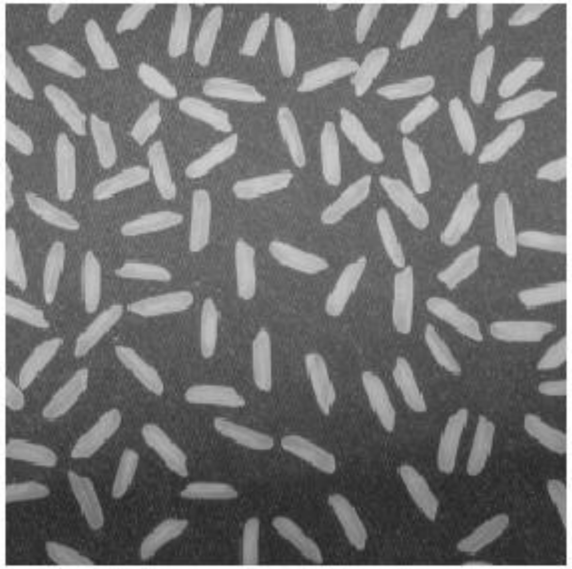
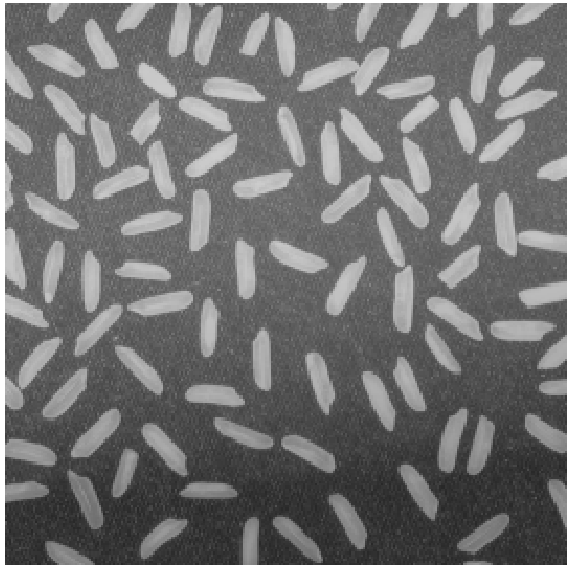
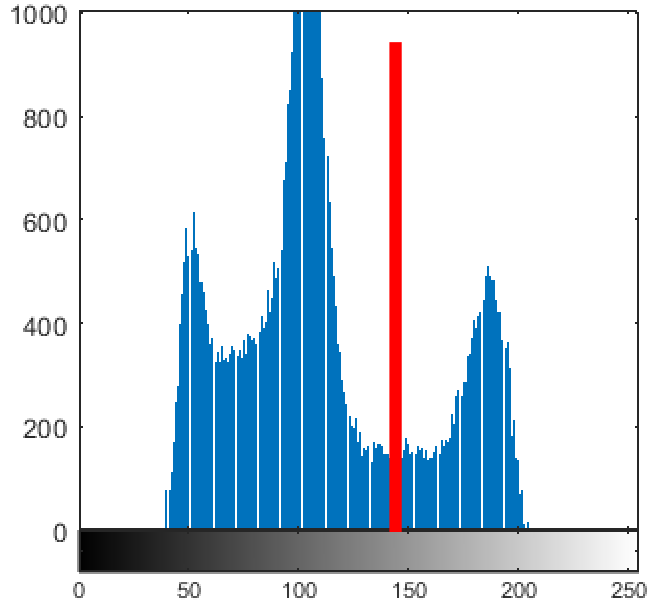


Image thresholding

- A gray-level image can be turned into a binary image by using a threshold

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

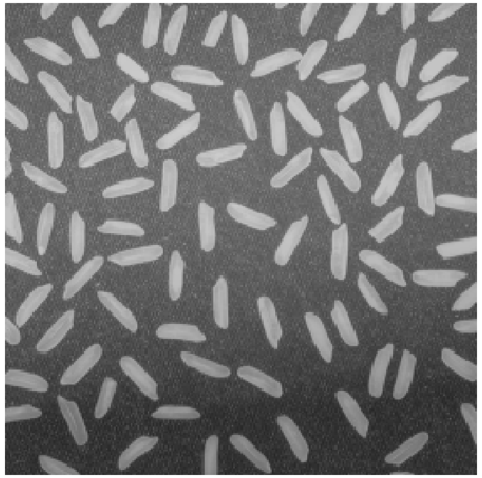


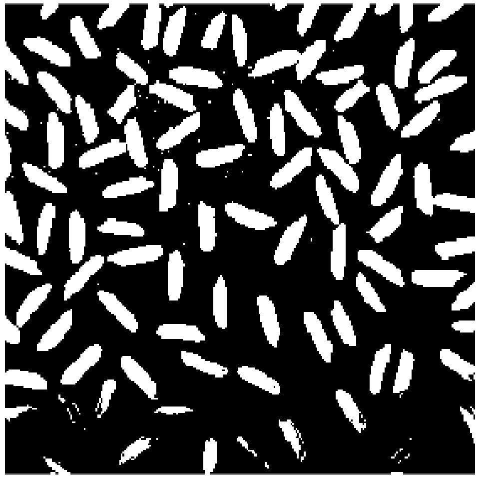


graythresh() and im2bw()

- graythresh() computes an optimal threshold level
- im2bw() converts an images into binary image

```
I = imread('rice.png');
level=graythresh(I);
round(level*255)
bw=im2bw(I, level);
subplot(1,2,1); imshow(I);
subplot (1,2,2); imshow(bw);
```





Exercise

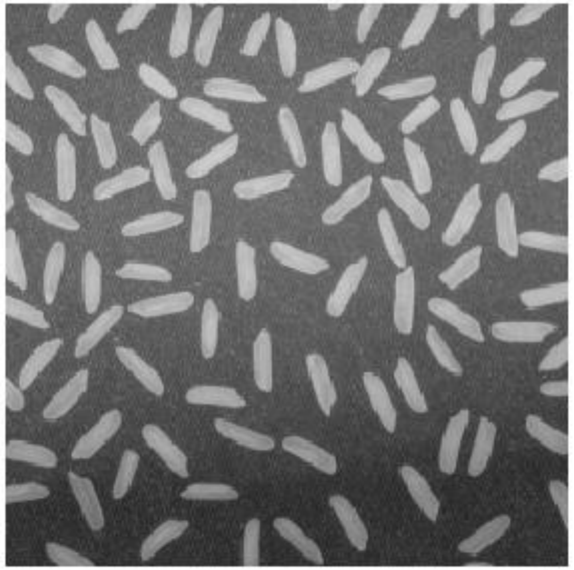
- Write a program to convert the image rice.png into a binary image using a threshold
- Do NOT use `im2bw()`
- Try different threshold values to see if you program works

- [Hint]: `for` and `if` loop

Background estimation

- Estimation for the gray level of the background:

```
I = imread('rice.png');
BG = imopen(I, strel('disk', 15));
imshow(BG);
```



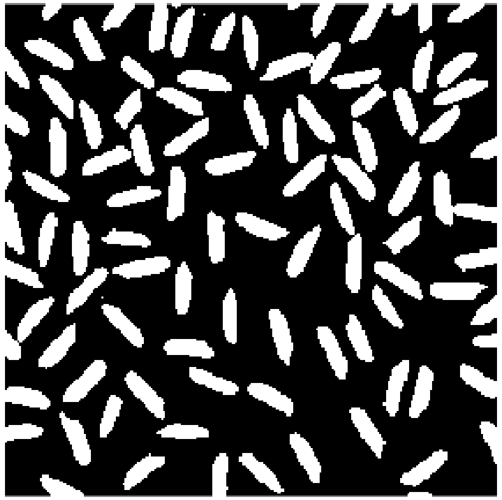
Background subtraction

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

Thresholding on background removed image

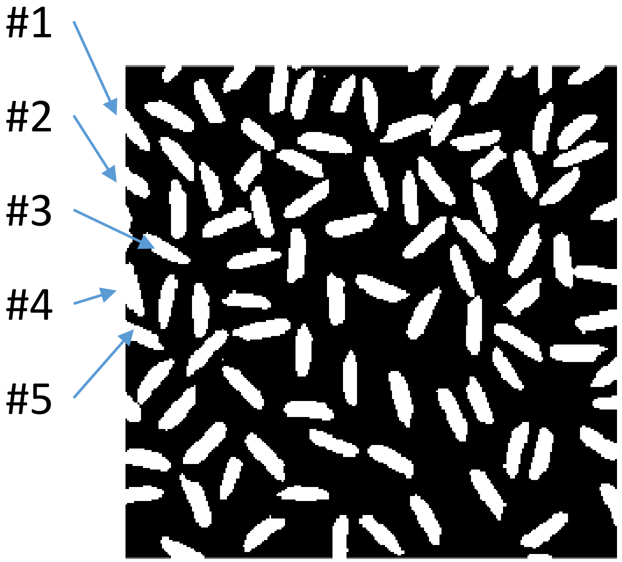
```
I = imread('rice.png');
level = graythresh(I); bw = im2bw(I, level);
subplot (1,2,1); imshow(bw);

BG = imopen(I, strel('disk', 15));
I2 = imsubtract(I, BG); level = graythresh(I2);
bw2 = im2bw(I2, level);
subplot(1,2,2); imshow(bw2);
```



What's next?

- Want to identify how many grains there in the image



Connected-component labeling

- A procedure for assigning a unique label to each object

Binary image

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	0	1	1	1	0
0	1	0	0	1	0	0
0	0	0	0	0	0	0

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	0	0	0
0	0	0	0	0	0	0

Step 1:

Binary image

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	0	1	1	1	0
0	1	0	0	1	0	0
0	0	0	0	0	0	0

Label matrix

0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Connected-component labeling (Continued)

- Finish labeling of a component

Step 2:

Binary image

0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	1	1	1	0
0	1	0	0	1	0	0
0	0	0	0	0	0	0

Label matrix

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Step 3:

Binary image

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	1	1	0
0	1	0	0	1	0	0
0	0	0	0	0	0	0

Label matrix

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Connected-component labeling (Continued)

- Iterative process until all the pixels are checked

Step 4:

Binary image

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	1	0	0
0	0	0	0	0	0	0

Label matrix

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	0	2	2	0	0
0	1	0	0	0	0	0
0	0	0	0	0	0	0

Step 5:

Binary image

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Label matrix

0	0	0	0	0	0	0
0	1	1	0	0	0	0
0	1	0	2	2	2	0
0	1	0	0	2	0	0
0	0	0	0	0	0	0

Connected-component labeling

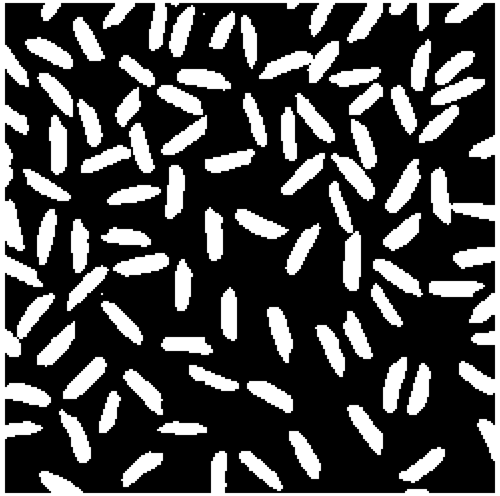
- Built-in connected-component labeling algorithm

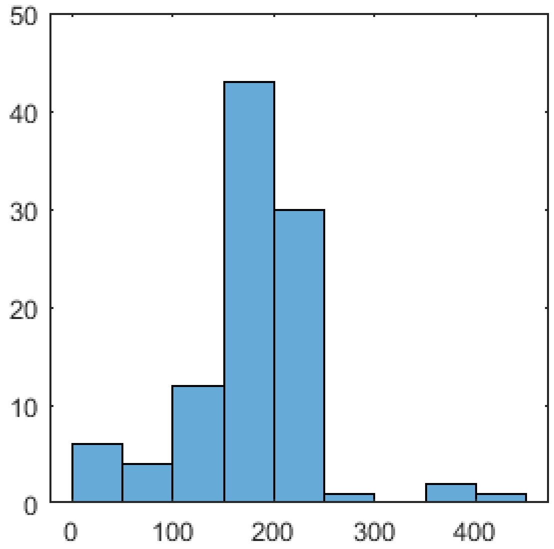
```
I = imread('rice.png');
BG = imopen(I, strel('disk', 15));
I2 = imsubtract(I, BG); level = graythresh(I2);
BW = im2bw(I2, level);
[labeled, numObjects] = bwlabel(BW, 8);
```

- Check the matrix `labeled`

Practice

- Plot the histogram of grain size





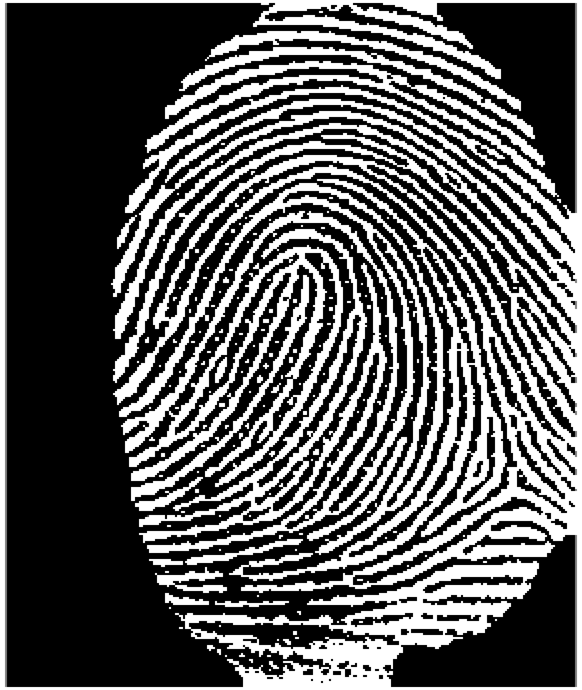
Object properties

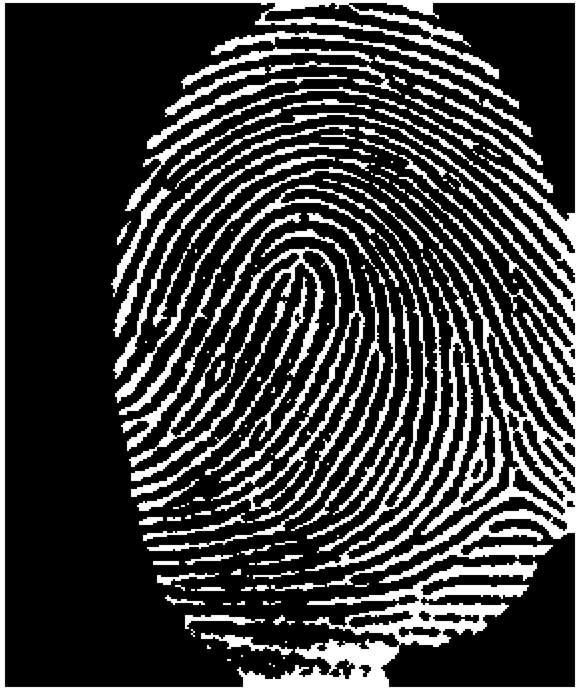
- Provides a set of properties for each connected component

```
I = imread('rice.png');
BG = imopen(I, strel('disk', 15));
I2 = imsubtract(I, BG); level = graythresh(I2);
BW = im2bw(I2, level);
[labeled, numObjects] = bwlabel(BW, 8);
graindata = regionprops(labeled, 'basic');
graindata(51)
```

Problem setup

- Improve the quality of binary image



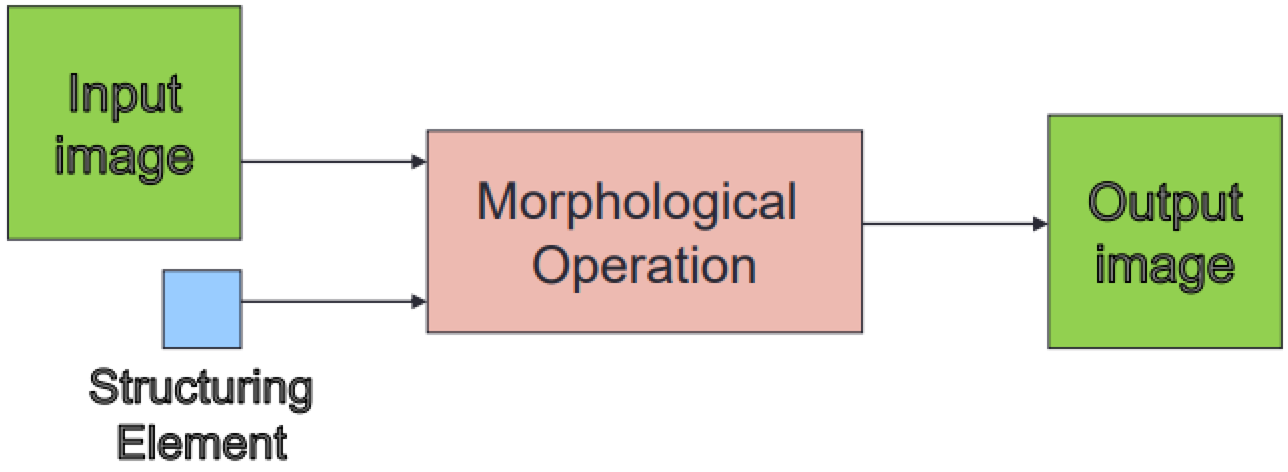


Morphology

- Techniques that deal with the shape of features in an image
  - “Morphological” image processing
- Typically applied to remove imperfections

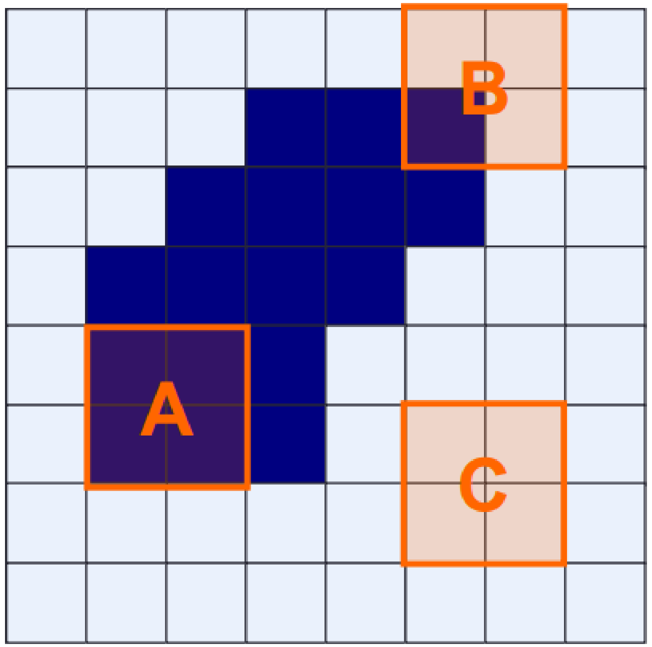
Morphology operations

- Inputs
  - A binary image
  - A “structuring element” image
- Going through the input image to find the regions that “fit” or “hit” the structure elements



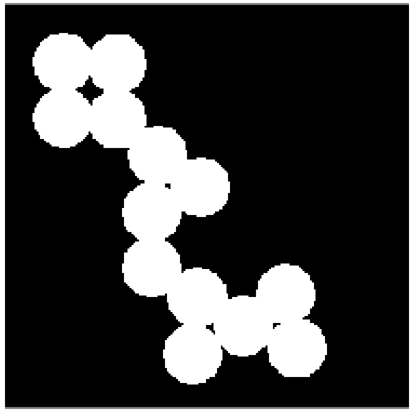
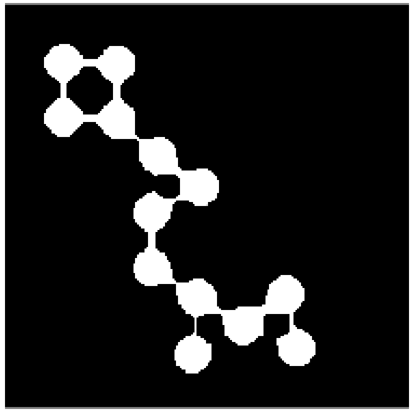
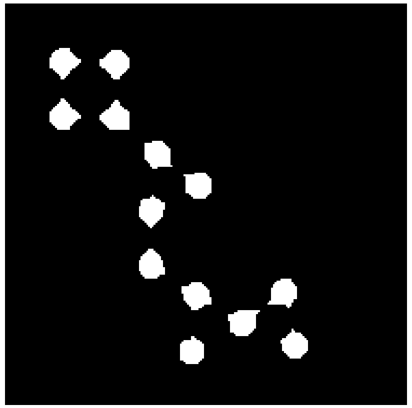
Structuring elements, fit, and hit

- Fit: All on pixels in the structuring element cover on pixels in the image
- Hit: Any on pixel in the structuring element covers an on pixel in the image



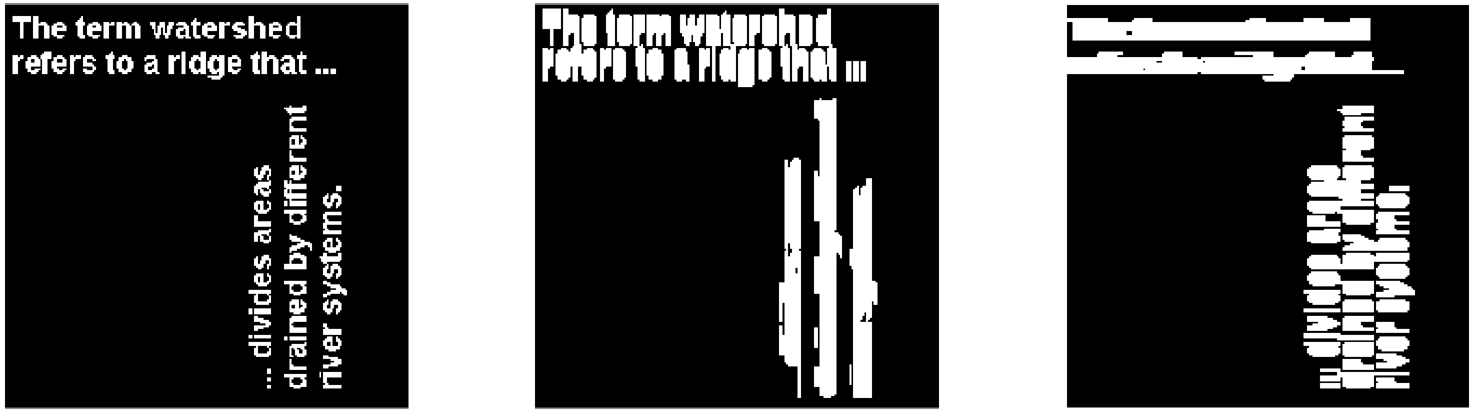
imerode()

```
originalBW = imread('circles.png');
subplot( 1, 3, 1); imshow(originalBW);
se = strel('disk', 7);
erodedBW = imerode(originalBW, se);
subplot( 1, 3, 2); imshow(erodedBW);
se = strel('disk', 11);
erodedBW = imerode(originalBW, se);
subplot( 1, 3, 3); imshow(erodedBW);
```

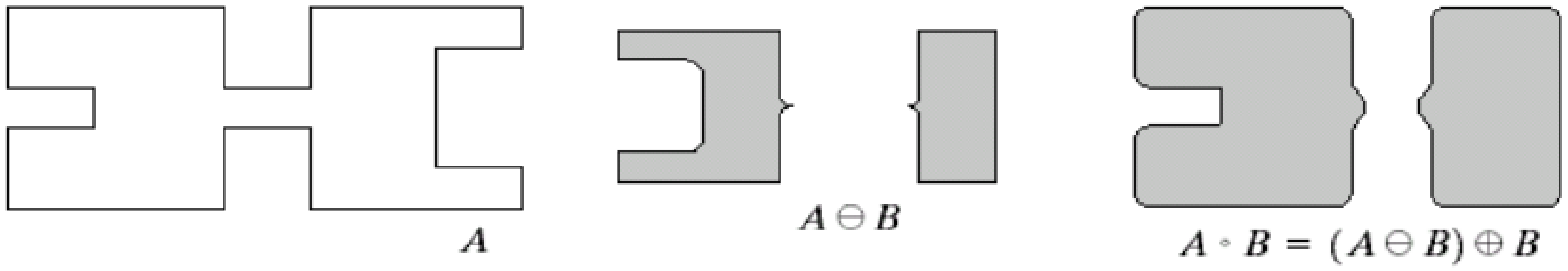
imdilate()

```
bw = imread('text.png');
subplot( 1, 3, 1); imshow(bw);
se = strel('line', 11, 90);
dilatedBW = imdilate(bw, se);
subplot( 1, 3, 2); imshow(dilatedBW);
se = strel('line', 11, 0);
dilatedBW = imdilate(bw, se);
subplot( 1, 3, 3); imshow(dilatedBW);
```



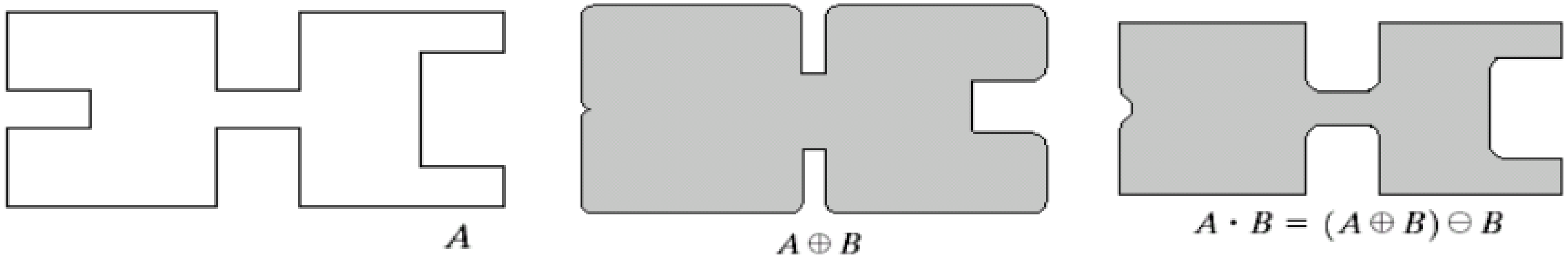
Compound operations

- Opening: erosion ➔ dilation



Original shape   After erosion   After dilation

- Closing: dilation ➔ erosion

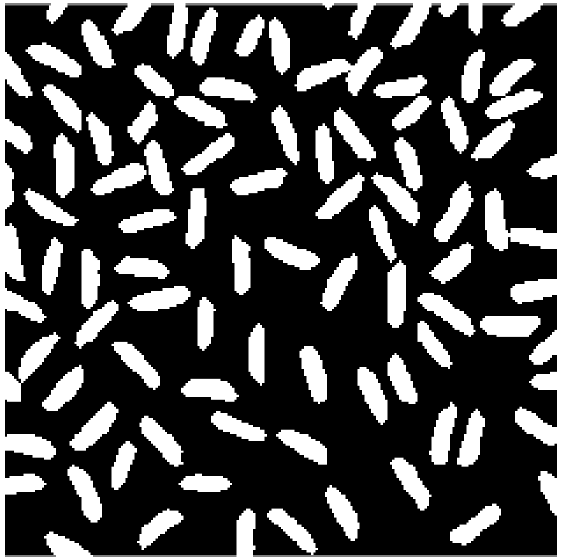


Original shape   After dilation   After erosion

Rice example

```
I = imread('rice.png');
BG = imopen(I, strel('disk', 15));
I2 = imsubtract(I, BG); level = graythresh(I2);
BW = im2bw(I2, level);
subplot(1, 2, 1); imshow(BW);

BW2 = imopen(BW, strel('diamond',2));
subplot(1, 2, 2); imshow(BW2);
```

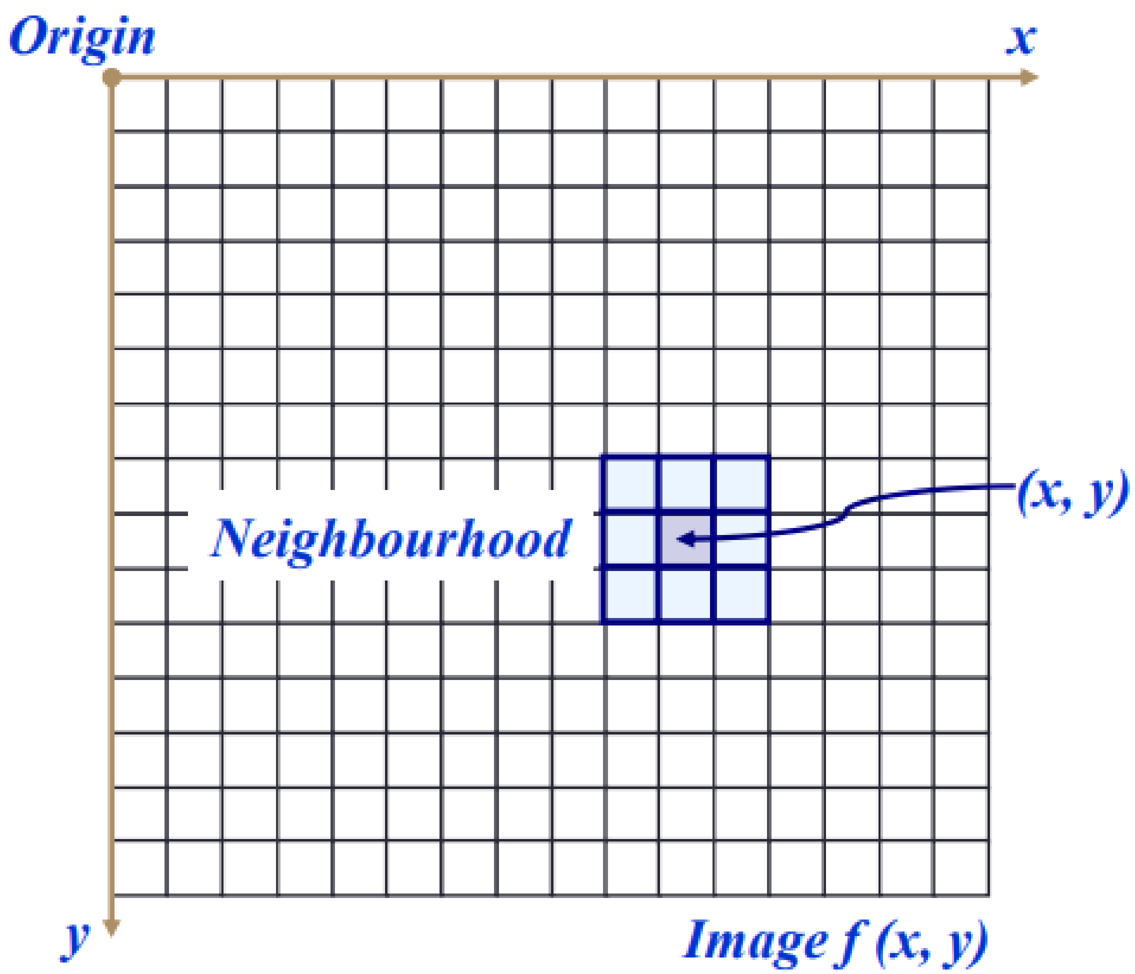


Problem Setup

- Improve the “quality” of gray-level image

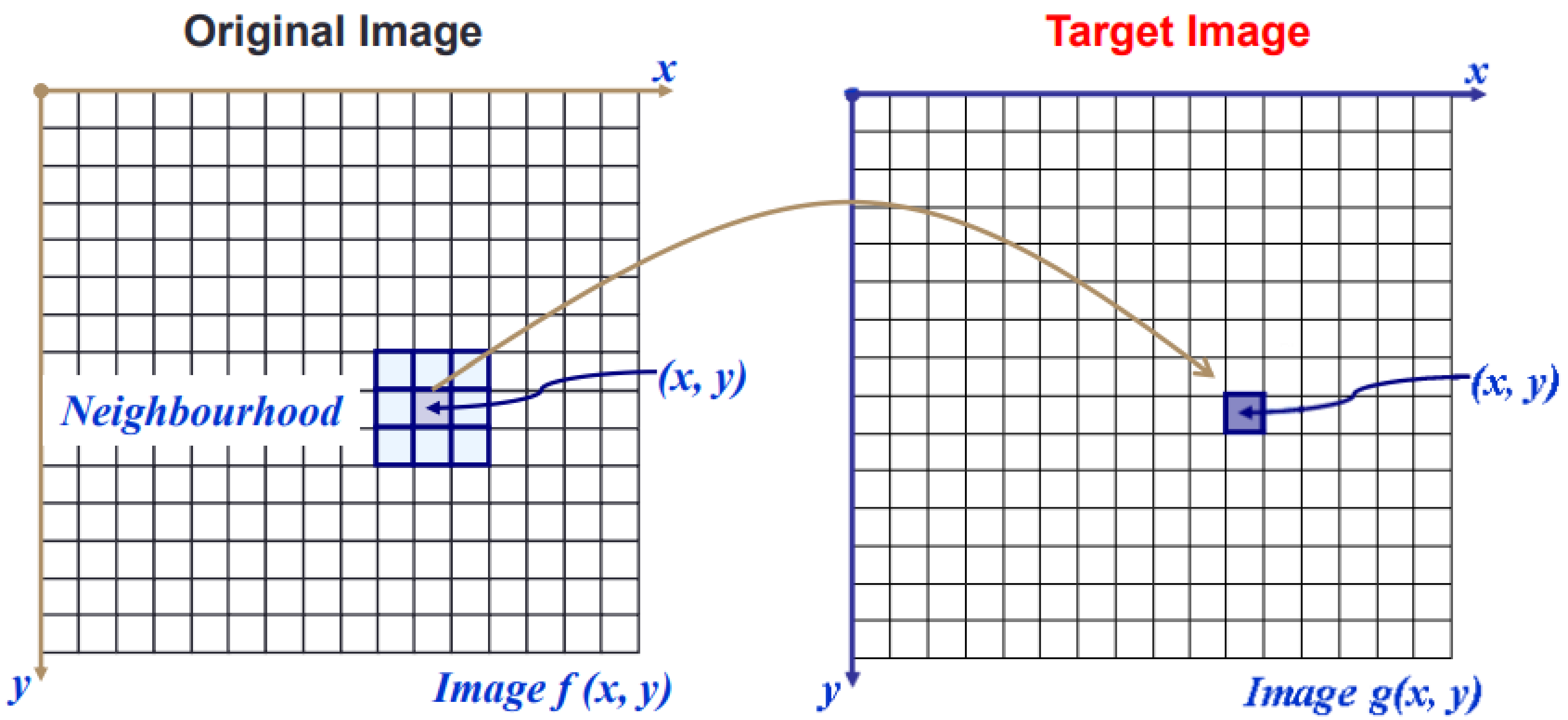
Neighborhood operations

- Operate on a larger neighborhood of pixels rather than point operations
- Neighborhoods are mostly a rectangle around a central pixel
- Any size rectangle and any shape filter are possible



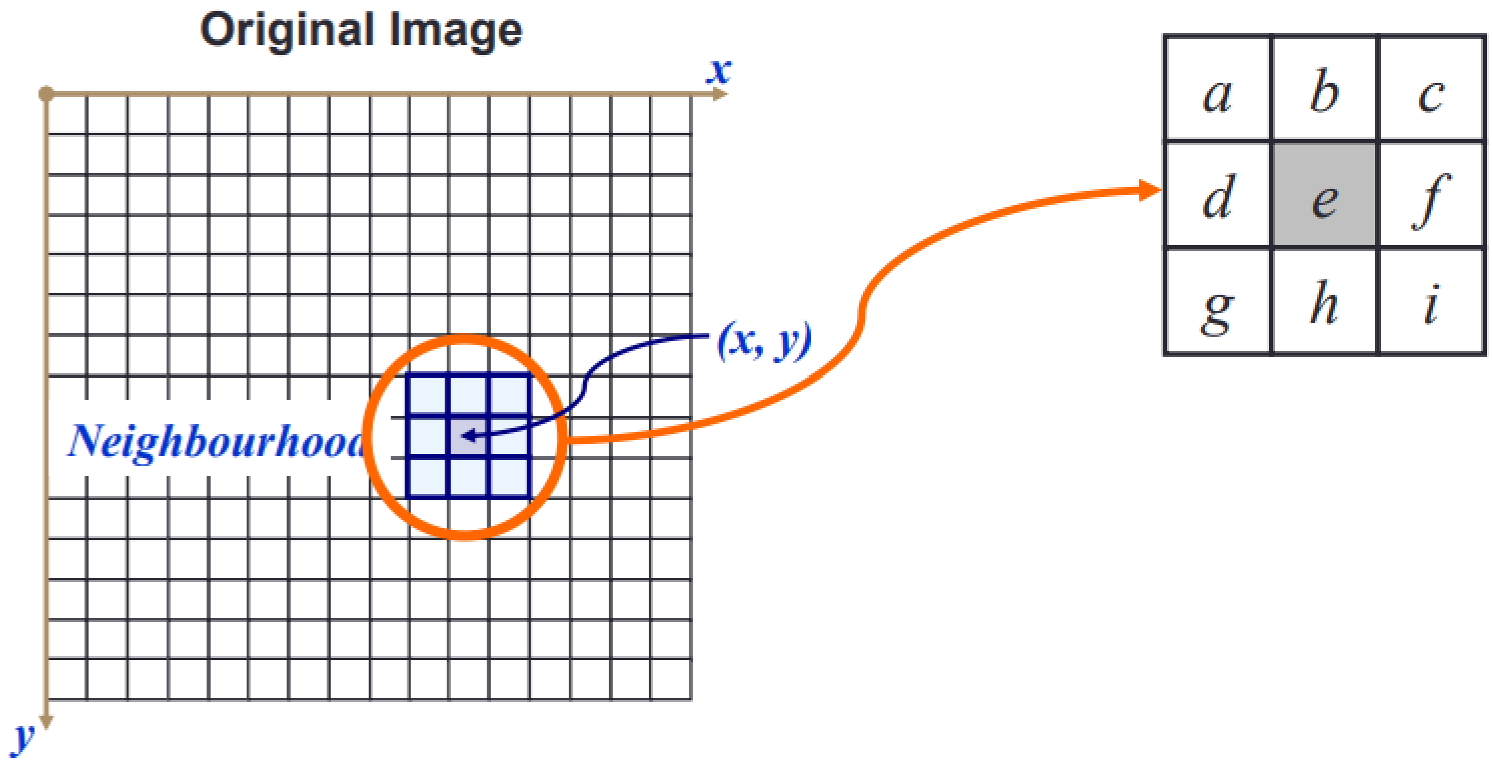
Neighborhood operations

- For each pixel in the origin image, the outcome is written on the same location at the target image



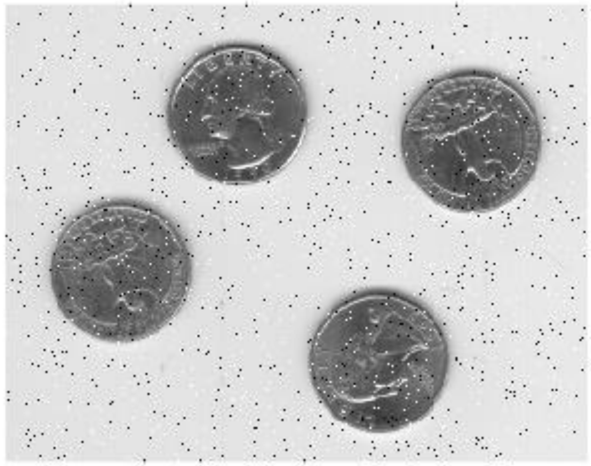
Simple neighborhood operations

- Set the pixel value to the maximum, minimum, or median in the neighborhood

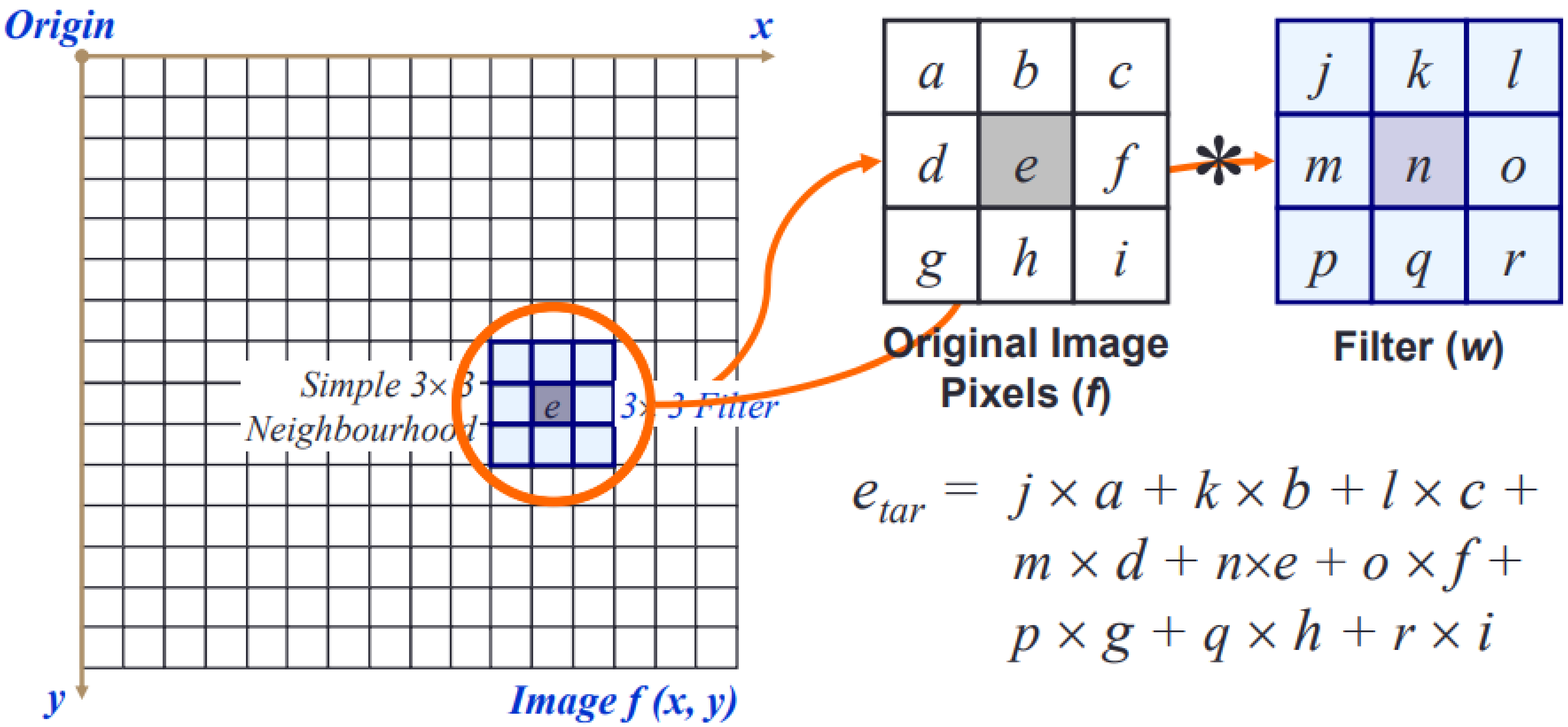


Median neighborhood operation

```
I = imread('eight.tif');
subplot( 1, 3, 1); imshow(I);
J = imnoise(I,'salt & pepper',0.02);
subplot( 1, 3, 2); imshow(J);
K = medfilt2(J);
subplot( 1, 3, 3); imshow(K);
```

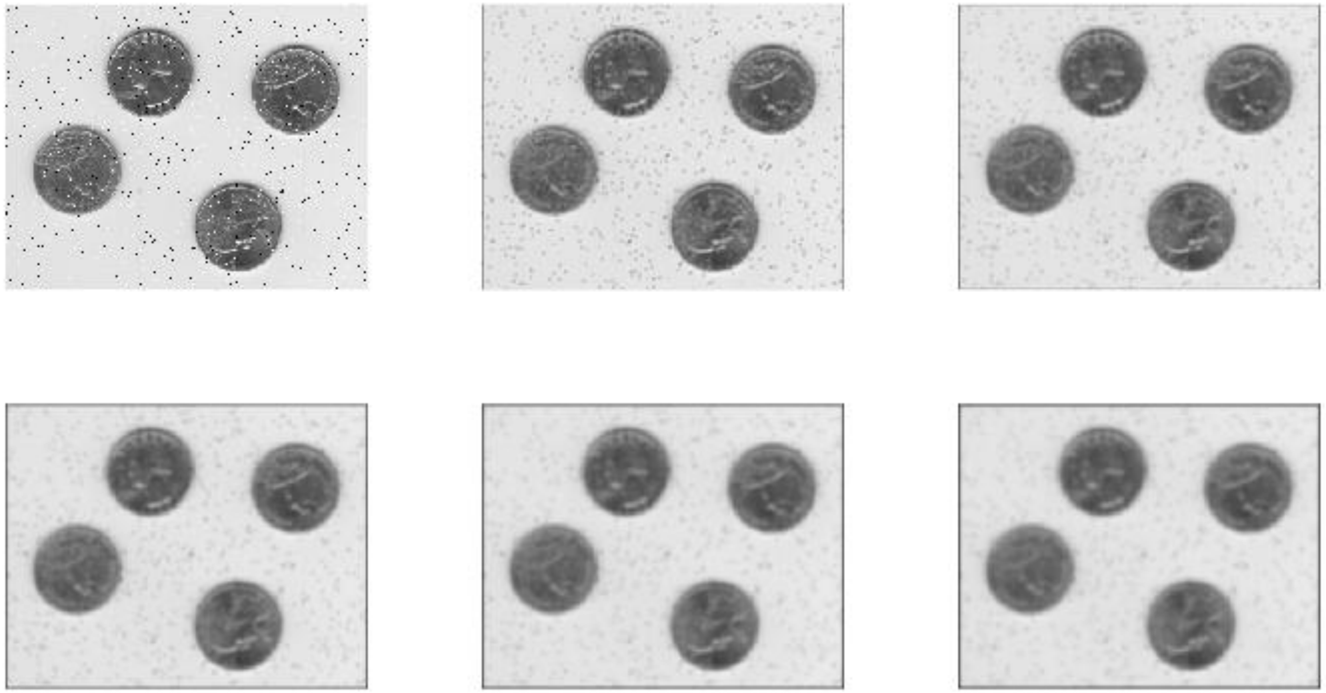

Spatial filtering



$$
\begin{aligned}
e_{tar} = {} & j \times a + k \times b + l \times c + \\
& m \times d + n \times e + o \times f + \\
& p \times g + q \times h + r \times i
\end{aligned}
$$

Averaging filtering

```
J = imnoise(imread('eight.tif'),'salt & pepper',0.02);
subplot( 2, 3, 1); imshow(J);
for i=1:5
    J = uint8(filter2(fspecial('average', 3), J));
    subplot( 2, 3, i+1); imshow(J);
end
```



`fspecial()`

- Creates a two-dimensional filter

Value	Description
`average`	Averaging filter
`disk`	Circular averaging filter (pillbox)
`gaussian`	Gaussian lowpass filter
`laplacian`	Approximates the two-dimensional Laplacian operator
`log`	Laplacian of Gaussian filter
`motion`	Approximates the linear motion of a camera
`prewitt`	Prewitt horizontal edge-emphasizing filter
`sobel`	Sobel horizontal edge-emphasizing filter

Sobel filter

```
I = imread('cameraman.tif');
subplot(1, 4, 1); imshow(I);
J = uint8(filter2(fspecial('sobel'), I));
subplot(1, 4, 2); imshow(J);
K = uint8(filter2(fspecial('sobel')', I));
subplot(1, 4, 3); imshow(K);
L = J + K; subplot(1, 4, 4); imshow(L);
```

Weighted averaging filters

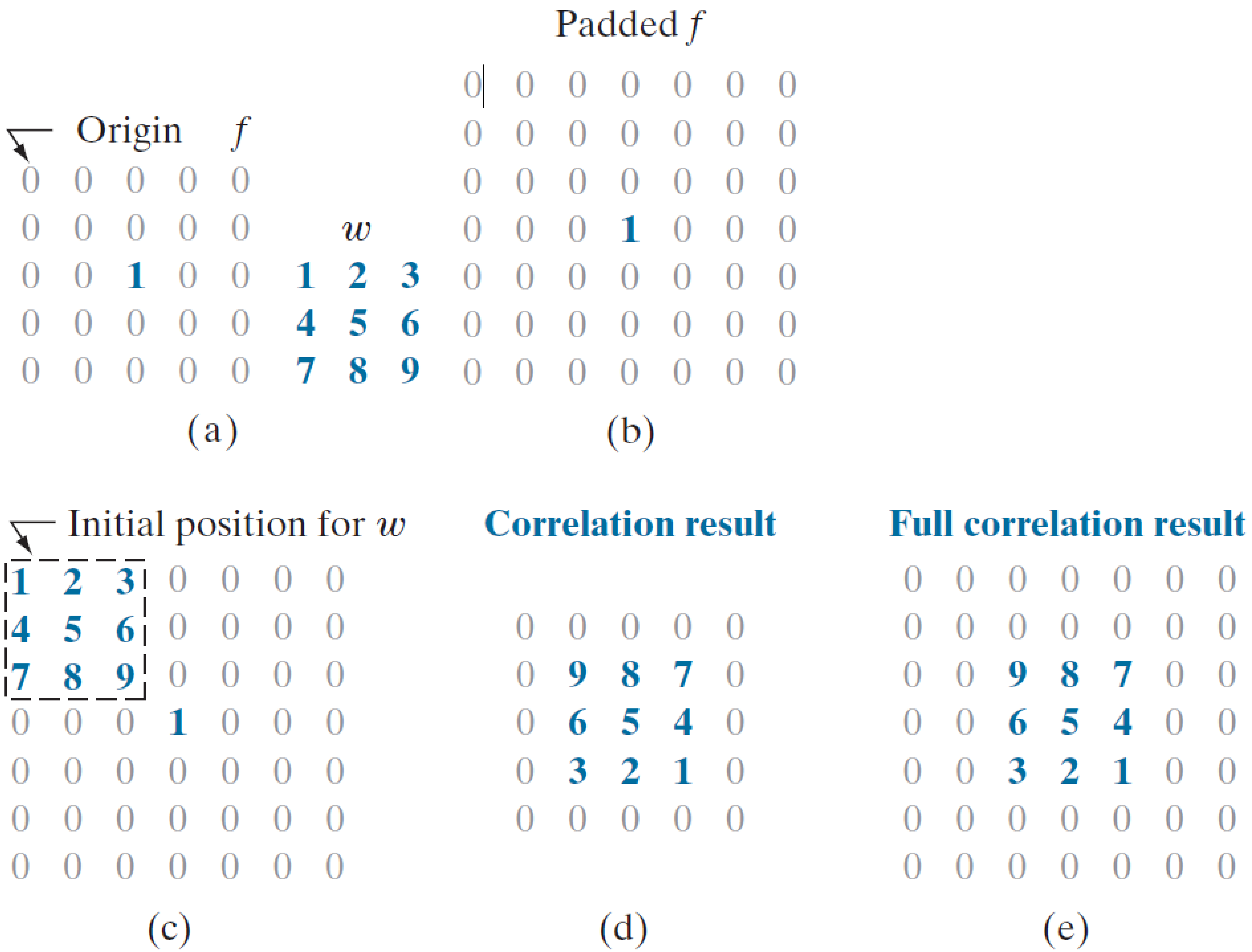
- Different pixels in the neighborhood associated with different weights in the averaging function
- Pixels closer to the central pixel are more important

$^{1}/_{16}$	$^{2}/_{16}$	$^{1}/_{16}$
$^{2}/_{16}$	$^{4}/_{16}$	$^{2}/_{16}$
$^{1}/_{16}$	$^{2}/_{16}$	$^{1}/_{16}$

Weighted averaging filter

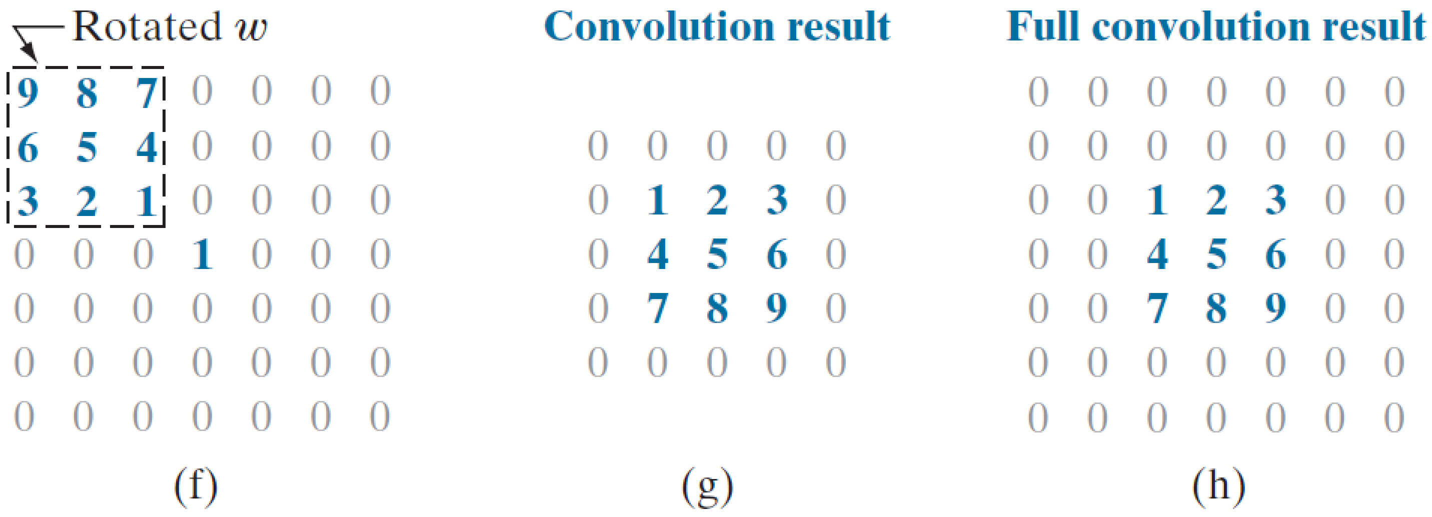
Convolution vs. correlation

- Combine two signals (input image and filter) to form a third signal (output image)



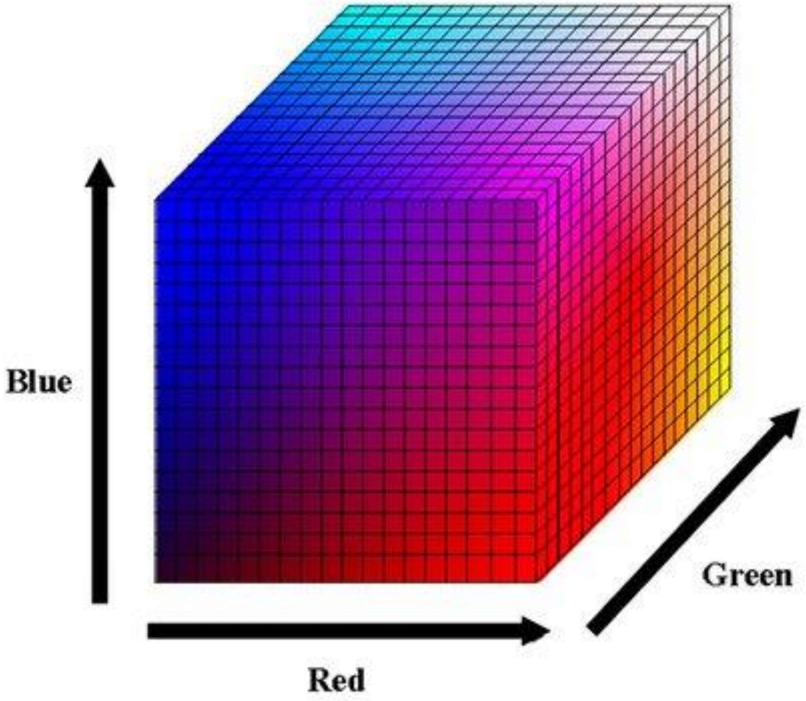
Convolution vs. correlation (continued)

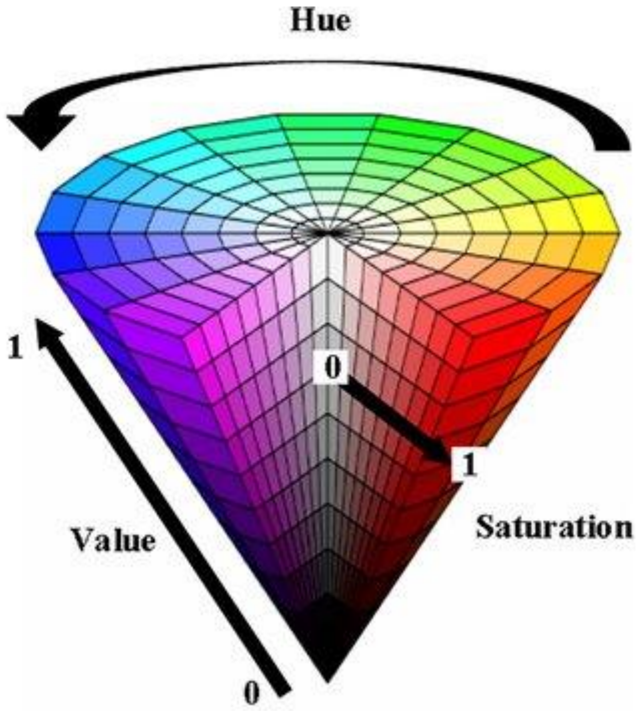
- Convolution
  - Enhance the spatial feature that matches with the filter



Common color space

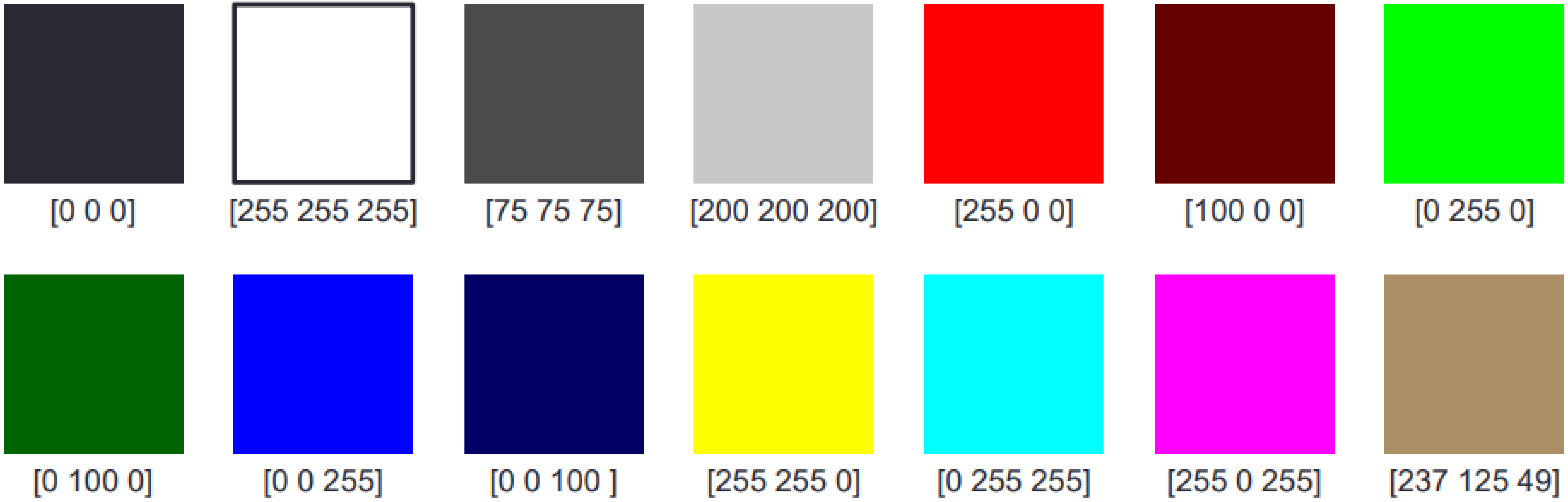
- RGB
- HSV (or HSI)





RGB color space

- For variables of class uint8
  - 0-255 are the possible integer values
  - 0 is minimum for any RGB color
  - 255 is max for any RGB color



RGB to HSV

$$I = \frac{1}{3}(R + G + B)$$

$$S = 1 - \frac{3}{R + G + B}\left[\min(R, G, B)\right]$$

$$H = \cos^{-1}\left[\frac{\frac{1}{2}[(R - G) + (R - B)]}{\sqrt{(R - G)^2 + (R - B)(G - B)}}\right]$$

Color Space Conversion

- `rgb2hsv`
- `hsv2rgb`
- `rgb2ycbcr`
- `ycbcr2rgb`
- `rgb2gray`
- `makecform`
- `applycform`

Typical process for image processing

- Problem domain

- Image acquisition
- Image preprocessing
- Segmentation
- Morphological processing
- Object recognition

- Representation and description

End of class

- Homework 02
  - Due 7/18

MATLAB Basic

Wednesdays 10:00-12:00 am | 溫室大樓101

Hao-Chun Hsu
domingo0201@gmail.com)

Lecture	Assignment Due
7/6 Introduction Operation | Programming	**Homework 01** 7/11
7/13 Image processing Blobing | Morphology | Filtering | Color space	**Homework 02** 7/18
7/20 Data analysis Basic plotting | Basic statistics	**Homework 03** 7/25