CSCE566-DATA MINING
WEEK 8

Image Data Mining

Min Shi
min.shi@louisiana.edu

Oct 14, 2024

Outline: Image Data Mining

1. Introduction
2. Digital Image Processing
3. Convolutional Neural Network
4. Computer Vision Tasks

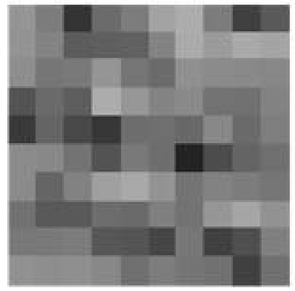
What is an image?

2-dimensional matrix of Intensity (gray or color) values

Set of Intensity values

Image coordinates are integers

$$I(u,v) \in \mathbb{P} \quad \text{and} \quad u,v \in \mathbb{N}.$$



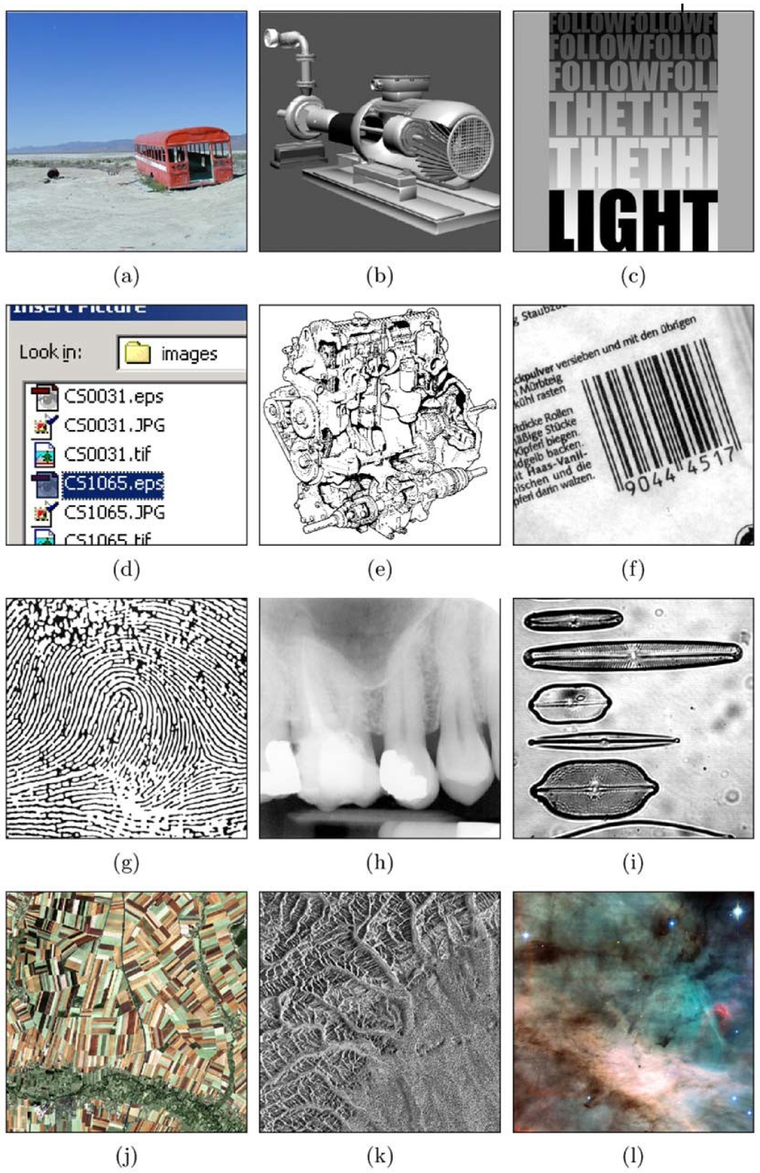
$F(x,y)$

$$\begin{bmatrix}
148 & 123 & 52 & 107 & 123 & 162 & 172 & 123 & 64 & 89 & \cdots \\
147 & 130 & 92 & 95 & 98 & 130 & 171 & 155 & 169 & 163 & \cdots \\
141 & 118 & 121 & 148 & 117 & 107 & 144 & 137 & 136 & 134 & \cdots \\
82 & 106 & 93 & 172 & 149 & 131 & 138 & 114 & 113 & 129 & \cdots \\
57 & 101 & 72 & 54 & 109 & 111 & 104 & 135 & 106 & 125 & \cdots \\
138 & 135 & 114 & 82 & 121 & 110 & 34 & 76 & 101 & 111 & \cdots \\
138 & 102 & 128 & 159 & 168 & 147 & 116 & 129 & 124 & 117 & \cdots \\
113 & 89 & 89 & 109 & 106 & 126 & 114 & 150 & 164 & 145 & \cdots \\
120 & 121 & 123 & 87 & 85 & 70 & 119 & 64 & 79 & 127 & \cdots \\
145 & 141 & 143 & 134 & 111 & 124 & 117 & 113 & 64 & 112 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots &
\end{bmatrix}$$

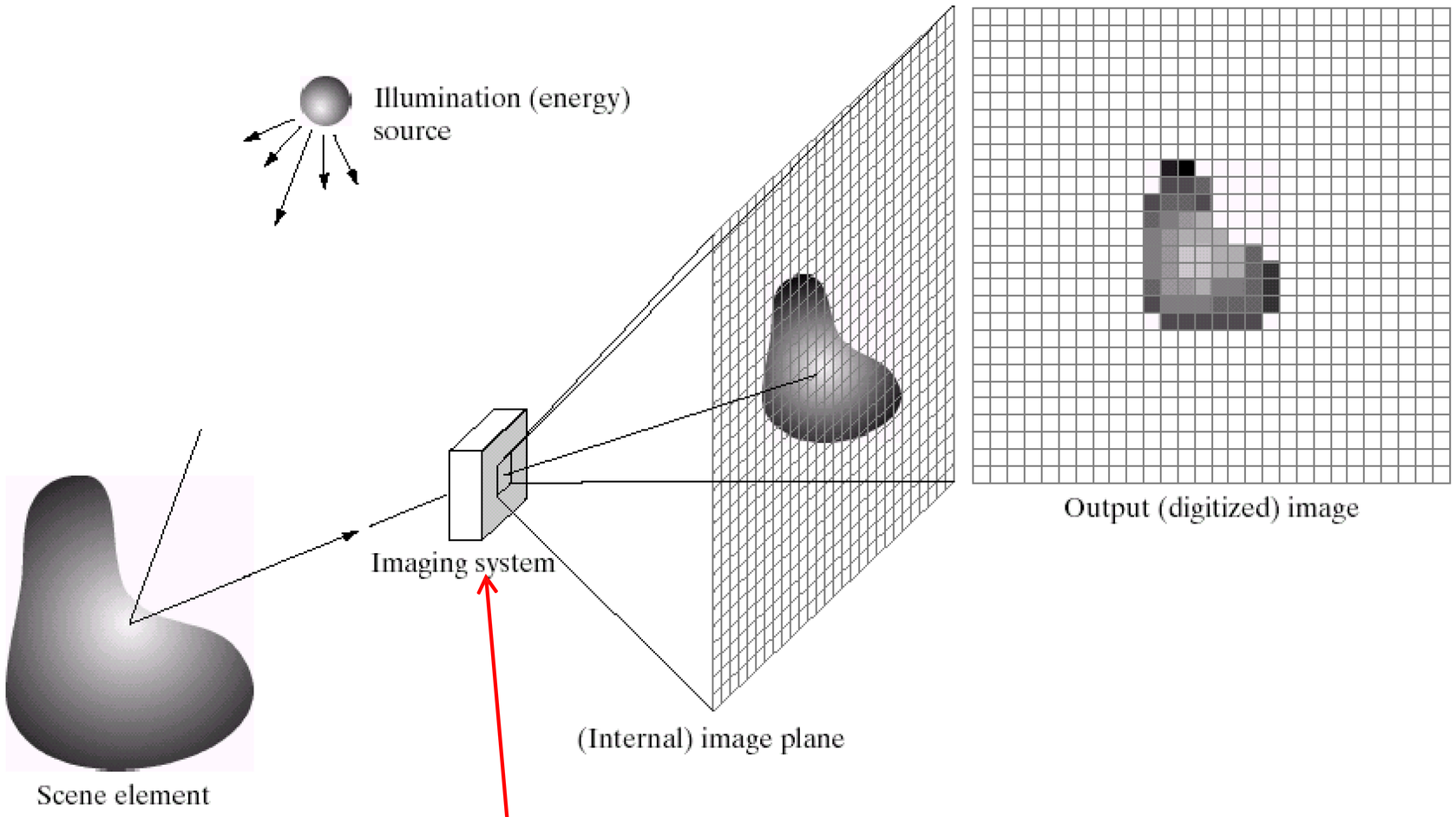
$I(u,v)$

Example of digital images

a) Natural landscape
b) Synthetically generated scene
c) Poster graphic
d) Computer screenshot
e) Black and white illustration
f) Barcode
g) Fingerprint
h) X-ray
i) Microscope slide
j) Satellite Image
k) Radar image
l) Astronomical object



Imaging system

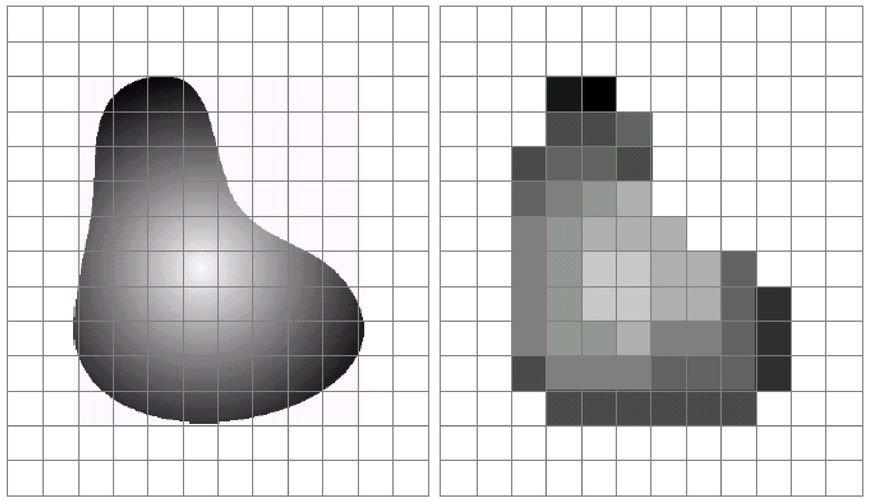


Example: a camera
Converts light to image

Credits: Gonzales and Woods

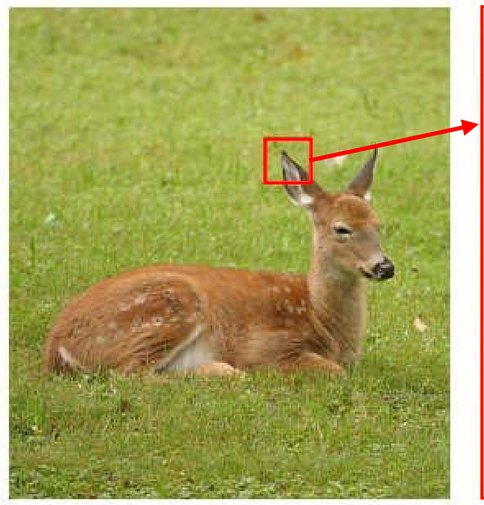
Digital image

Digitization causes a digital image to become an *approximation* of a real scene

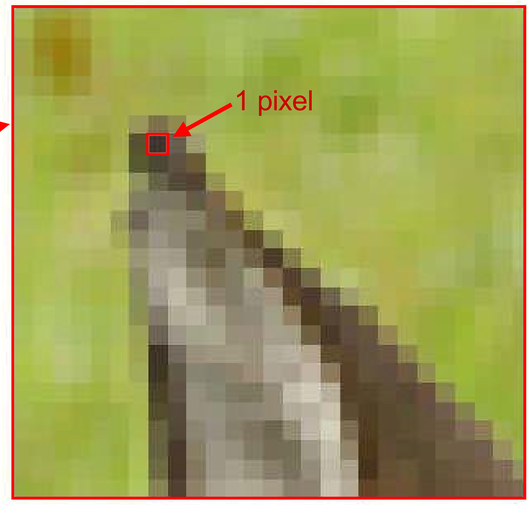


Real image

Digital Image
(an approximation)



Real image



Digital Image
(an approximation)

Digital image

- Common image formats include:
  - 1 values per point/pixel (B&W or Grayscale)
  - 3 values per point/pixel (Red, Green, and Blue)
  - 4 values per point/pixel (Red, Green, Blue, + “Alpha” or Opacity)

Grayscale RGB RGBA

- We will start with gray-scale images, extend to color later

What is image processing?

- **Algorithms that alter an input image to create new image**
- **Input is image, output is image**

Original Image

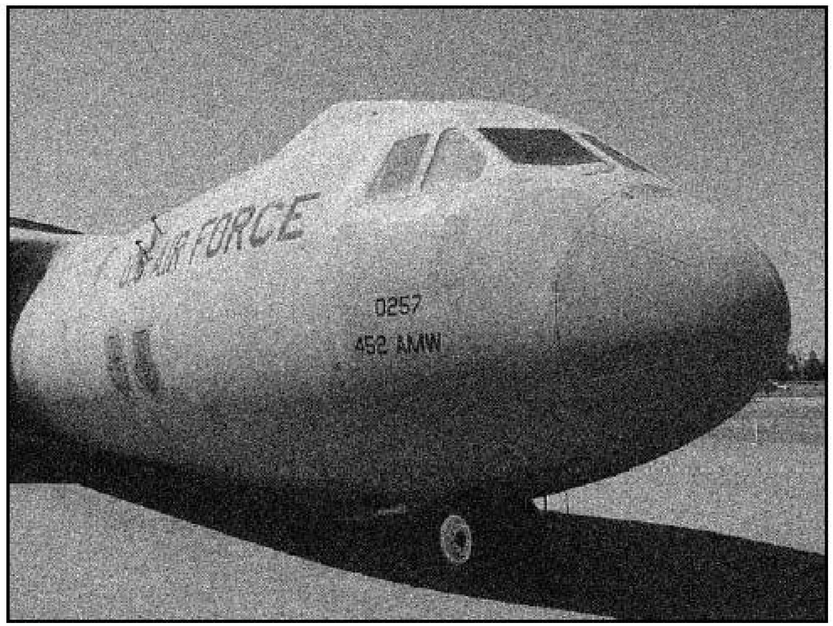
Image Processing Algorithm (e.g. Sobel Filter)

Processed Image

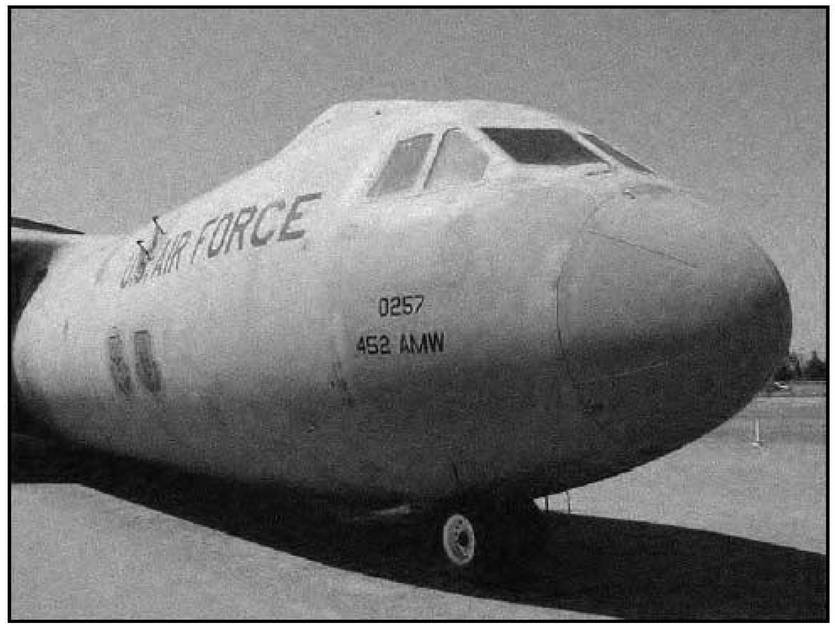
- **Improves an image for human interpretation in ways including:**
  - Image display and printing
  - Image editting
  - Image enhancement
  - Image compression

Example operation: noise removal

Noisy Image

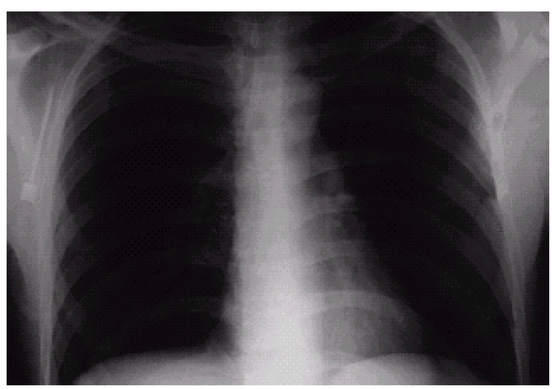


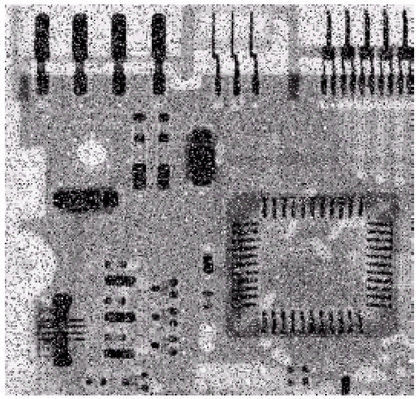
Denoised Image

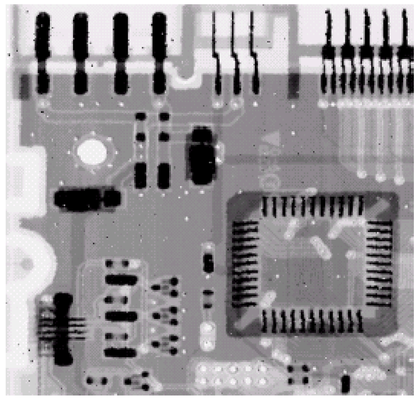


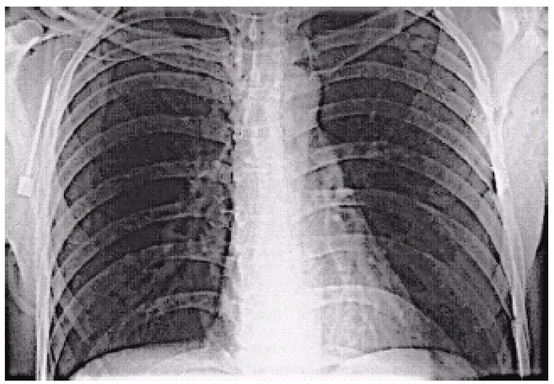
Think of noise as white specks on a picture (random or non-random)

Example operation: noise removal





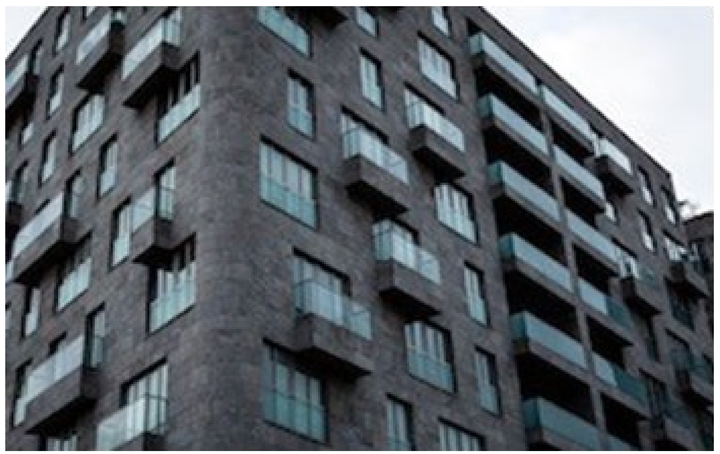




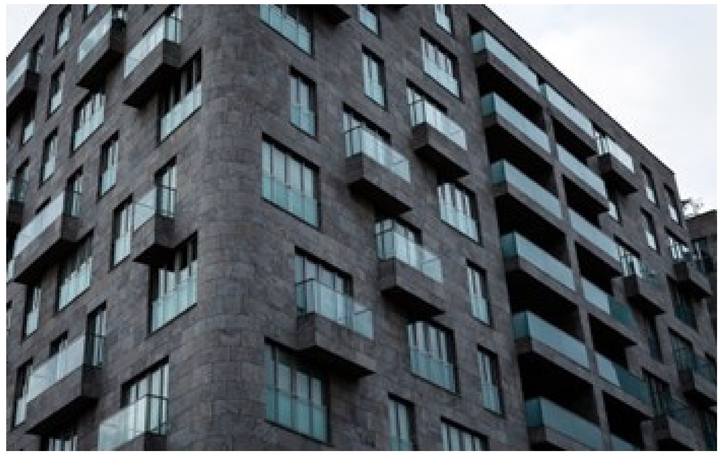
Example: contrast adjustment

Low Contrast | Original Contrast | High Contrast

Example: image super-resolution



input

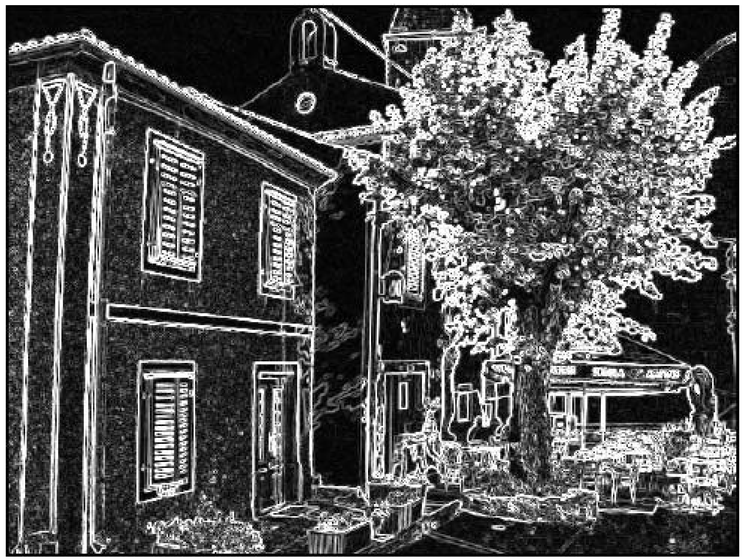


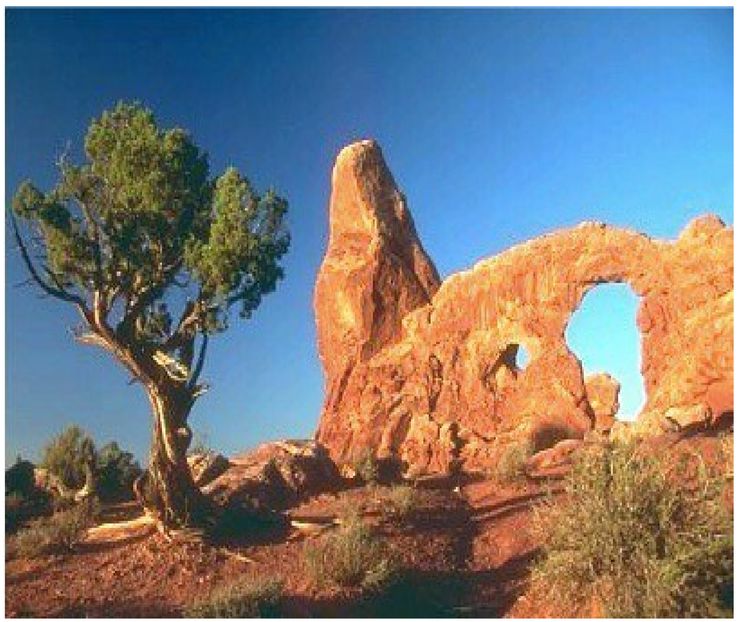
4x original

input

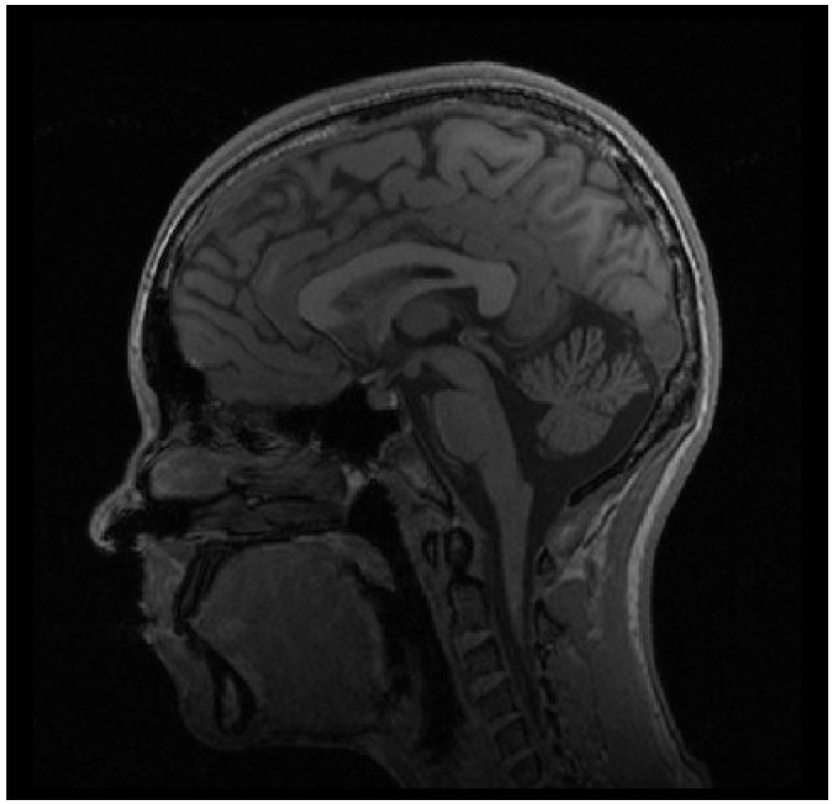
4x output

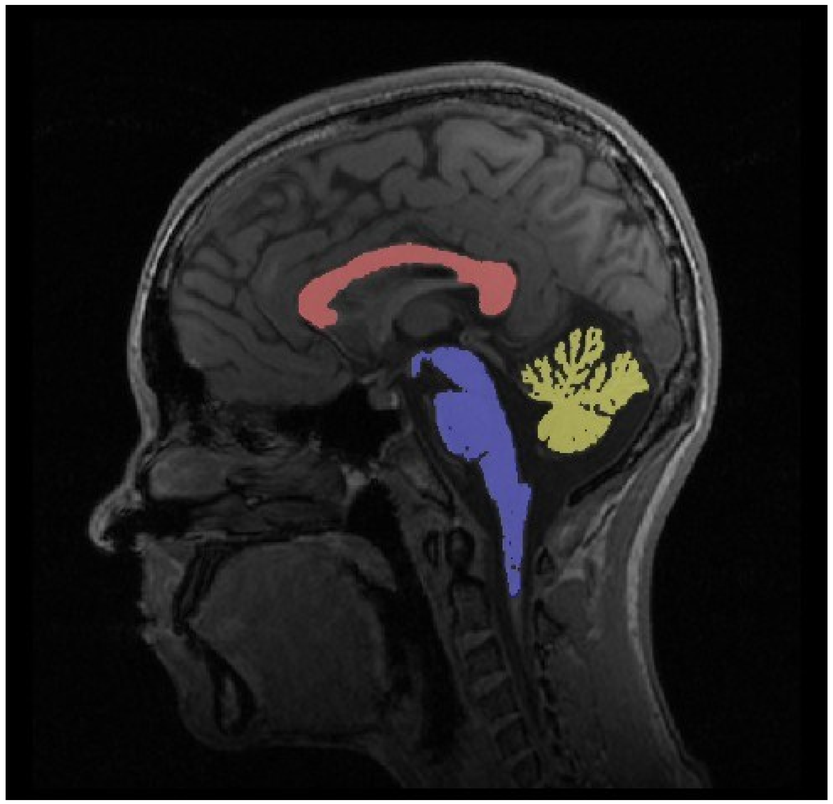
Example: edge detection





Example: region detection / segmentation





Example: image compression

Original, 2.1MB

JPEG Compression, 308KB (15%)

Example: image inpainting

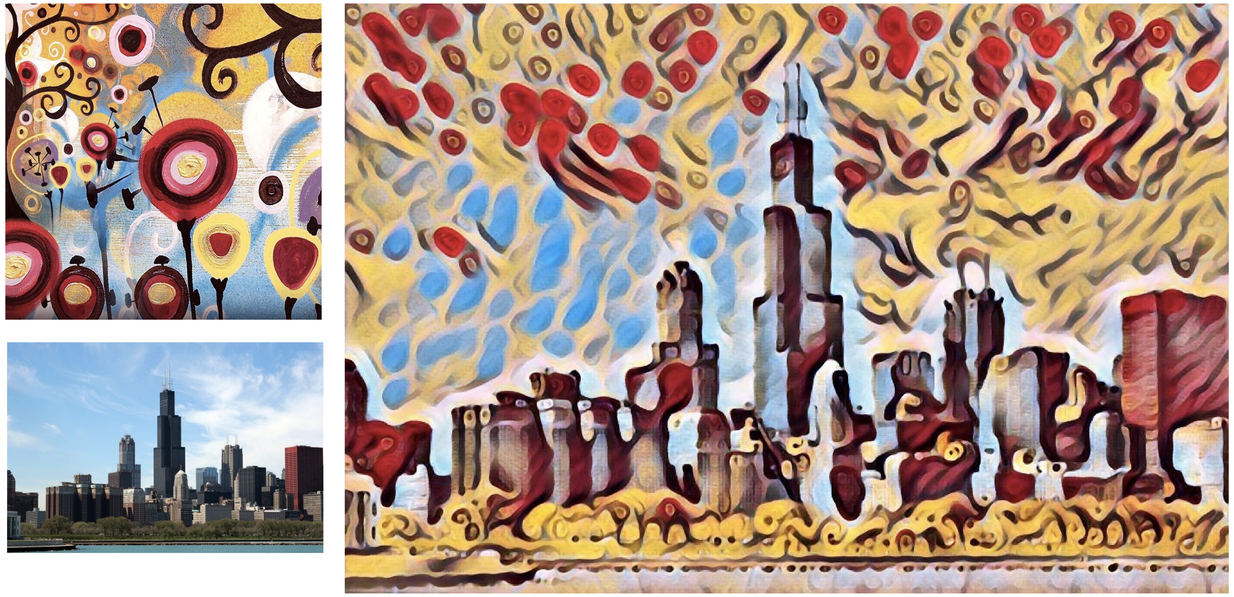
Damaged Image

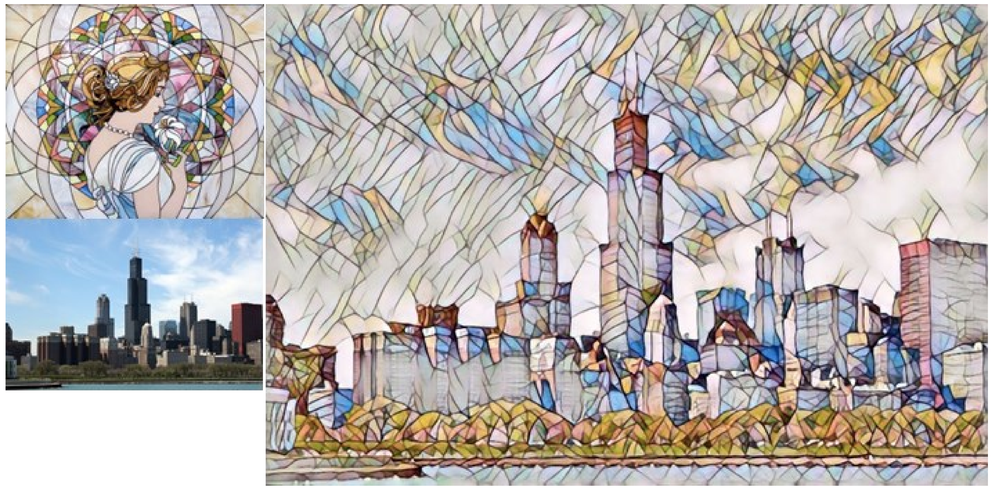
Restored Image

Credit: M. Bertalmio, G. Sapiro, V. Caselles, C. Ballester: Image Inpainting, SIGGRAPH 2000

Inpainting? Reconstruct corrupted/destroyed parts of an image

Example: image style transfer





Application of image processing

Examples: Artistic (Movie Special)Effects

Application of image processing

Law Enforcement

- Number plate recognition for speed cameras or automated toll systems
- Fingerprint recognition

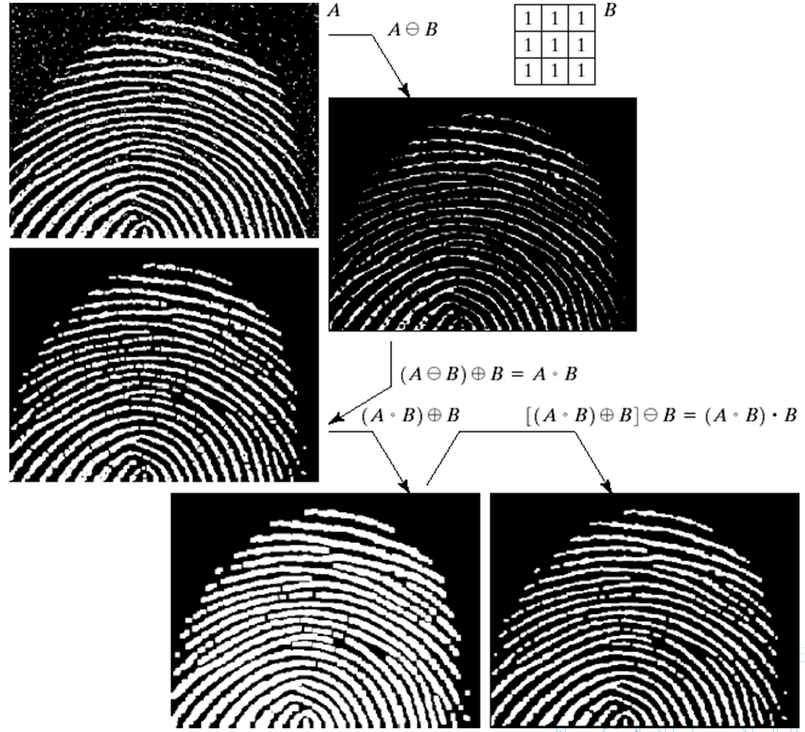
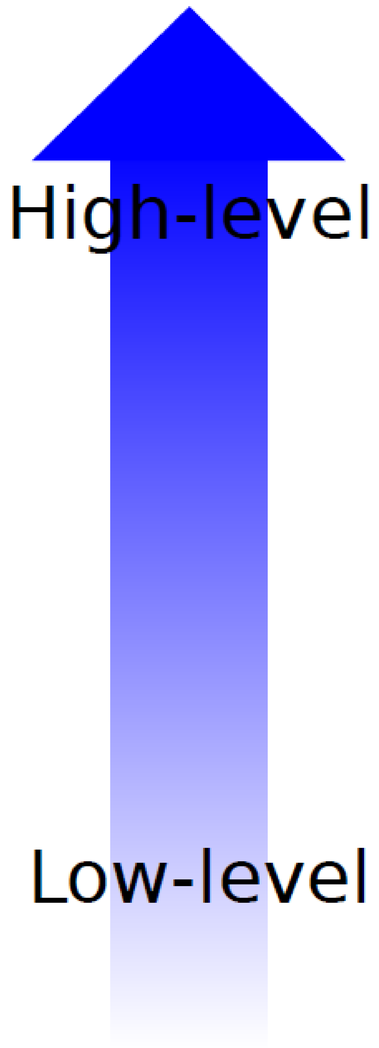


Image processing, image analysis, computer vision



Computer Vision

Object detection, recognition, shape analysis, tracking
Use of Artificial Intelligence and Machine Learning

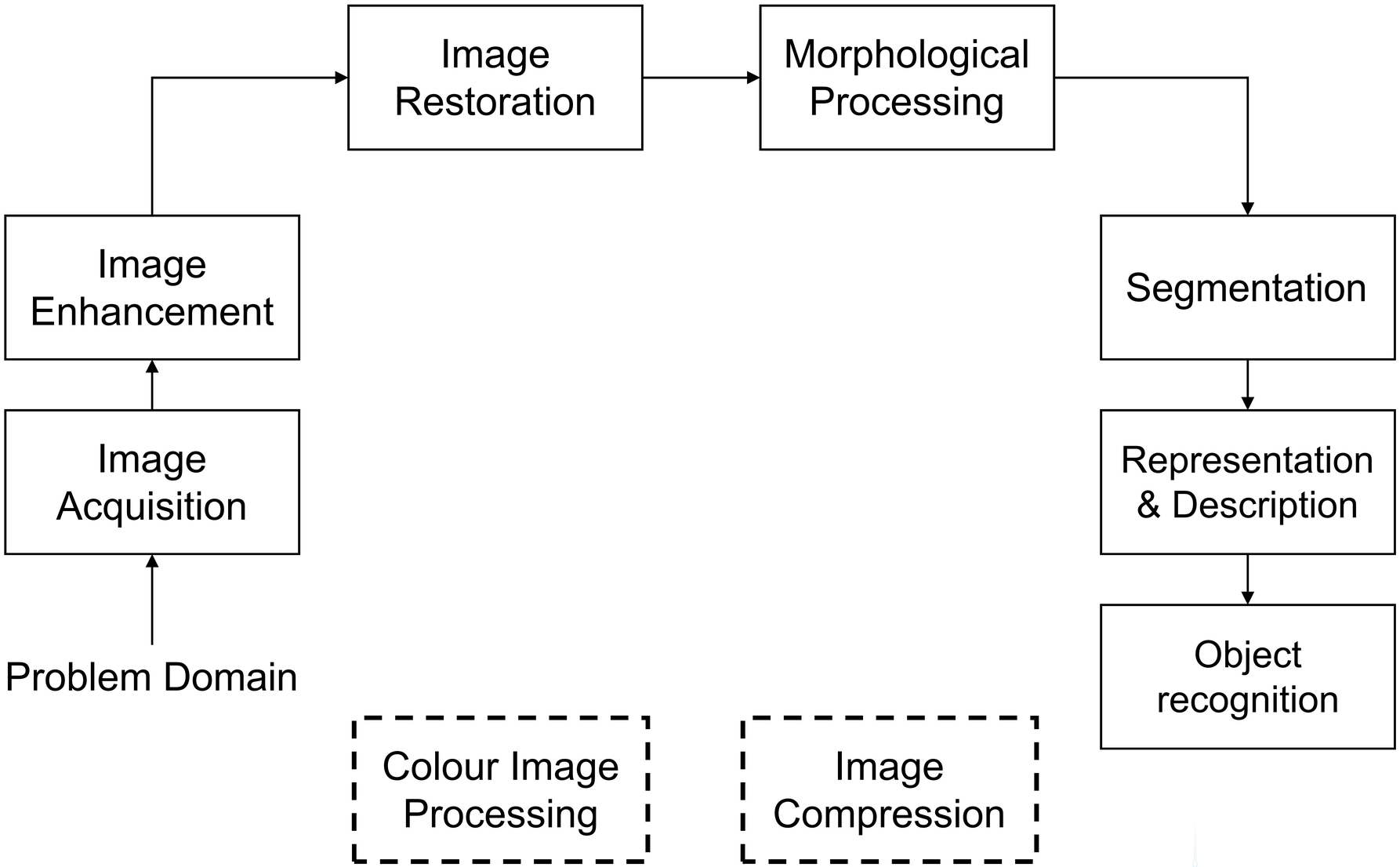
Image Analysis

Segmentation, image registration, matching

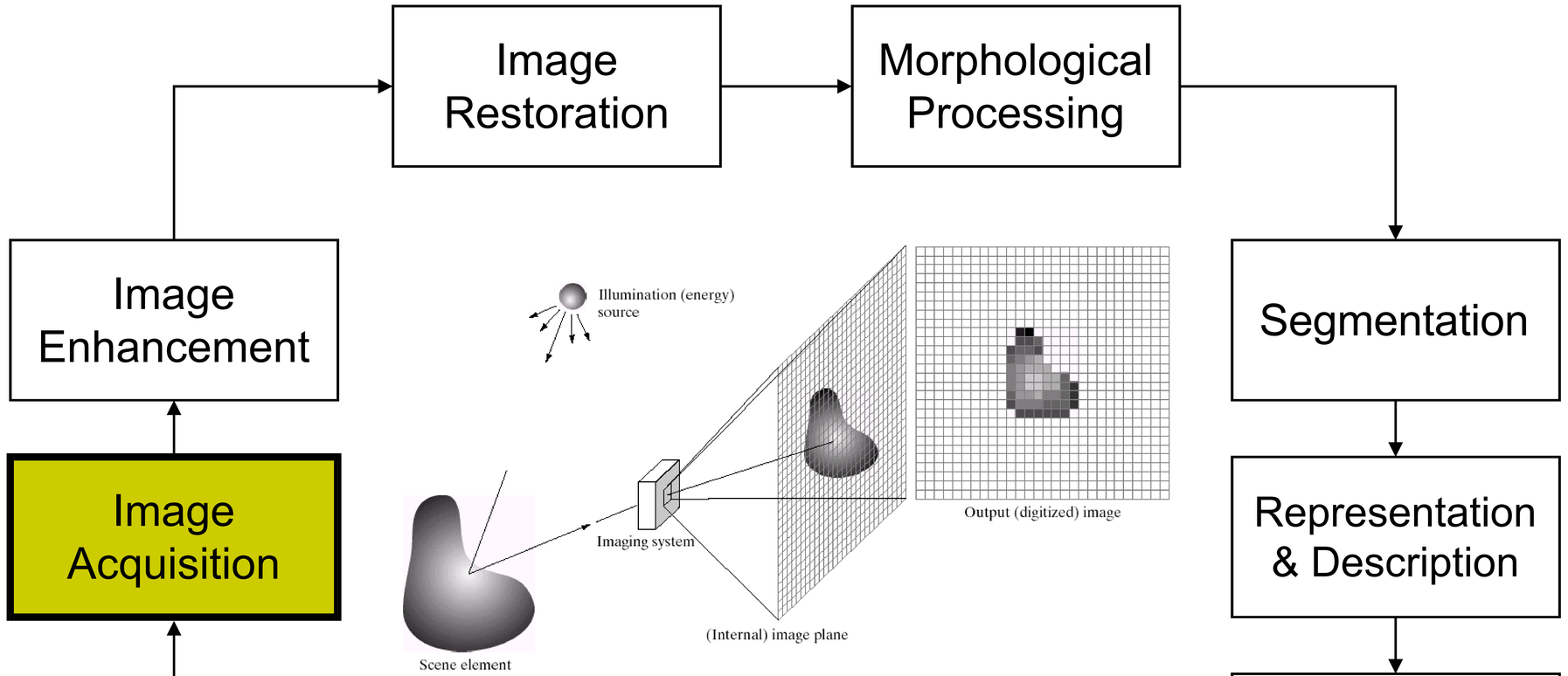
Image Processing

Image enhancement, noise removal, restoration,
feature detection, compression

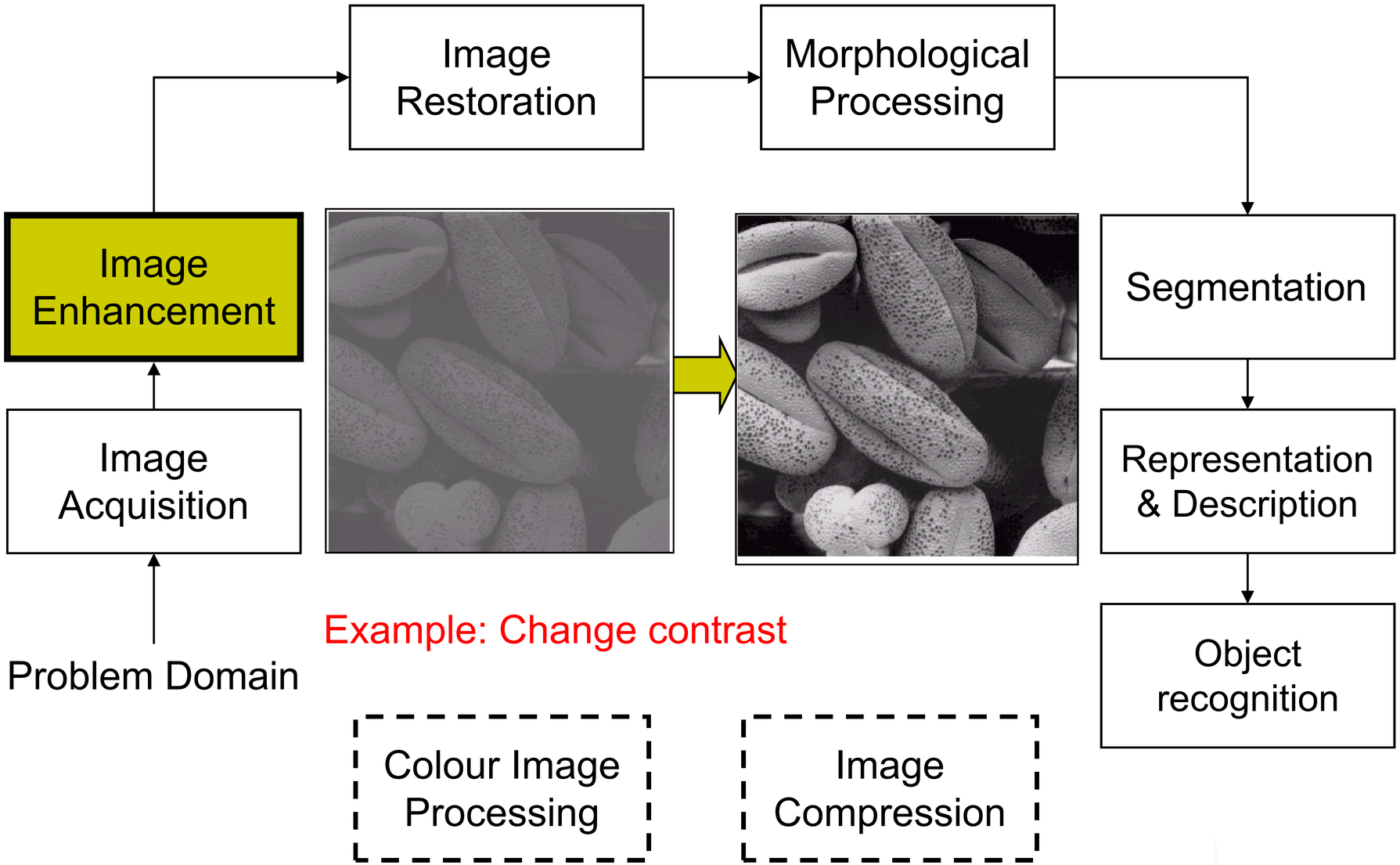
Key stags in digital image processing



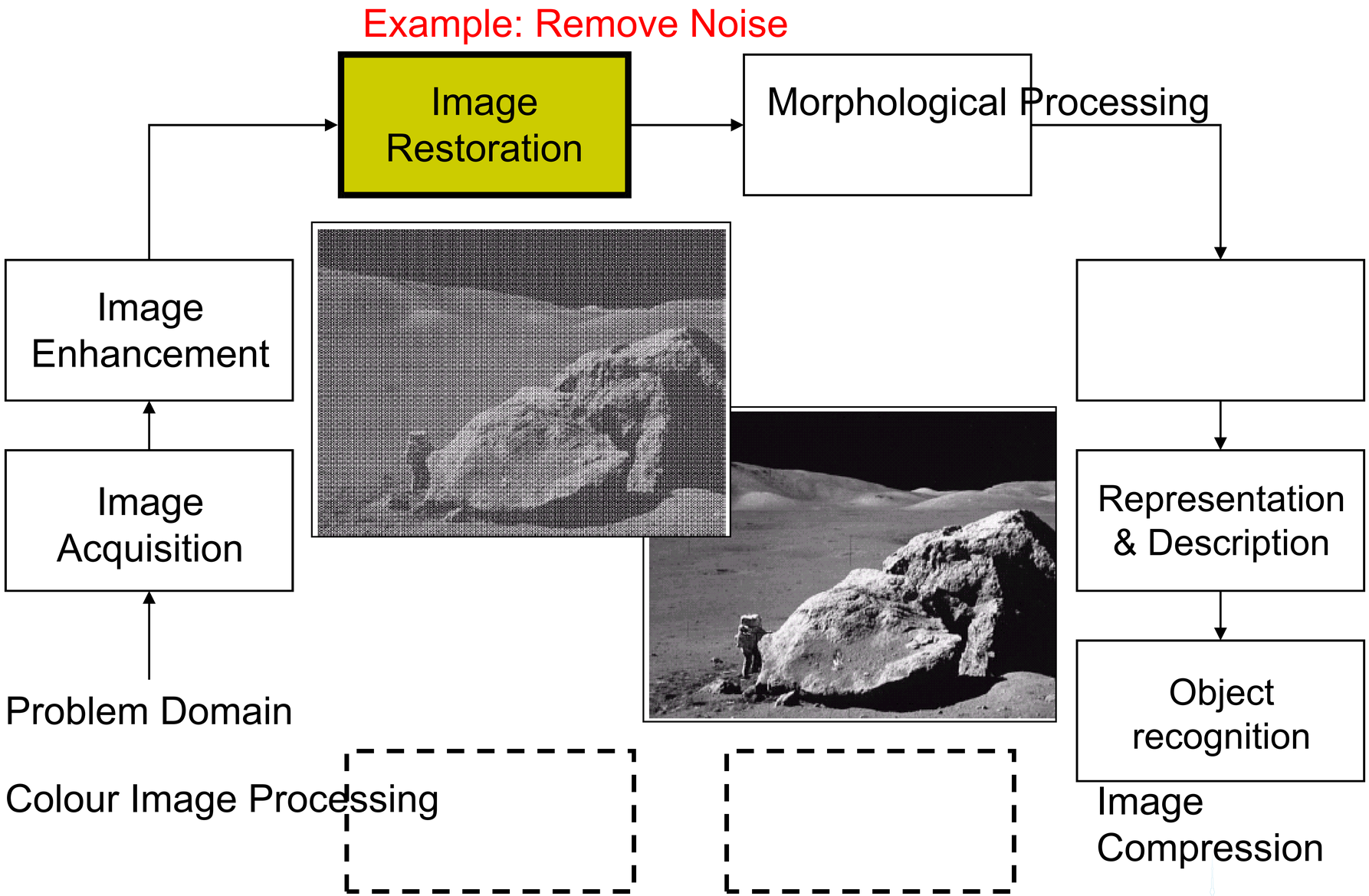
Key stags in digital image processing



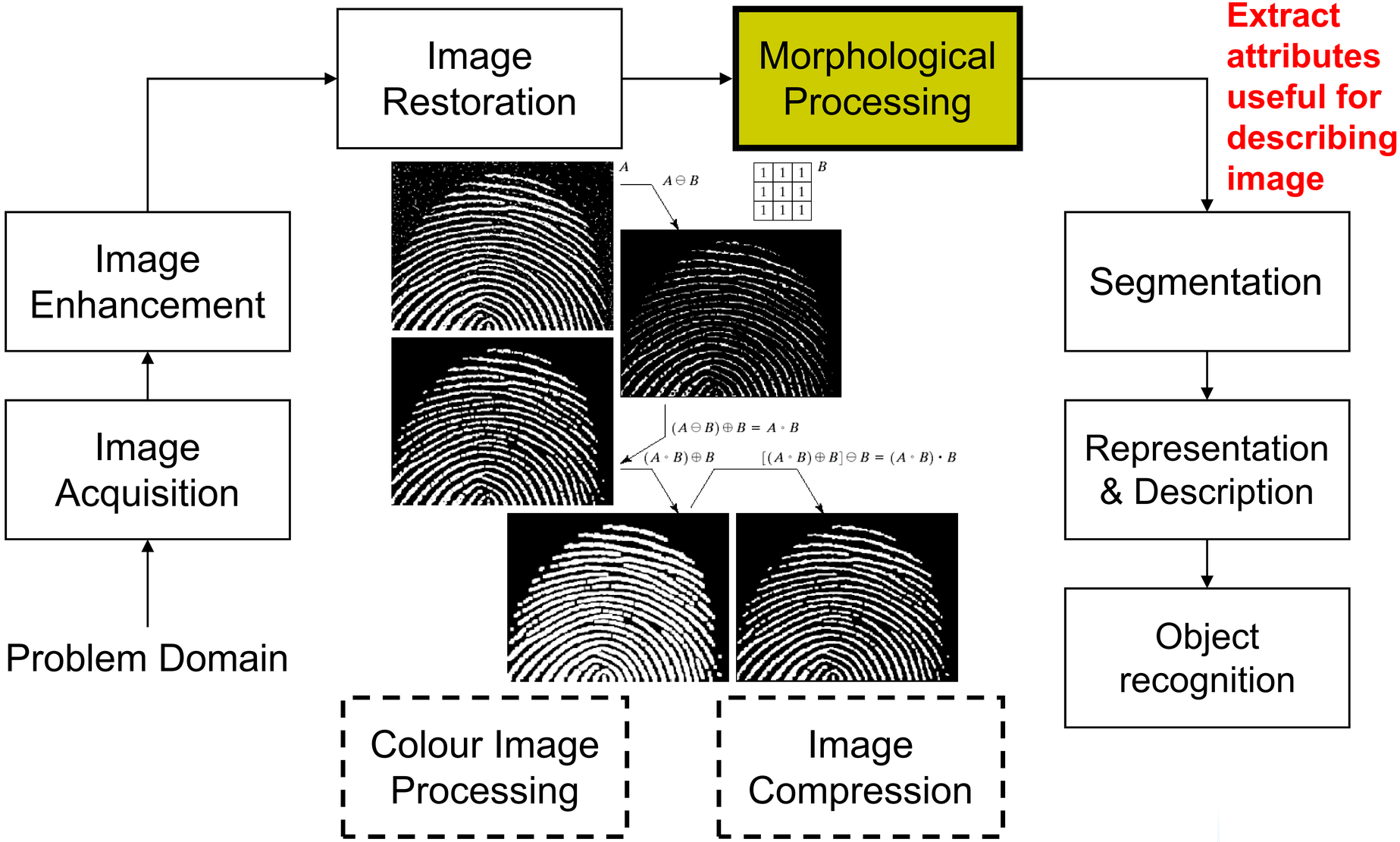
Key stags in digital image processing



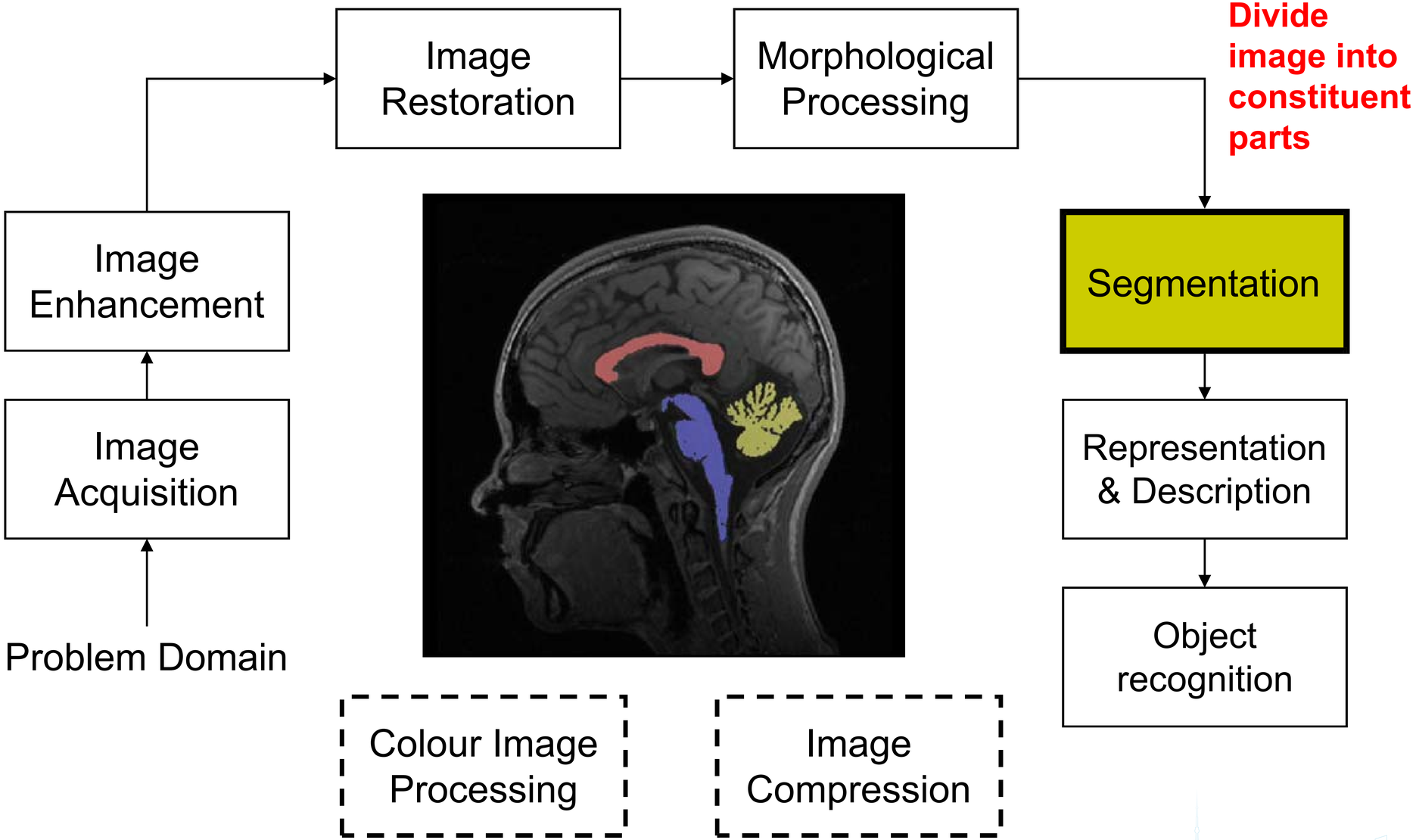
Key stags in digital image processing



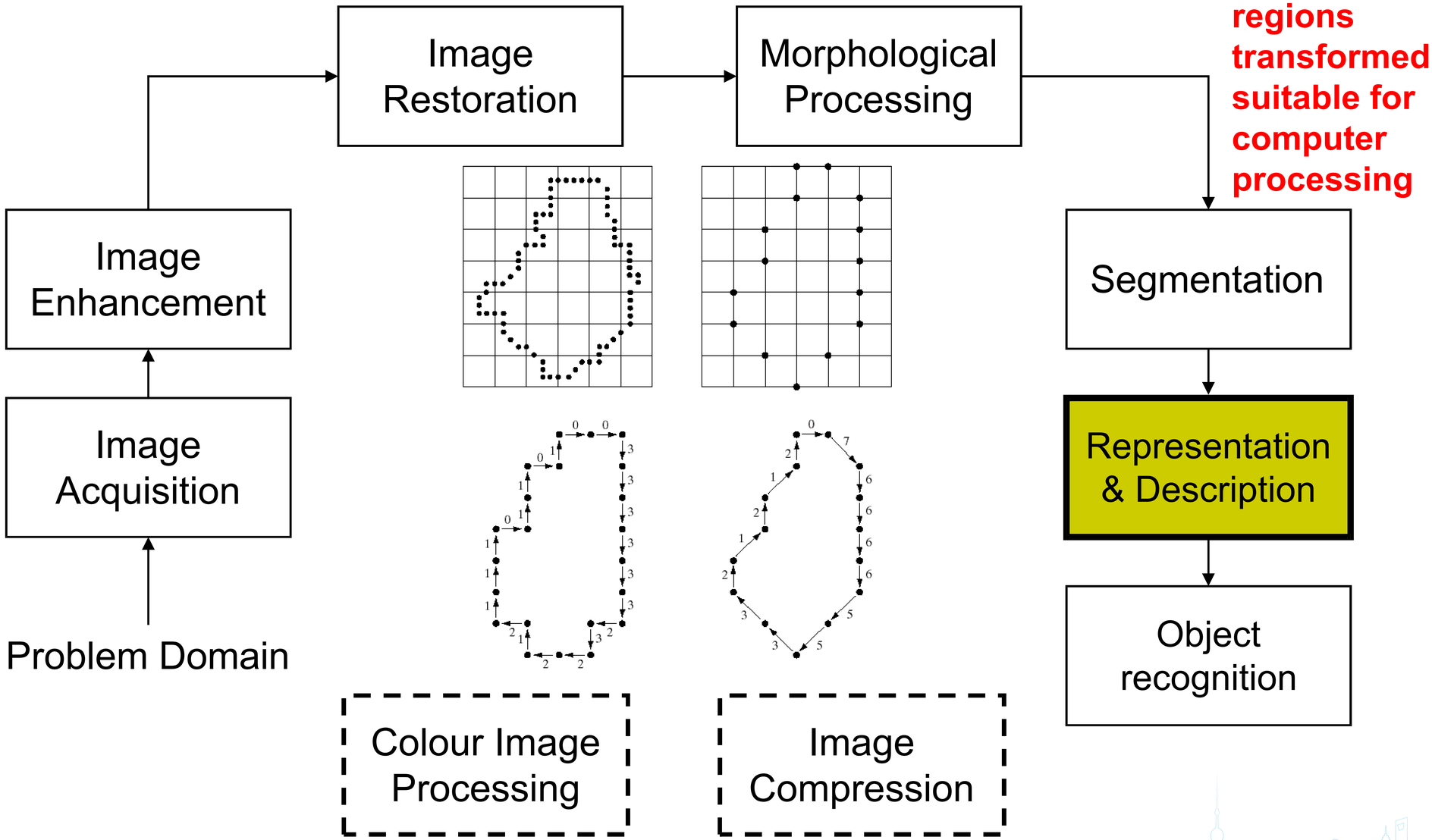
Key stags in digital image processing



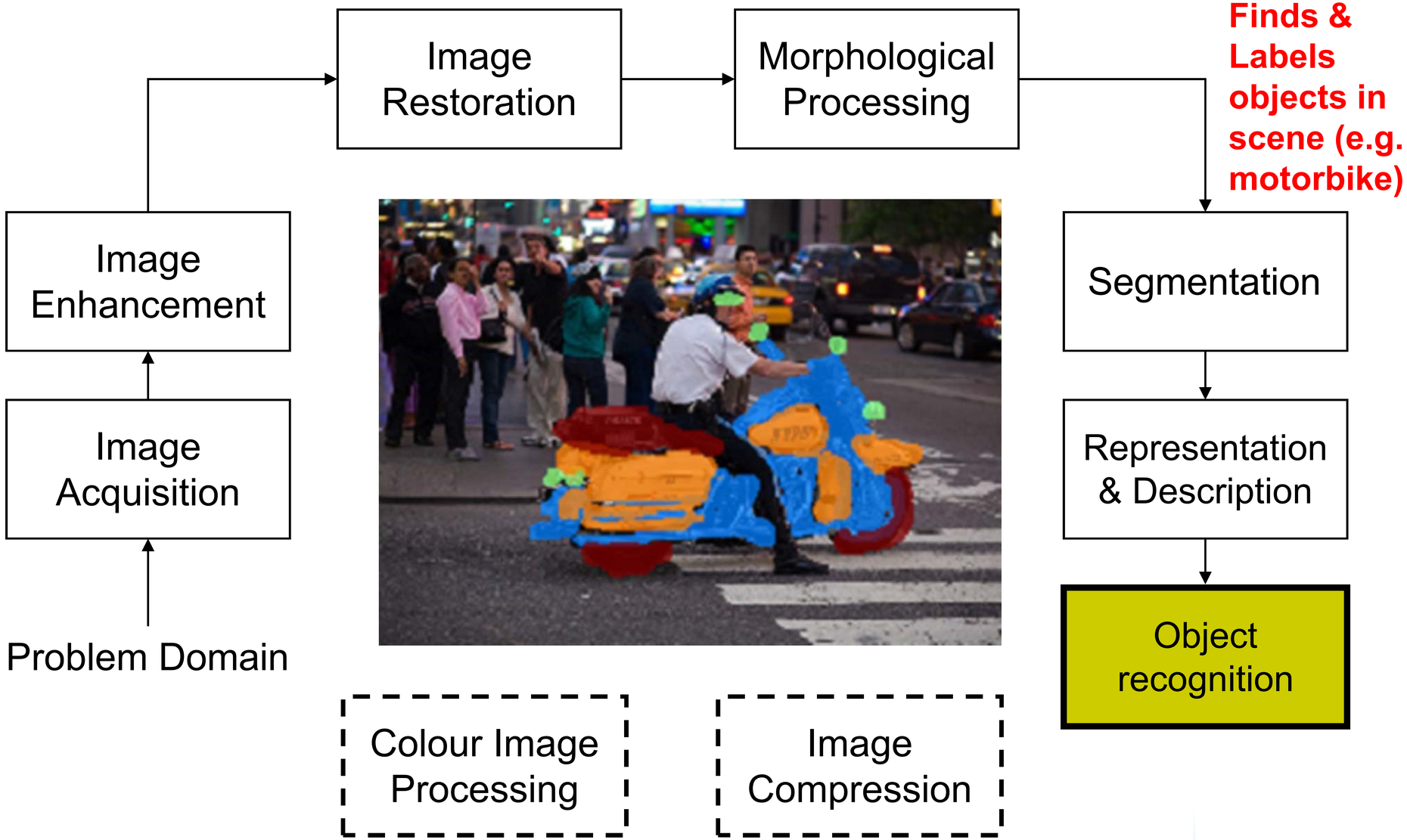
Key stags in digital image processing



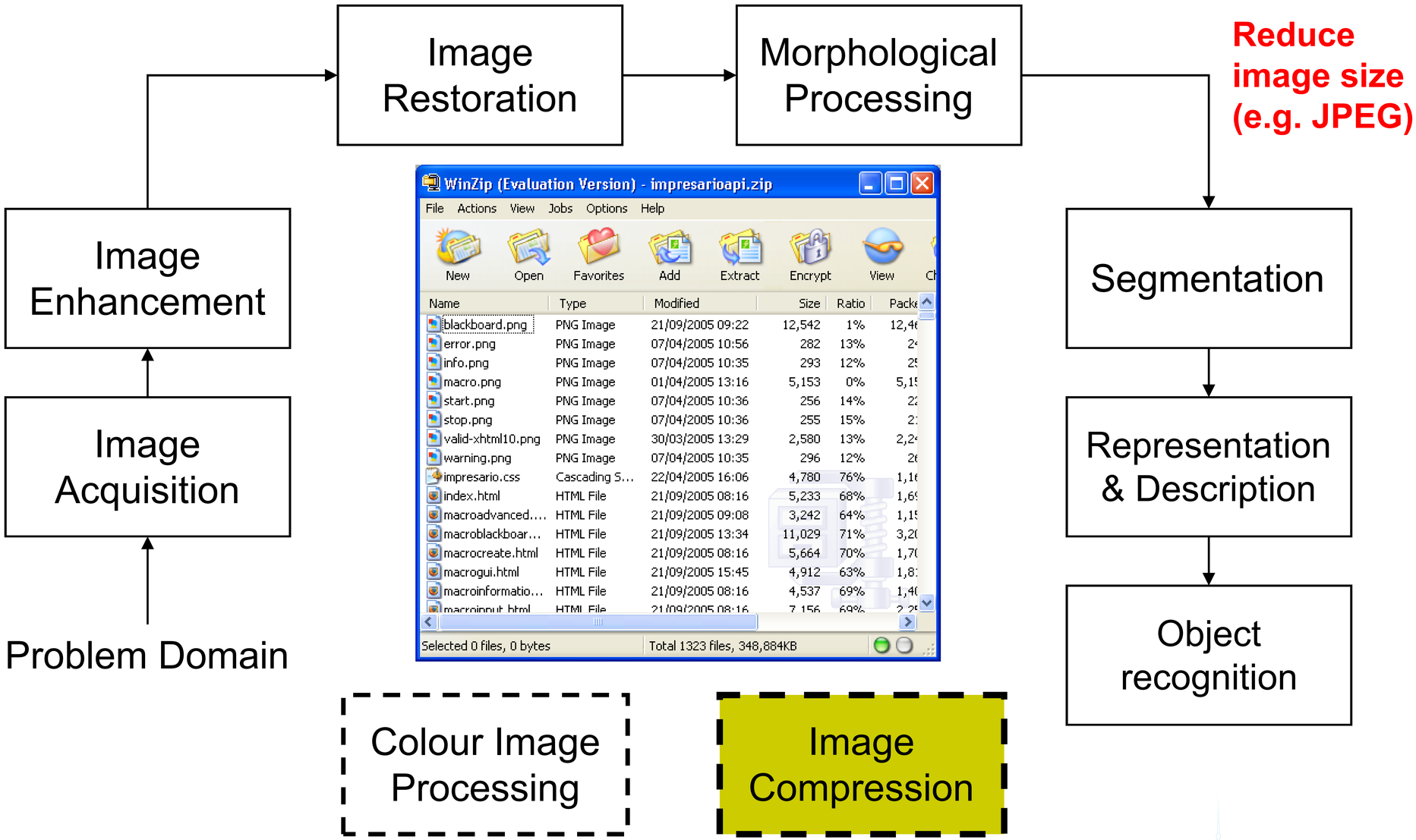
Key stags in digital image processing



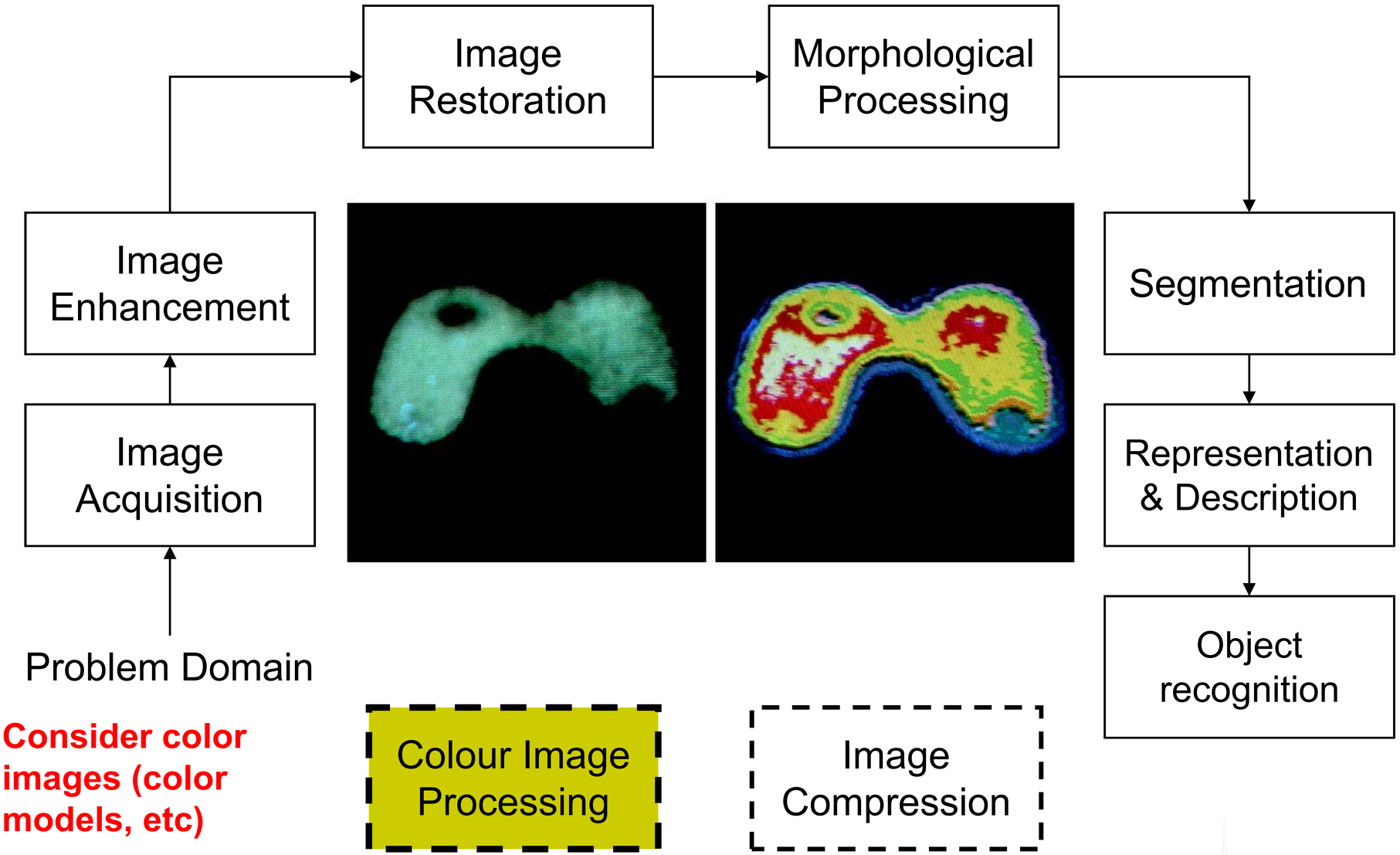
Key stags in digital image processing



Key stags in digital image processing



Key stags in digital image processing

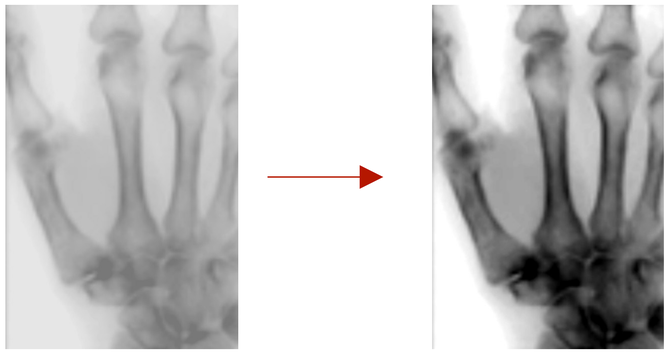


Outline: Image Data Mining

1. Introduction
2. Digital Image Processing
3. Convolutional Neural Network
4. Computer Vision Tasks

Pixel-level processing

Processing



Input image
(A_{in})

Output image
(A_{out})

Function $A_{out} = f(A_{in})$

Pixel Level

90	3	241	65	213
242	143	122	71	8
203	103	45	167	240
29	239	2	93	218
243	242	183	54	123

Special type of function

$$A_{out}[i,j] = f(A_{in}[i,j])$$

Pixel-level processing

- Linear **functions**
- Nonlinear **functions**

Pixel Level

90	3	241	65	213
242	143	122	71	8
203	103	45	167	240
29	239	2	93	218
243	242	183	54	123

Linear functions

Build functions in Numpy

Maths

$$= \begin{bmatrix} 20 & 22 \\ 1 & 25 \end{bmatrix}$$

$$= \quad + 2$$

Submatrix:
Row[0-3], Col[5-13]

Nested list implementation

```
I_list = [[20,22],[1,25]]
```

```
for row in range(2):
    for col in range(2):
        I_list[row][col] += 2
```

...

Numpy Implementation

```
import numpy as np
I_np = np.array([[20,22],[1,25]])
```

```
I_np += 2
```

```
I_np[0:3, 5:13]
```


Brightness adjustment: f=x+b

243	222	171
203	145	137
209	132	193

123	102	51
83	25	17
89	12	73

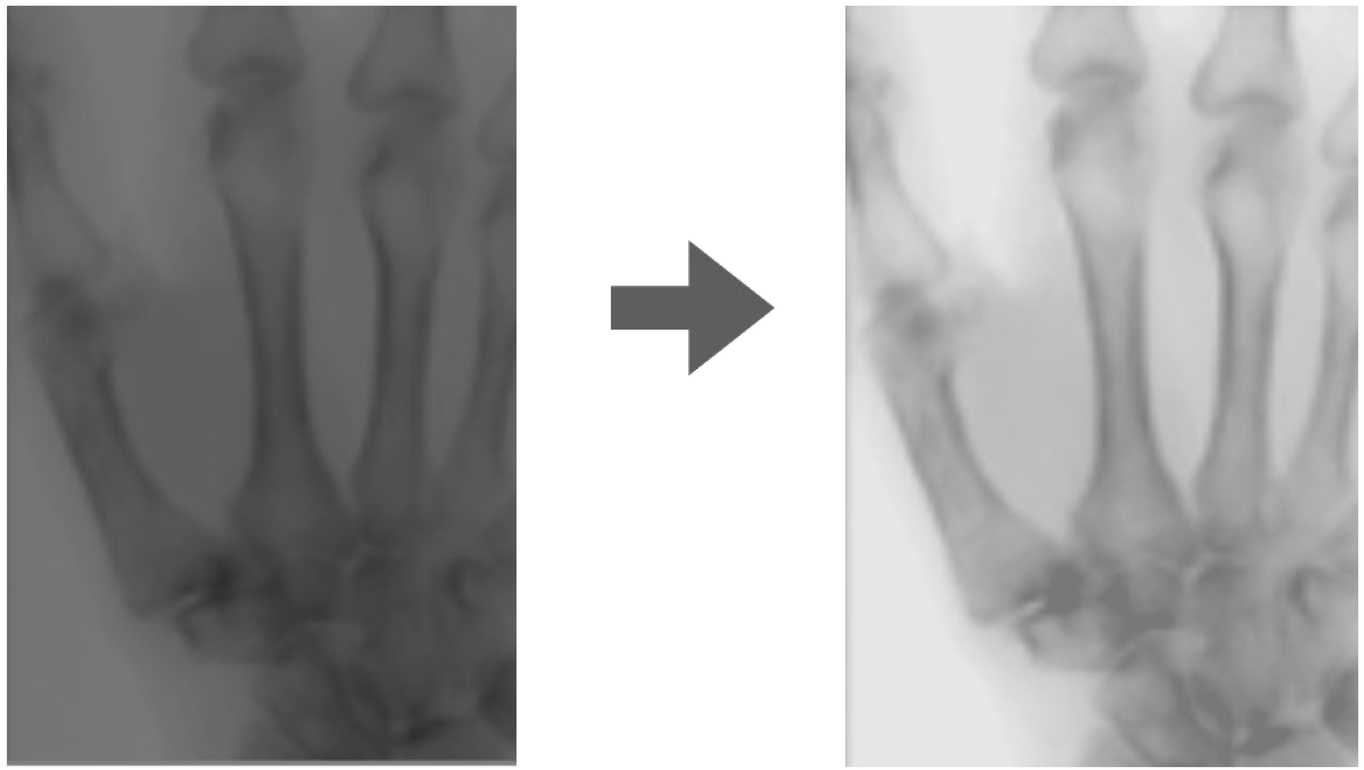
$p \longrightarrow p - 120$

Input pixel value → Output pixel value

```
I = imread('image.png')
I = I - 120
```

Contrast adjustment

High contrast: $(I_{max} - I_{min})$ is bigger



Contrast adjustment: f(x) = ax

121	111	85
101	72	68
104	66	96

242	222	170
202	144	136
208	132	192

p → $p * 2$

Input pixel value — Output pixel value

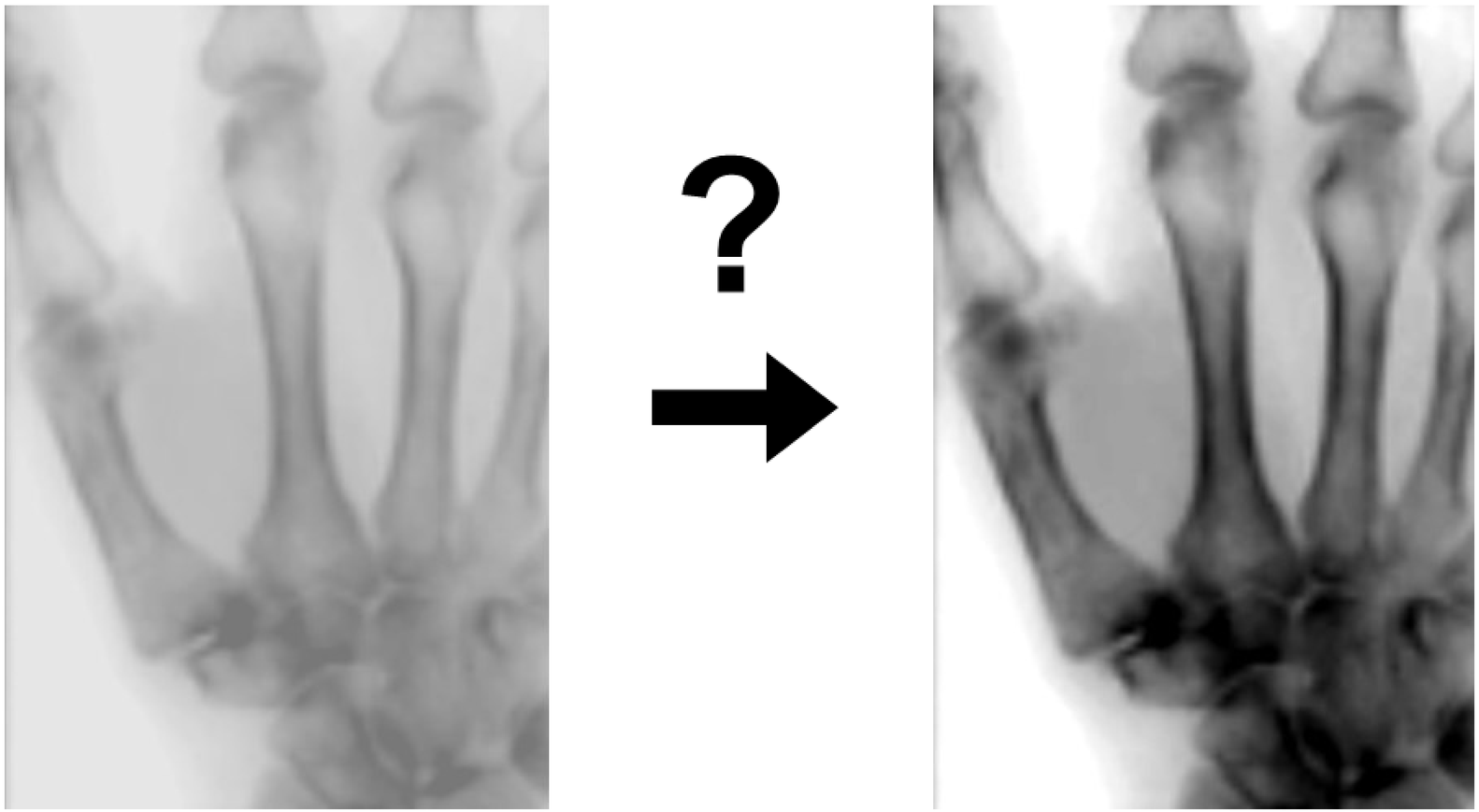
(max-min = 55) — (max-min = **110**)

Scale pixel value (image range is different)

```
I = imread('image.png')
```

```
I = I * 2
```

Application: image auto-contrast



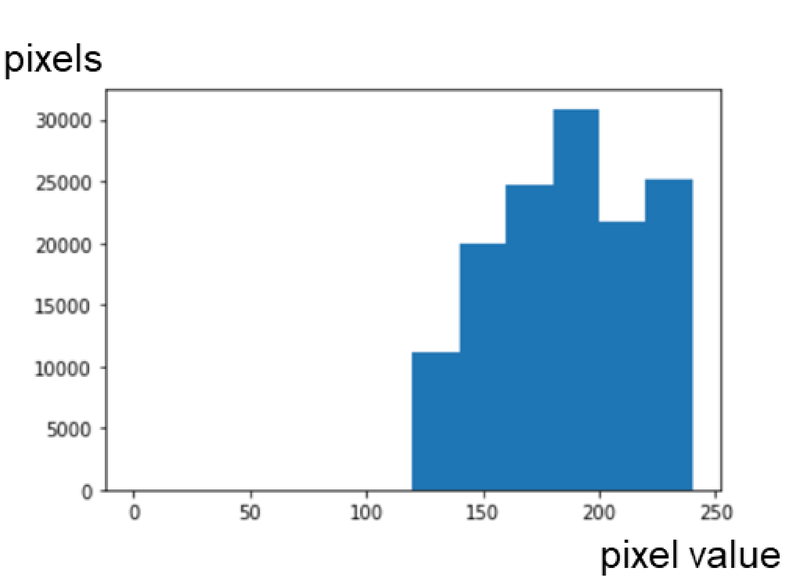
Auto-contrast is a technique used to enhance the contrast of an image automatically. It adjusts the intensity levels of the pixels to use the full range of available color values (usually 0 to 255 in an 8-bit image)

Application: image auto-contrast

Image: Statistical view

243	222	171
203	145	137
209	132	193

,



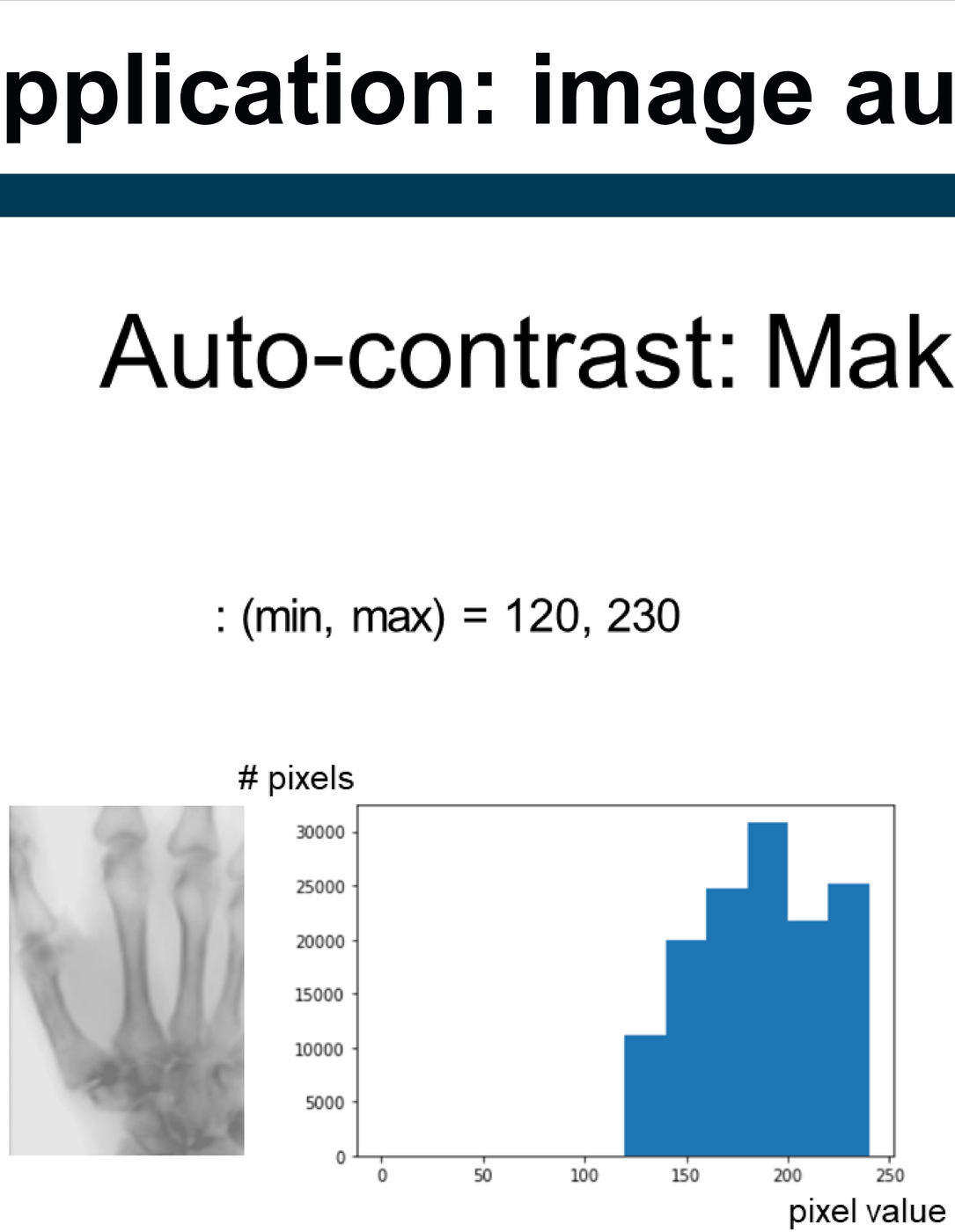
Pixel value histogram

Application: image auto-contrast

Auto-contrast: Make use of the full range

: (min, max) = 120, 230

: (min, max) = 0, 255



Pixel value histogram

Application: image auto-contrast

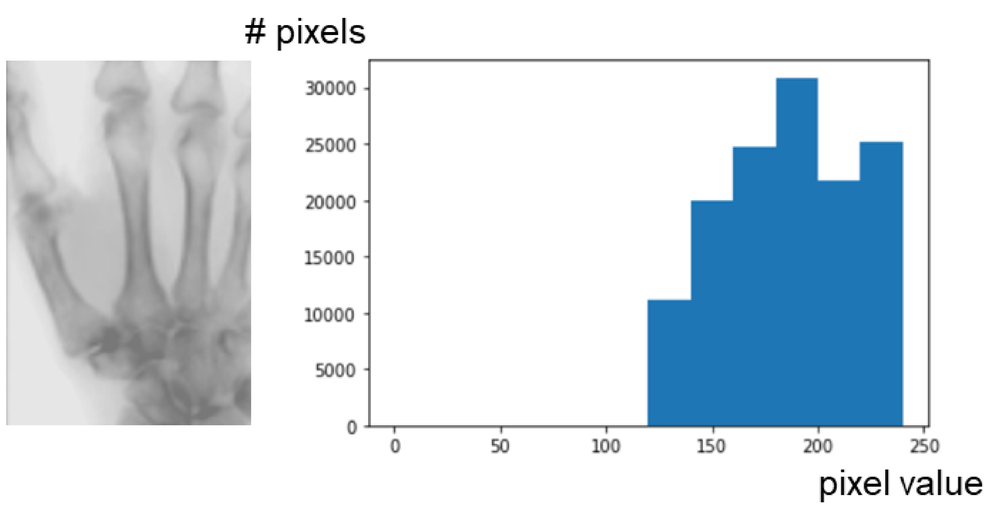
Step 1: make I_{min} to be 0

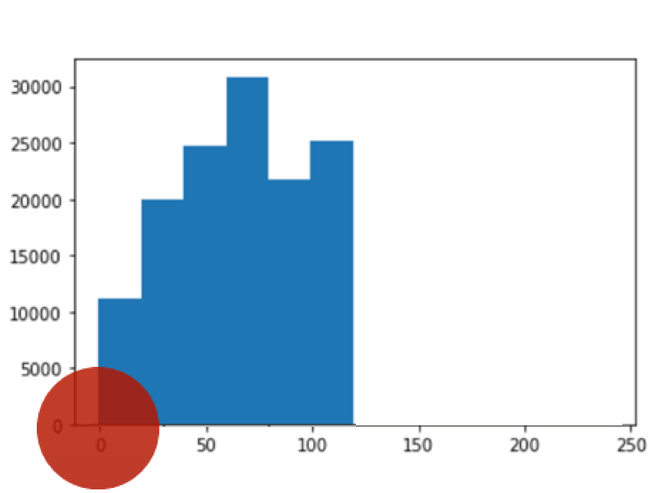
: (min, max) = 120, 230

```
I = I - 120
```

(min, max)

0, 110





Pixel value histogram

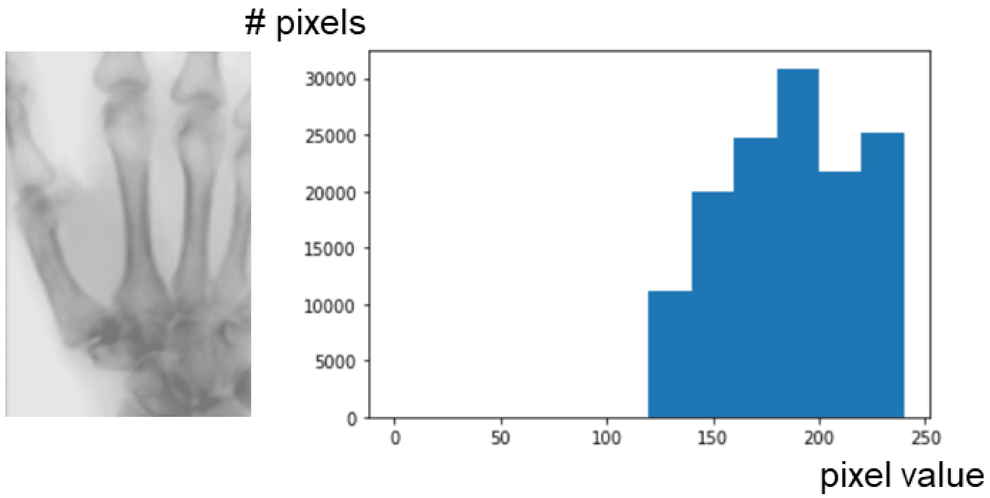
Application: image auto-contrast

Step 2: make I_{max} to be 255

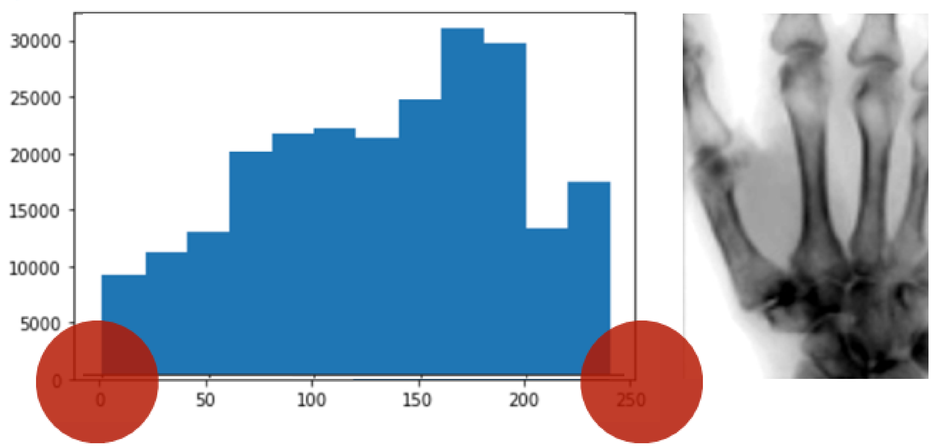
(min, max)

: (min, max) = 120, 230

```
I = I - 120                          0, 110
I = I / (I.max() - I.min())          0, 1
I = I * 255                          0, 255
```



Pixel value histogram

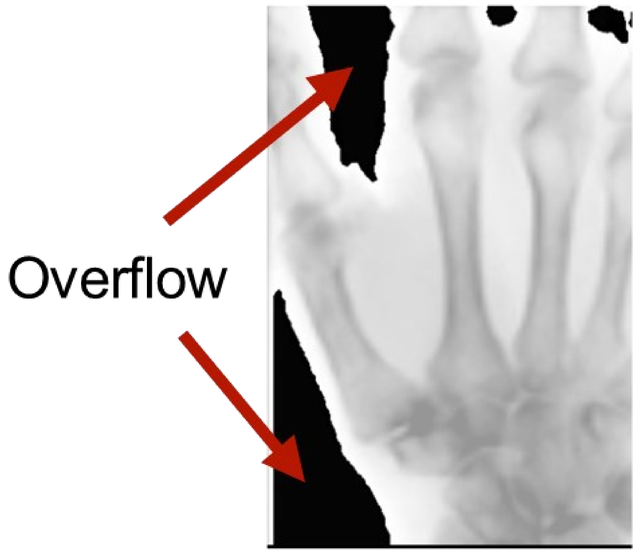


Linear functions

Implementation: uint8 arithmetic

100 + 200 = ?

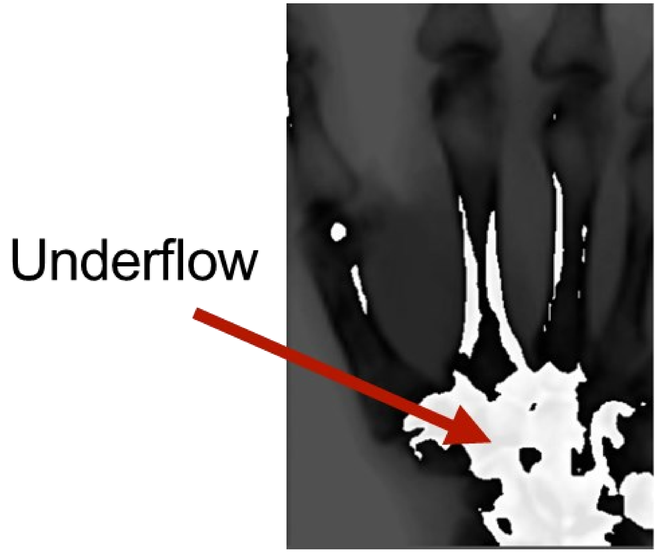
Overflow: 300-256 = **44**




```
I = I + 30
```

100 - 120 = ?

Underflow: -20+256 = **236**




```
I = I - 150
```

Image arithmetic

- convert to “Float”
- do arithmetic
- clip it [0,255]

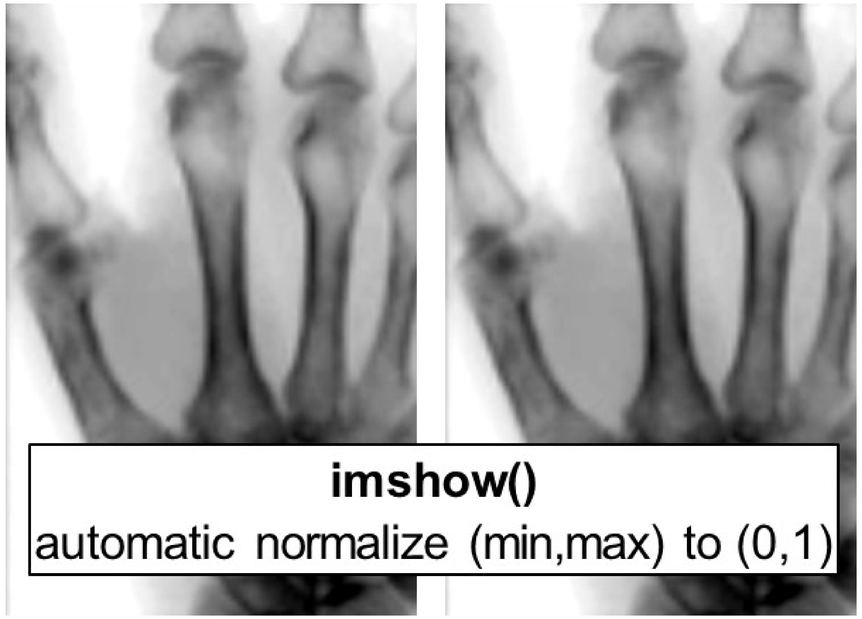
```
I.astype(float)
```

```
np.clip(im,0,255).astype(np.uint8)
```

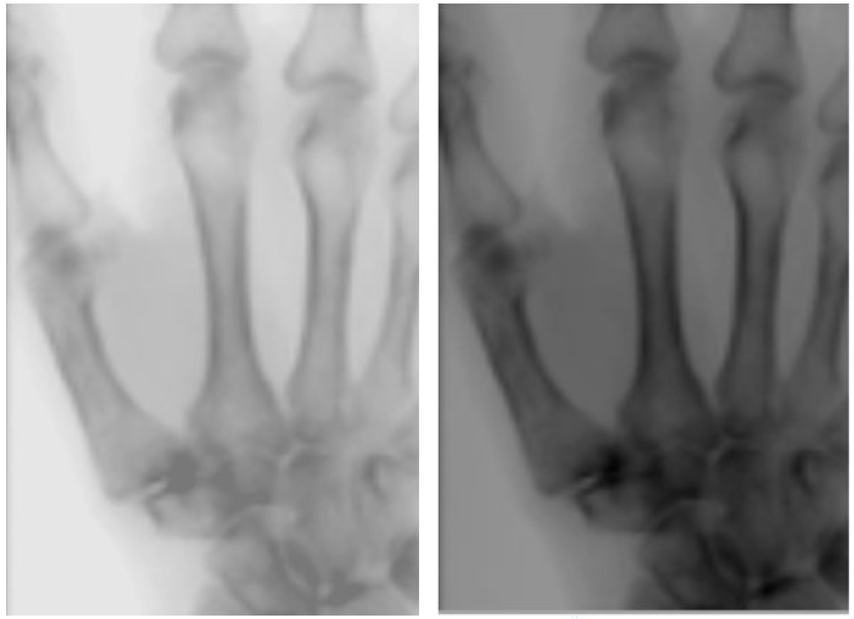
Linear functions

Implementation: plt.imshow()

```
plt.subplot(121)
plt.imshow(image, cmap='gray')
plt.axis('off')
plt.subplot(122)
plt.imshow(image - 100, cmap='gray')
plt.axis('off')
```

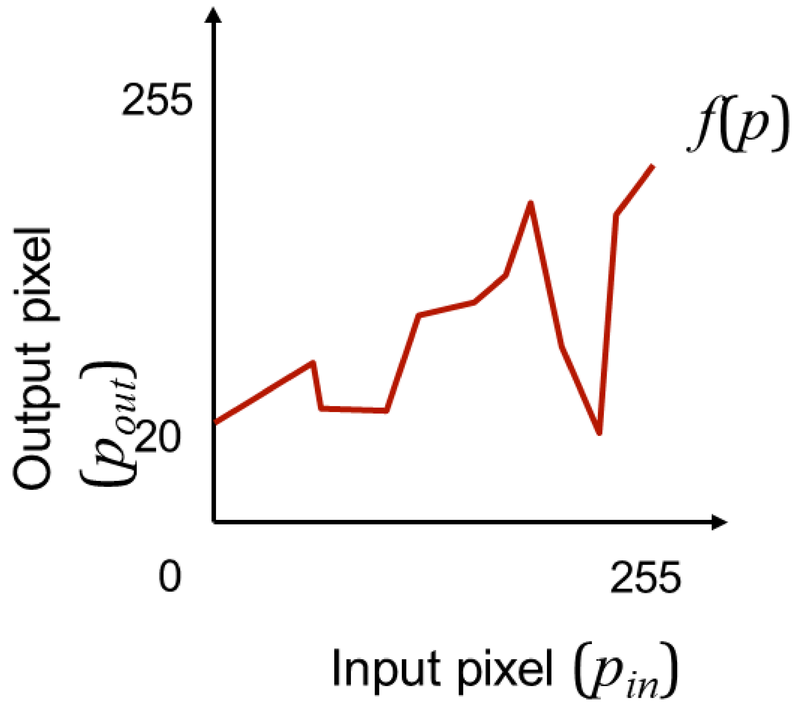



```
plt.subplot(121)
plt.imshow(image, cmap='gray', vmin=0, vmax=255)
plt.axis('off')
plt.subplot(122)
plt.imshow(image - 100, cmap='gray', vmin=0, vmax=255)
plt.axis('off')
```

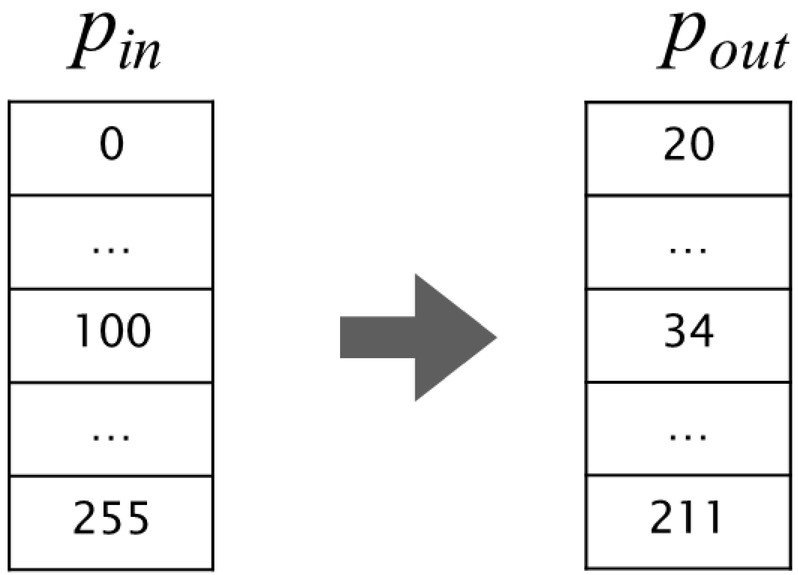


Linear transfer functions

Visualization

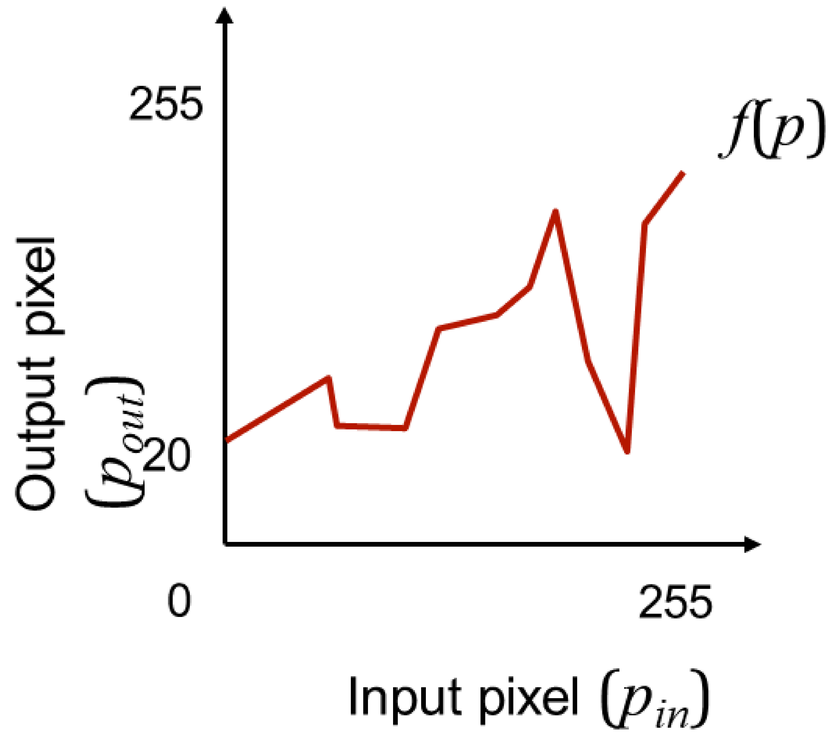


Example: an arbitrary function



Nonlinear transfer functions

Visualization

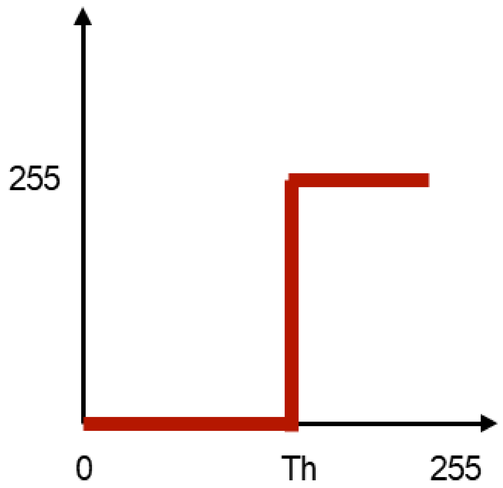


Example: an arbitrary function

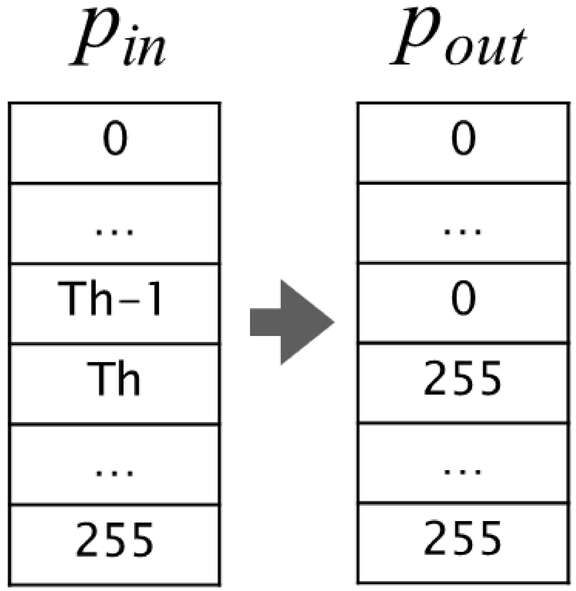
p_{in}	p_{out}
0	20
…	…
100	34
…	…
255	211

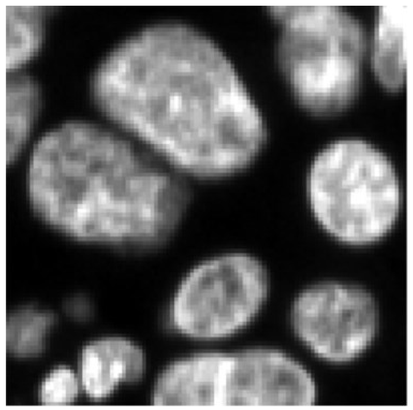
Application: image thresholding

```
I[I >= Th] = 255
I[I < Th]  =  0
```

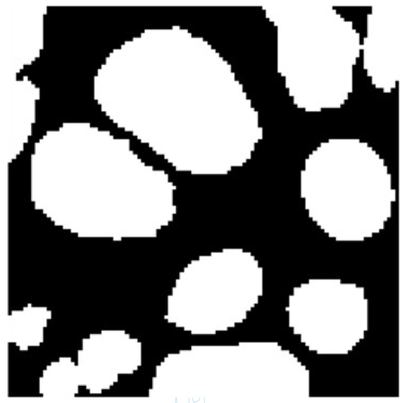


Binary segmentation:
Find the threshold





Image

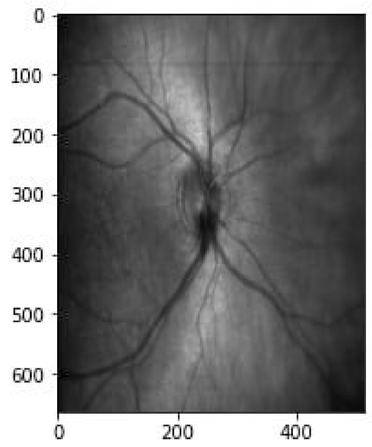


Binary segmentation

Application: image enhancement

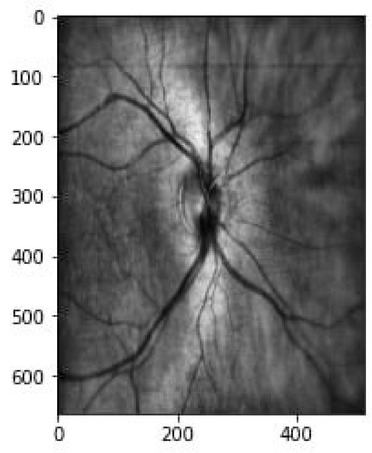
```
In [107]: 1 plt.imshow(dataf['slo_fundus'], cmap='gray')
```

```
Out[107]: <matplotlib.image.AxesImage at 0x7f64260349a0>
```




```
In [108]: 1 clahe = cv2.createCLAHE(clipLimit=4.0, tileGridSize=(8,8))
          2 fundus_slo_enhance = clahe.apply(dataf['slo_fundus'])
          3
          4 plt.imshow(fundus_slo_enhance, cmap='gray')
```

```
Out[108]: <matplotlib.image.AxesImage at 0x7f6425798fd0>
```



Patch-level processing

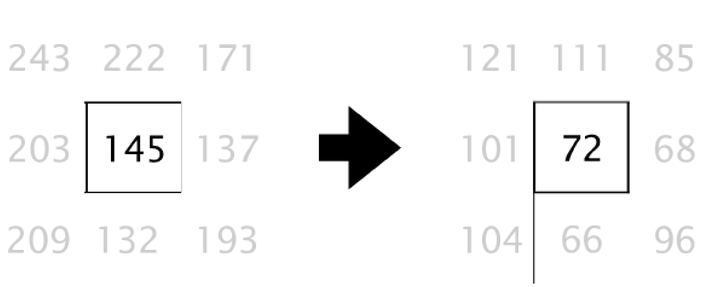
Pixel-level
(Mon)

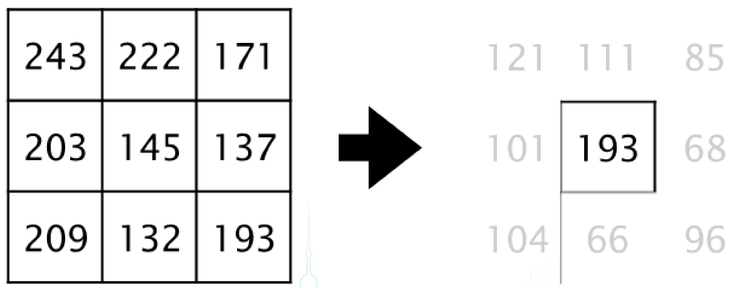
90	3	241	65	213
242	143	122	71	8
203	103	45	167	240
29	239	2	93	218
243	242	183	54	123

Patch-level
(Today)

90	3	241	65	213
242	143	122	71	8
203	103	45	167	240
29	239	2	93	218
243	242	183	54	123

Intuition: Need bigger input domain (patch)





Patch-level processing: image noise

Clean image

Noisy image
(Salt&pepper noise)

$$f : p \rightarrow p'$$

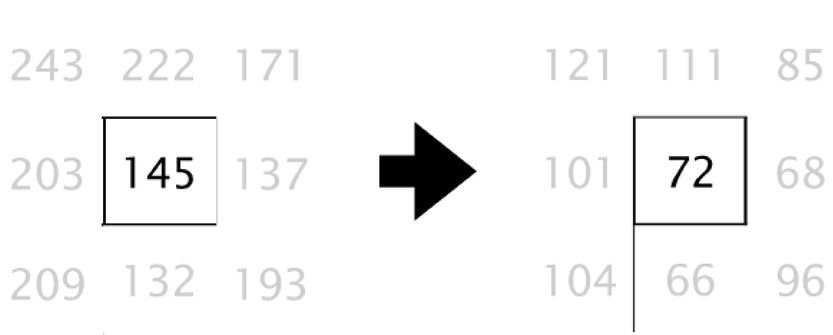
Input pixel	Output pixel
0	0
…	…
100	50

Transfer function doesn't help

- Same pixel value has different "ideal" values

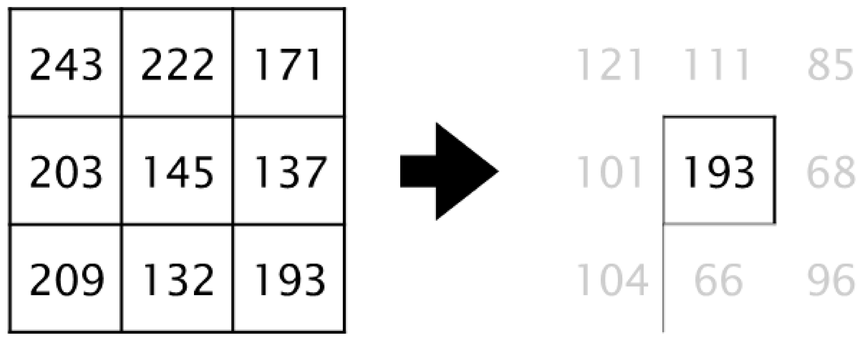
Patch-level processing

Maths: make a patch function



$$f : p \rightarrow p'$$

Input pixel Output pixel

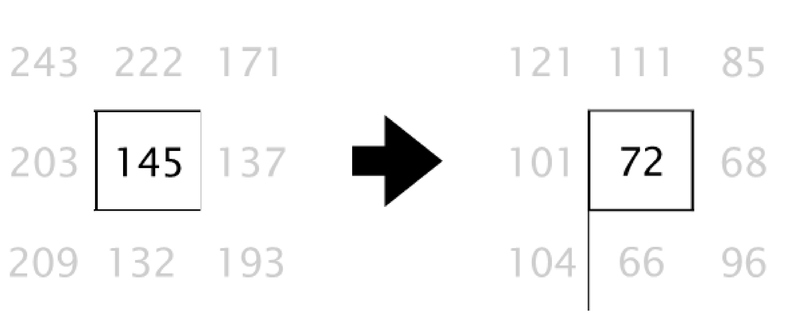


$$f : pa \rightarrow p'$$

Input **patch** Output pixel

Patch-level processing

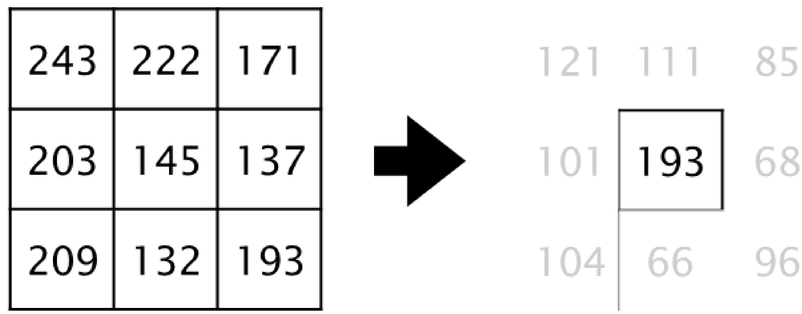
Example: Median filter



$$f : p \rightarrow p'$$

Input pixel	Output pixel
0	0
…	…
100	50

Transfer function (lookup table)



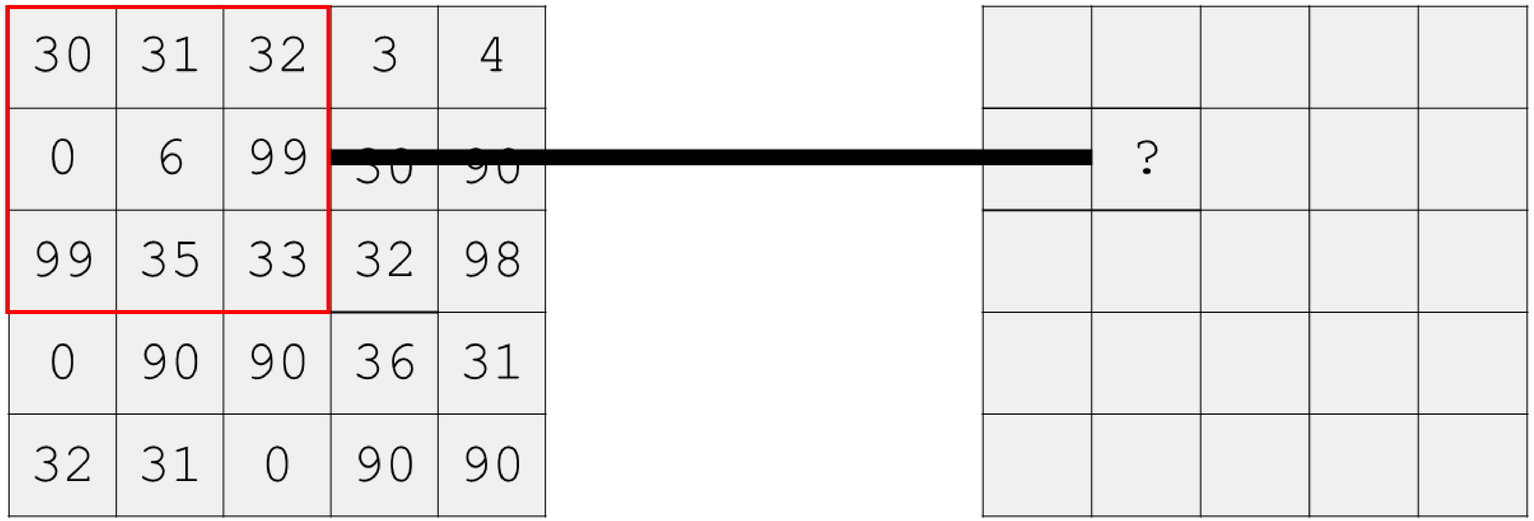
$$f : pa \rightarrow p'$$

Input **patch** Output pixel

$$p' = \text{median}(pa)$$

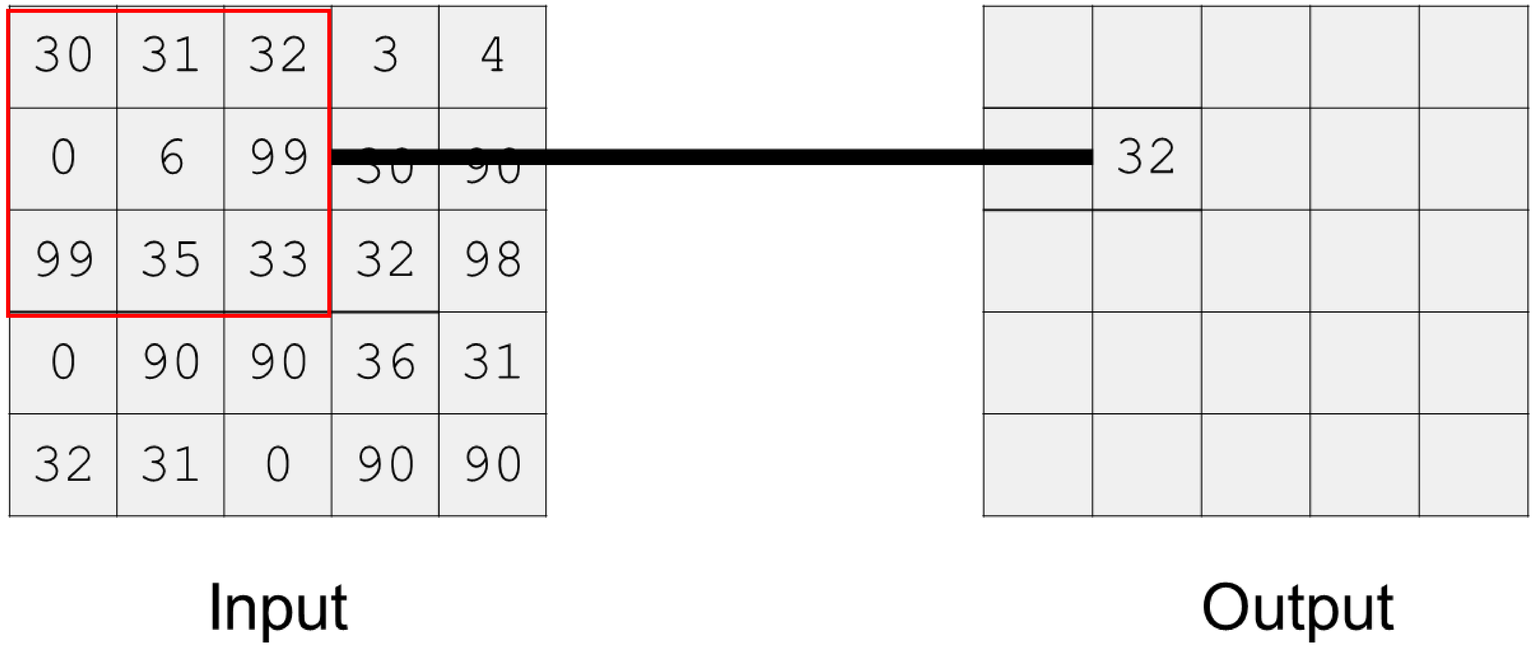
Patch-level processing

Example: Median filter



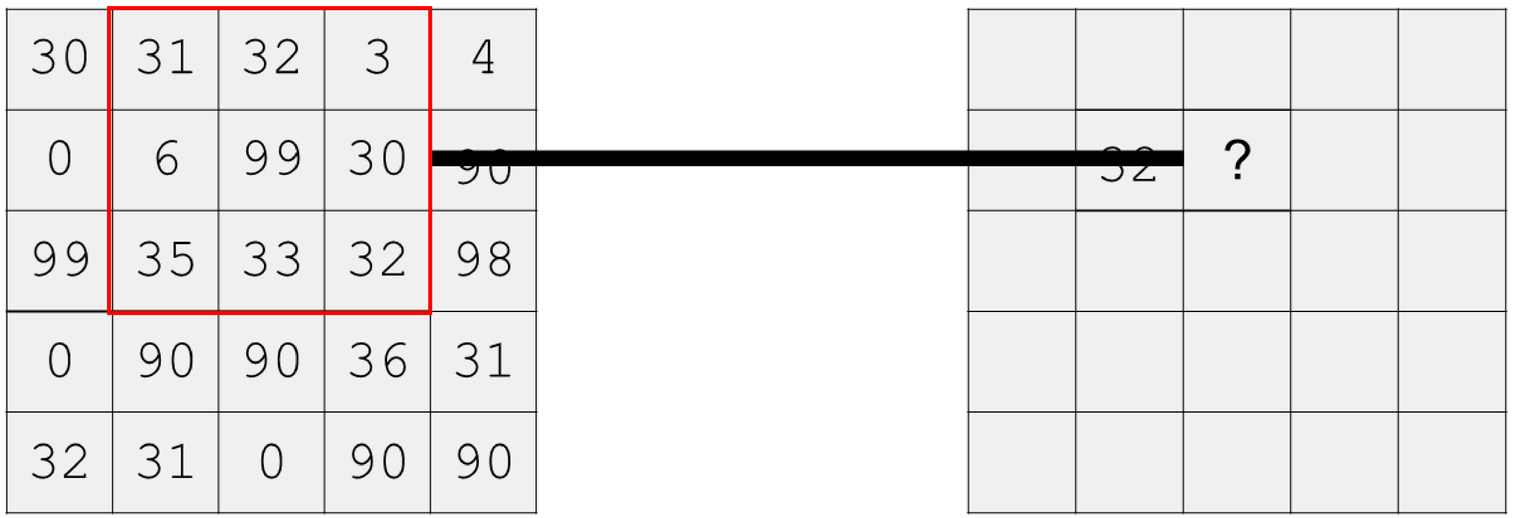
Patch-level processing

Example: Median filter



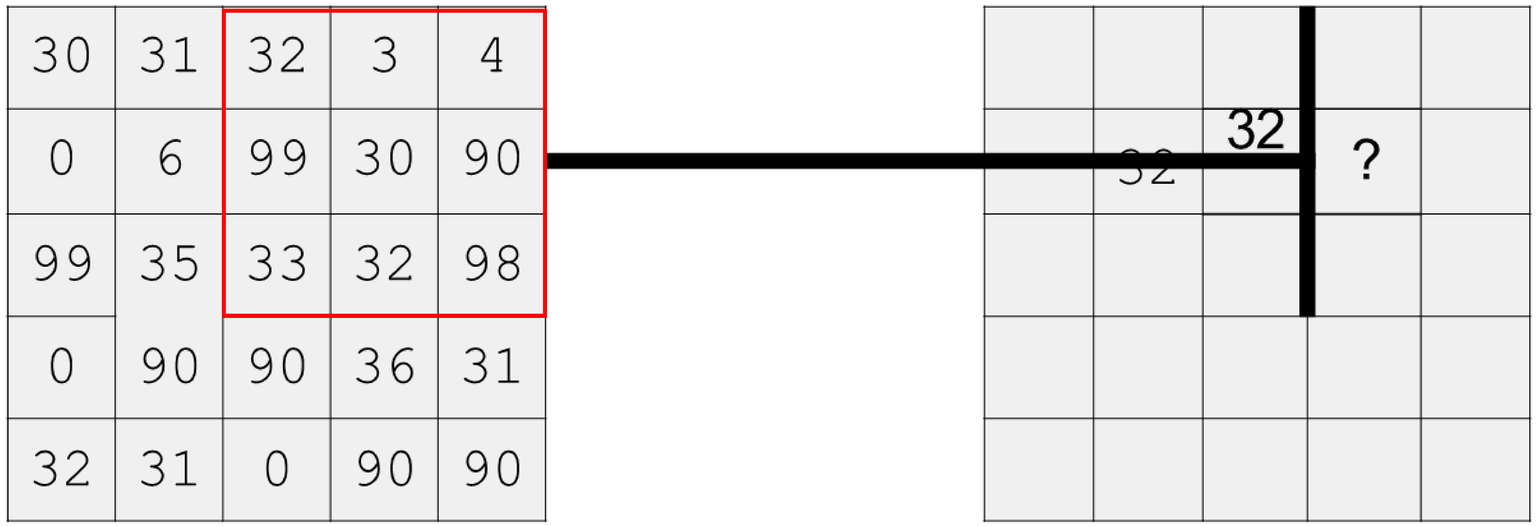
Patch-level processing

Loop through each column



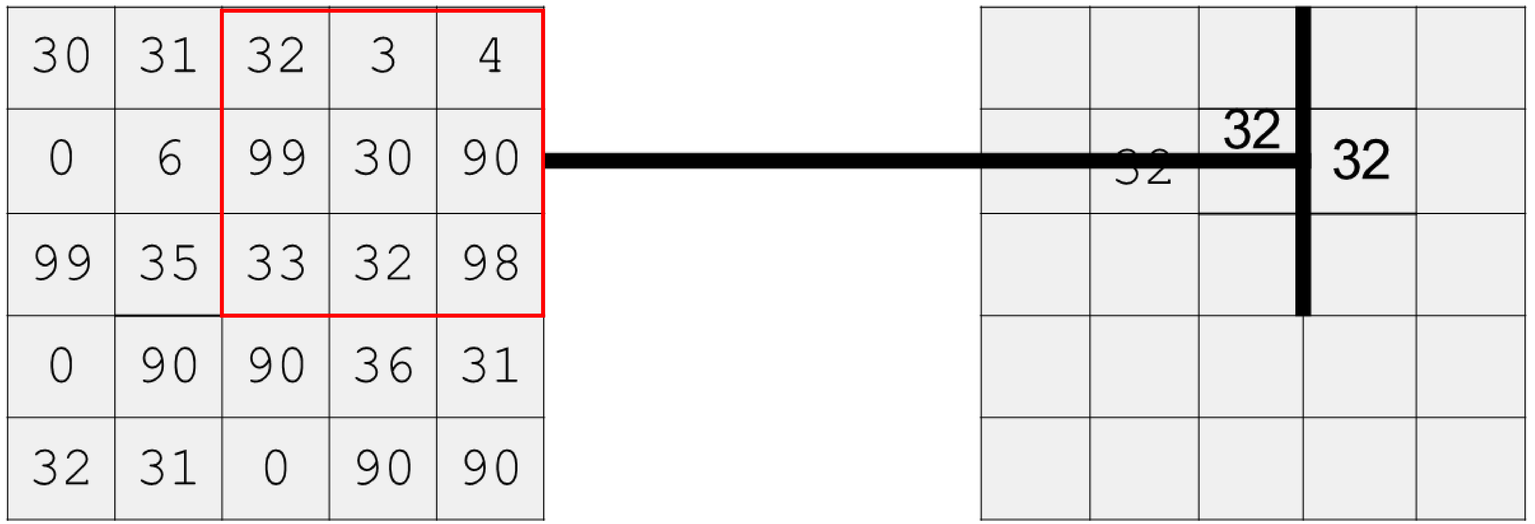
Patch-level processing

Loop through each column



Patch-level processing

Loop through each column



Patch-level processing

Loop through each row

30	31	32	3	4
0	6	99	30	90
99	35	33	32	98
0	90	90	36	31
32	31	0	90	90

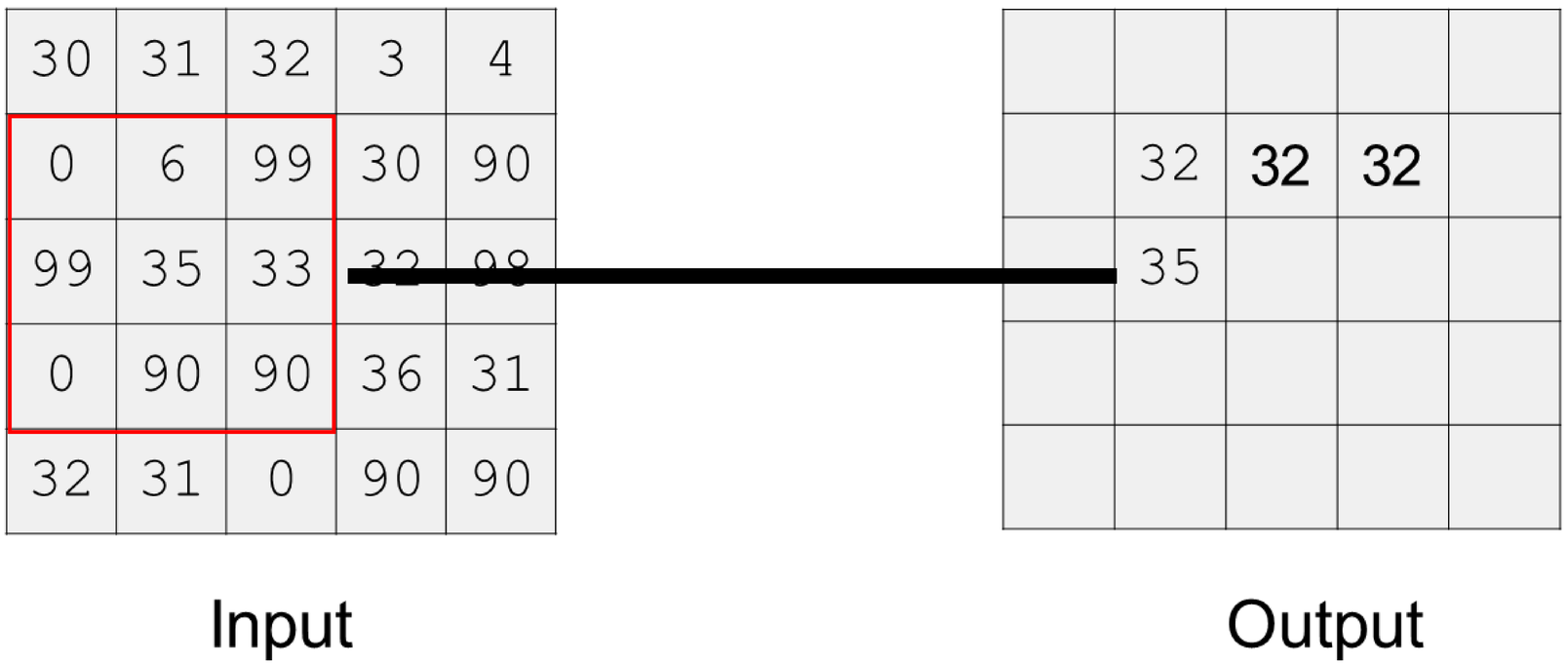
Input

	32	32	32	
	?			

Output

Patch-level processing

Loop through each row



Patch-level processing

Loop through each row

30	31	32	3	4
0	6	99	30	90
99	35	33	32	98
0	90	90	36	31
32	31	0	90	90

Input

	32	32	32	
	35	35	36	
			36	

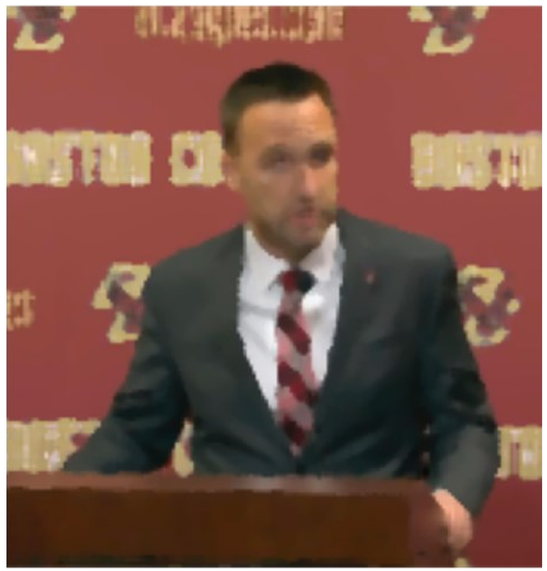
Output

Patch-level processing

Median filter: image denoising

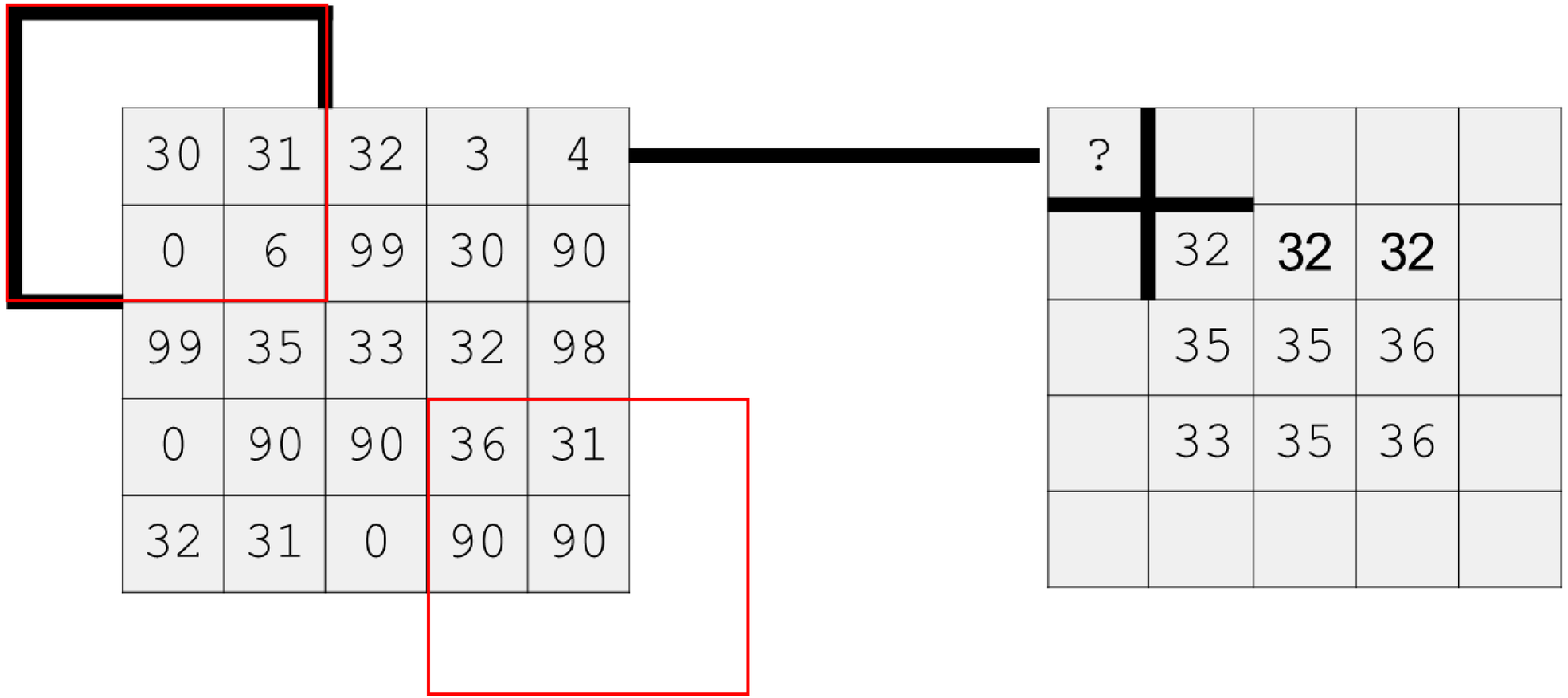
Ideal image

Real-world image



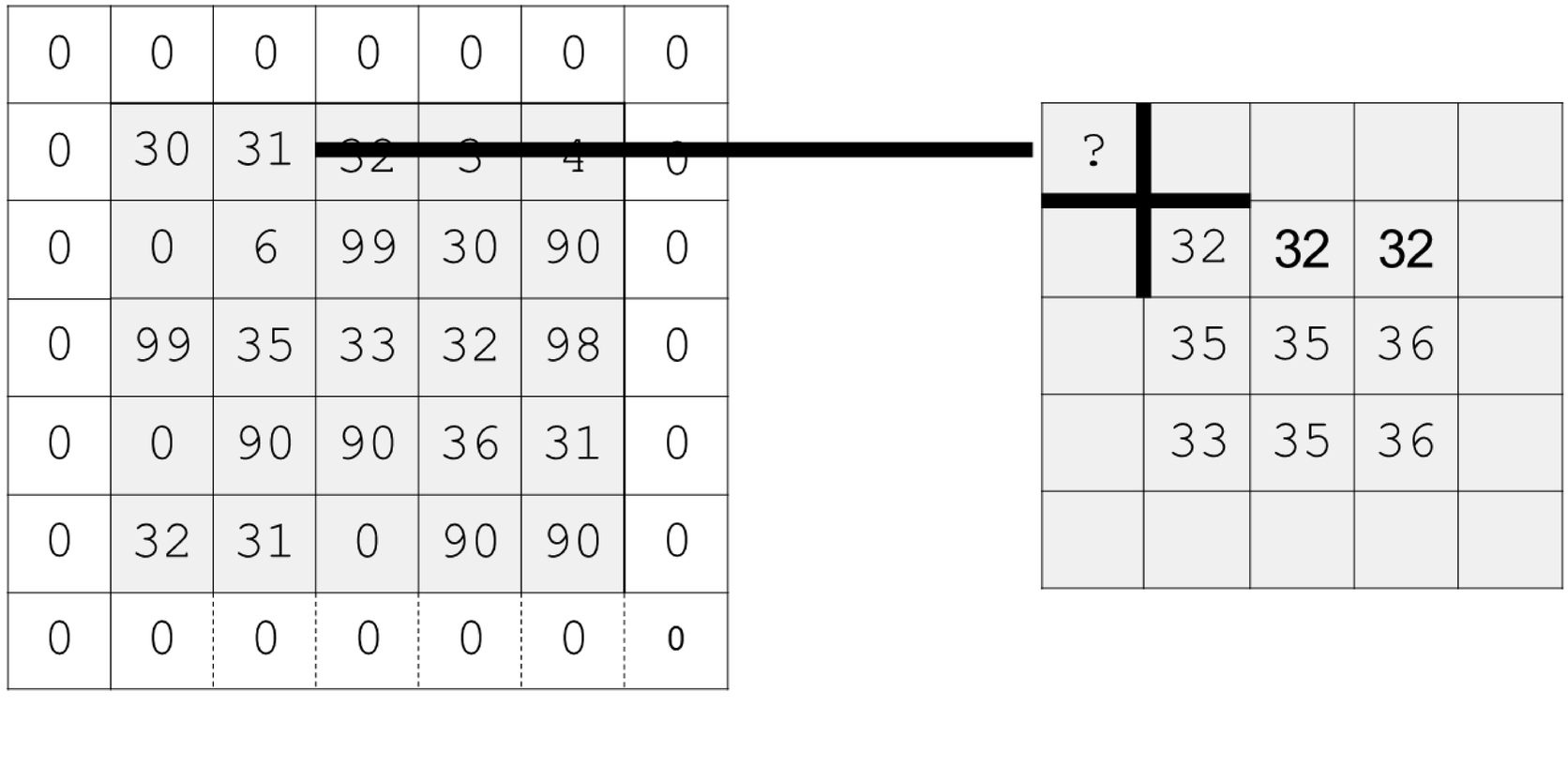
Median filter result

Patch-level processing



Patch-level processing

Solution: Pad the input image

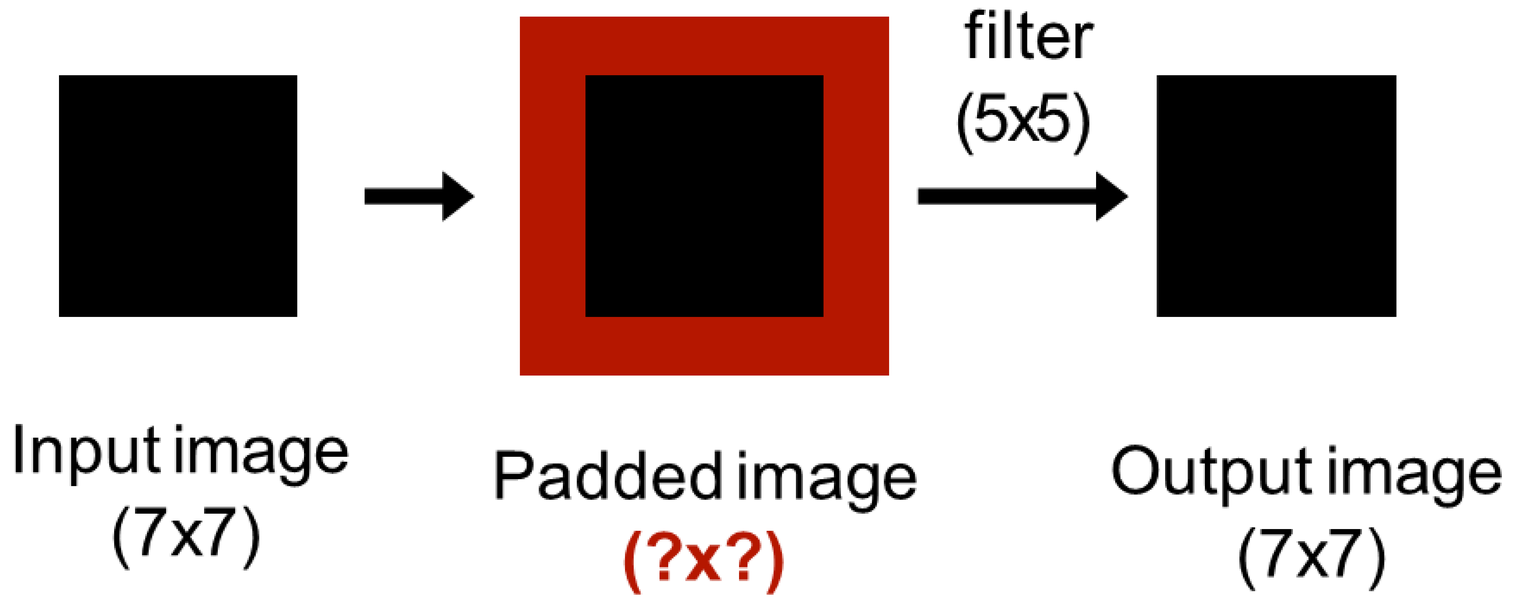


Input

Output

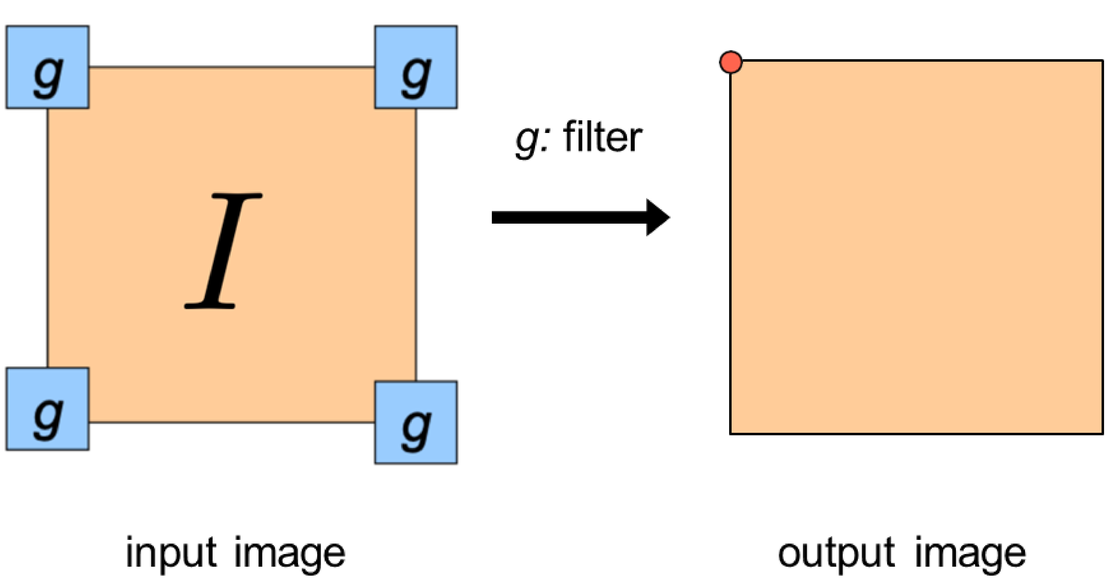
Patch-level processing: padding

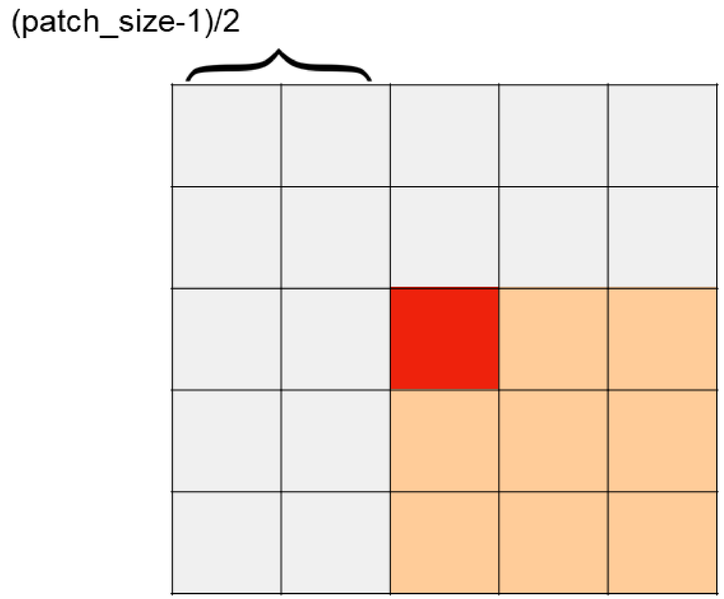
How many pixels to pad
on each side?



Patch-level processing: padding

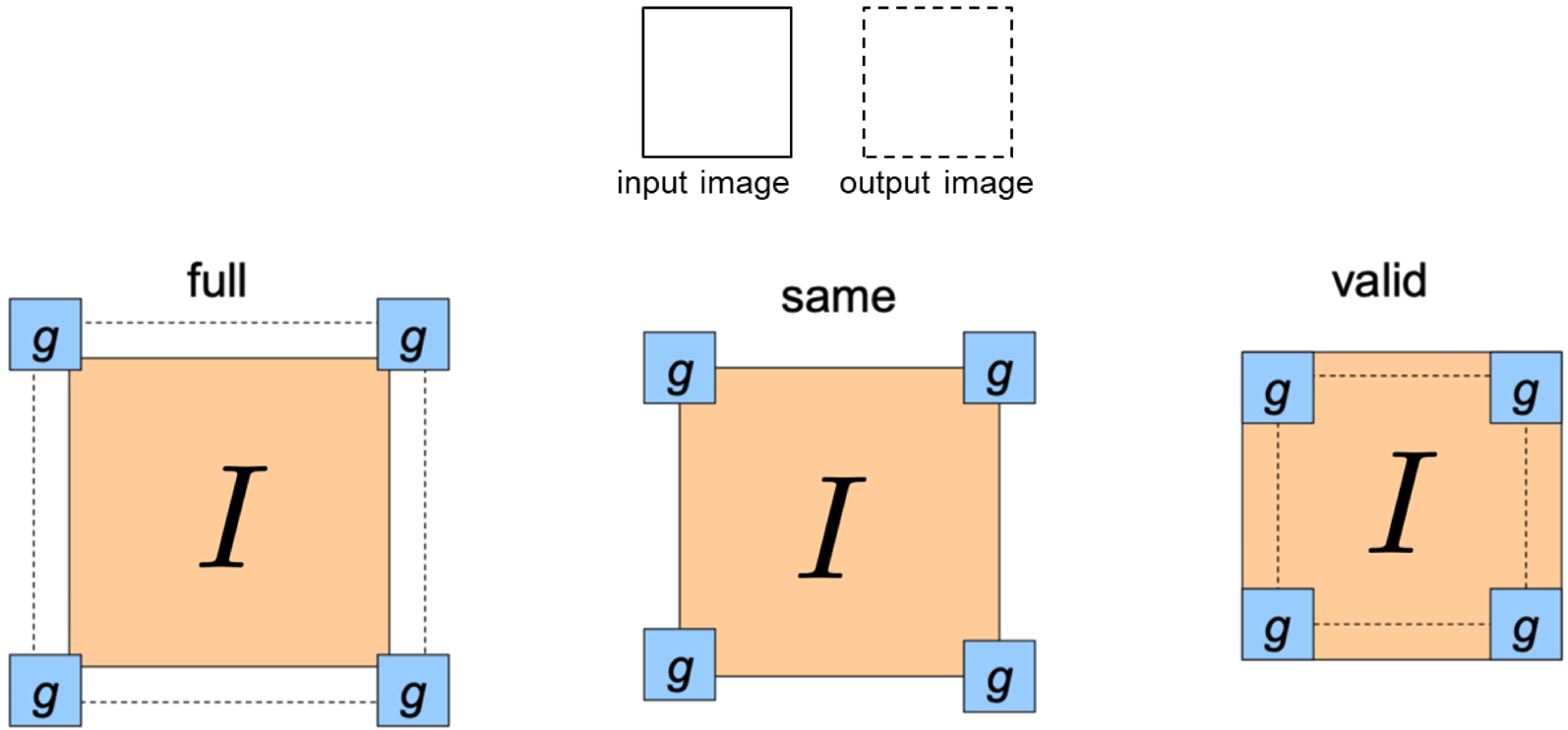
Zoom into the corner pixel





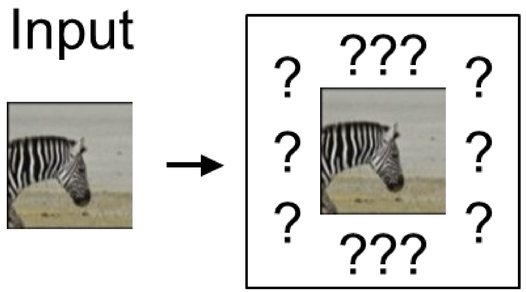
Patch-level processing: padding

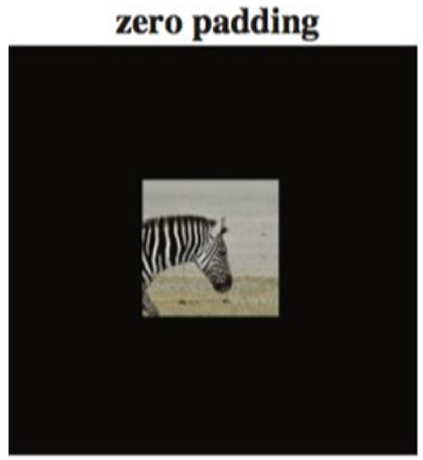
Design choice 1: How big to pad

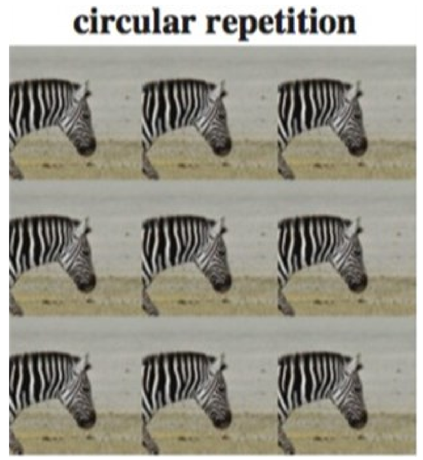


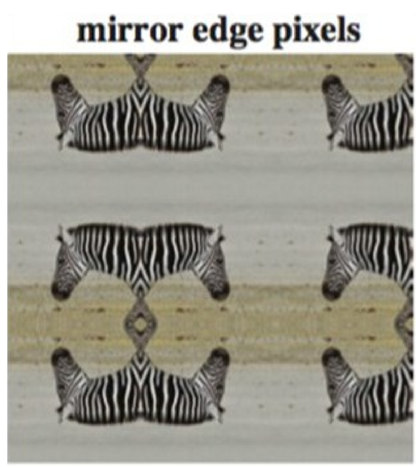
Patch-level processing: padding

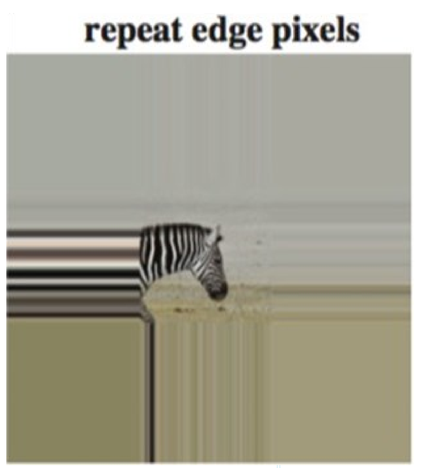
Design choice 2: What to pad











Email: min.shi@louisiana.edu