

MATLAB Function

Image Smoothing and Edge Detection

盧家鋒 Chia-Feng Lu, Ph.D.
Department of Biomedical Imaging
and Radiological Sciences, NYCU
alvin4016@nycu.edu.tw

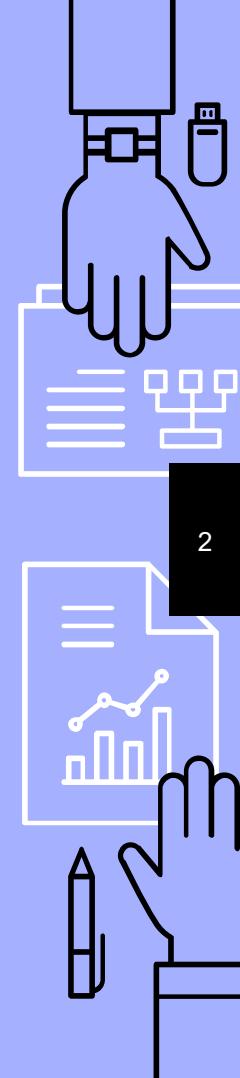
Contents

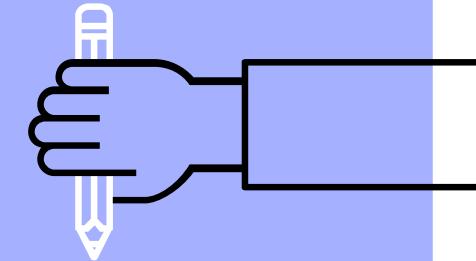
- ▶ Function declaration and usage
- ▶ Image smoothing and edge detection

2

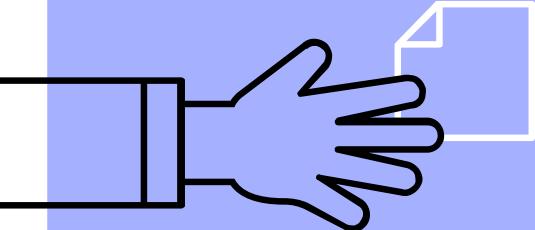
Please download the handout and materials from

http://cflu.lab.nycu.edu.tw/CFLu_course_matlabimage.html



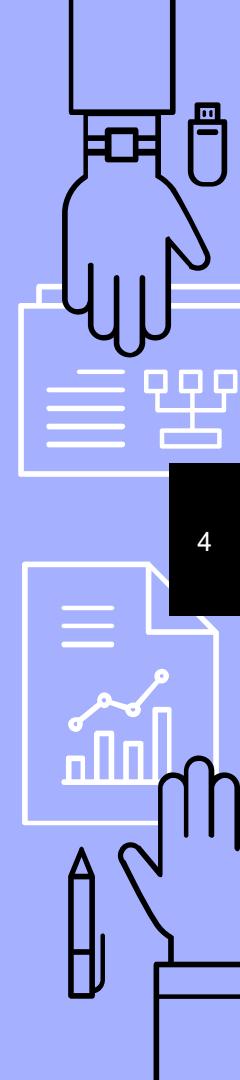


Function Declaration & usage



Useful Functions

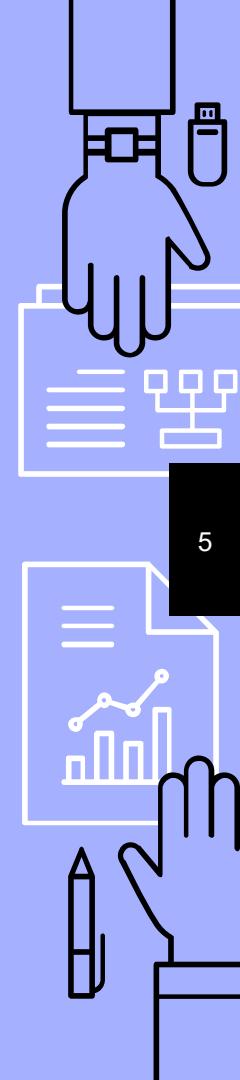
- ▶ A function is a group of statements that together perform a task.



abs log ceil
sin rand randn num2str
floor size zeros length
sort
exp
asin
min sum std
dicomread round
str2num **dicominfo**
ones **find** mean sqrt
mean median max
sinh

Basic Format

- ▶ Start with **function**
- ▶ Function name = file name
- ▶ <Optional> Input: right-hand side, within parentheses
- ▶ <Optional> Output: left-hand side, within brackets



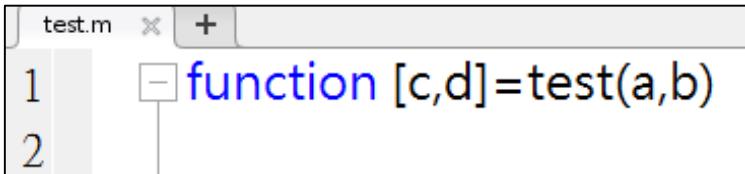
```
corr.m x +  
1 function [coef, pval] = corr(x, varargin)  
2 %CORR Linear or rank correlation.  
3 % RHO = CORR(X) returns a P-by-P matrix c  
4 % correlation coefficient between each pair  
5 % matrix X.
```

Common Errors

- ▶ Correct



```
test.m
1 function test
```

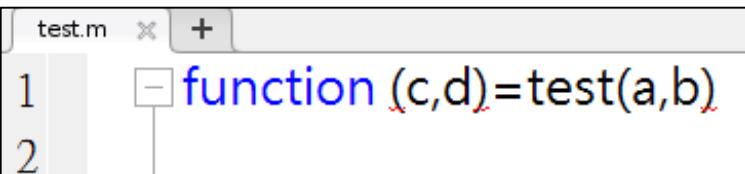


```
test.m
1 function [c,d]=test(a,b)
2
```

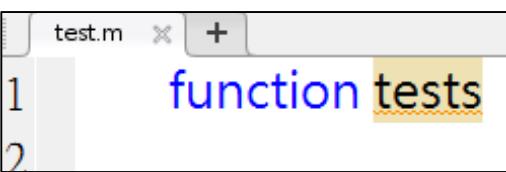
- ▶ Errors...



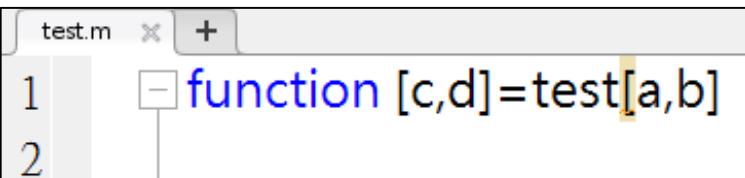
```
test.m
1 - functoin test
2
```



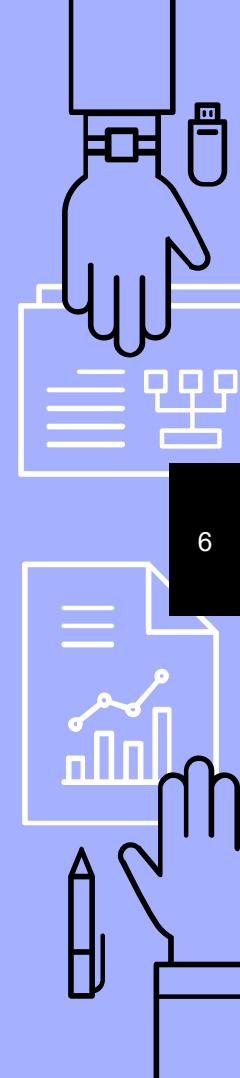
```
test.m
1 function (c,d)=test(a,b)
2
```



```
test.m
1 function tests
2
```

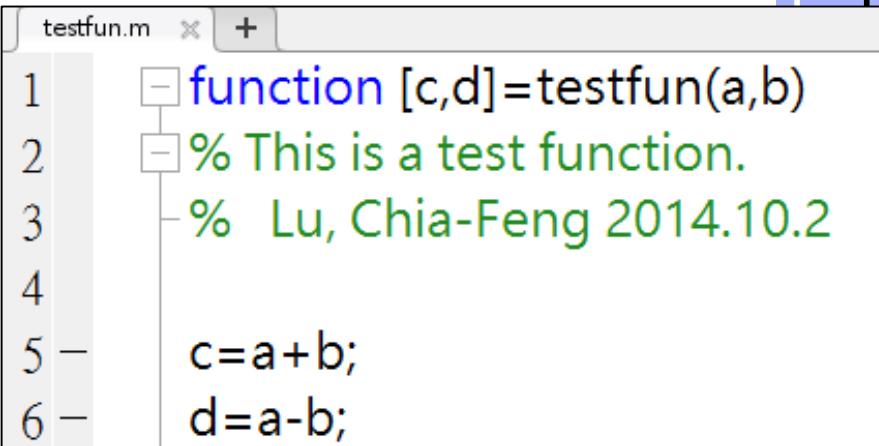


```
test.m
1 function [c,d]=test[a,b]
2
```



Basic Format

- ▶ **Function declaration**
 - Function name
 - Input and output variables
- ▶ **Function instruction**
 - Brief description
 - Introduction of variables
 - Examples
 - Author
- ▶ **Function content**
 - Remember to call all inputs
 - Remember to set all outputs

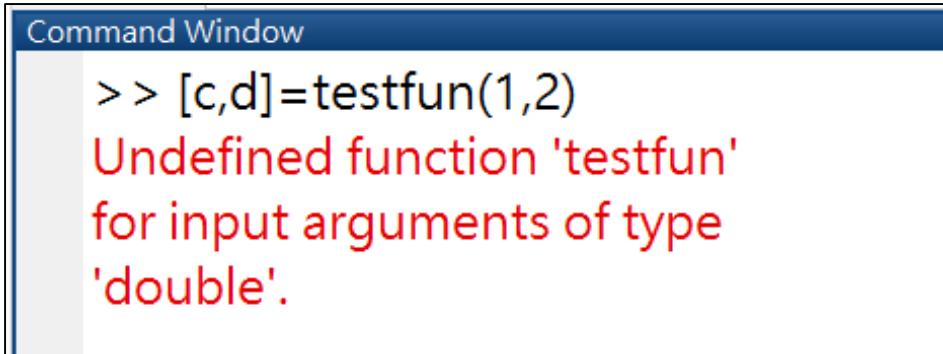


```
testfun.m
1 function [c,d]=testfun(a,b)
2 % This is a test function.
3 % Lu, Chia-Feng 2014.10.2
4
5 c=a+b;
6 d=a-b;
```

Try to create a test function!

Calling a Function

- ▶ Change "current directory"
- ▶ Set path to include the folder of function

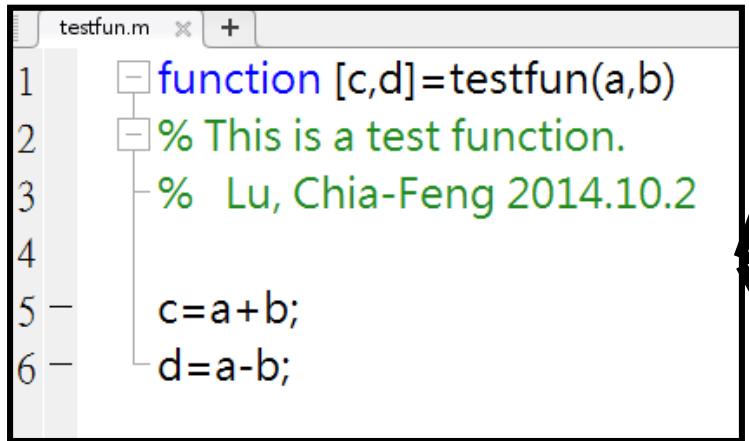


Command Window

```
>> [c,d]=testfun(1,2)
Undefined function 'testfun'
for input arguments of type
'double'.
```

Calling a Function

- ▶ Incorrect usages...



A screenshot of a MATLAB code editor window titled "testfun.m". The code contains the following:

```
function [c,d]=testfun(a,b)
% This is a test function.
% Lu, Chia-Feng 2014.10.2
c=a+b;
d=a-b;
```

```
>> [c,d,e]=testfun(1,2)
Error using testfun
Too many output arguments.
```

Forget to set input variables first!

```
>> [c,d]=testfun(a,b)
```

Undefined function or variable
'a'.

Forget to set path first!

```
>> [c,d]=testfun(1,2)
```

Undefined function 'testfun'
for input arguments of type
'double'.

```
>> [c,d]=testfun(1)
```

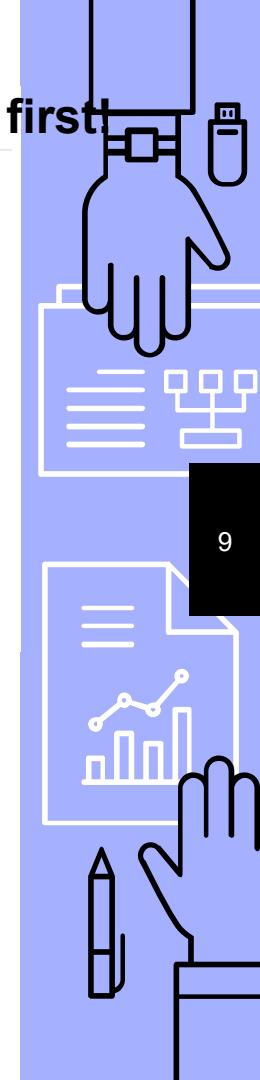
Error using testfun (line 5)

Not enough input arguments.

```
>> [c,d]=testfun(1,2,3)
```

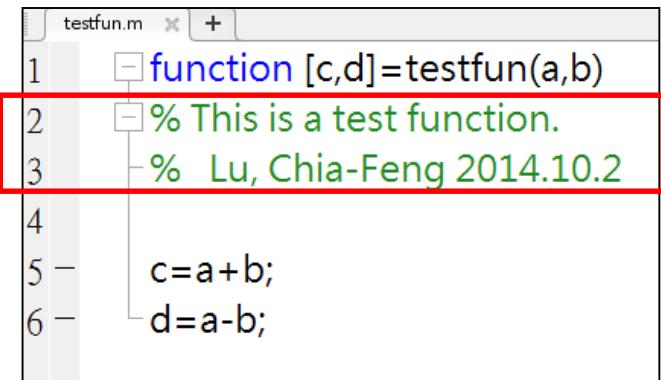
Error using testfun

Too many input arguments.



Instruction of Functions

- ▶ **help**
 - Display help text in Command Window.



```
testfun.m x +
1 function [c,d]=testfun(a,b)
2 % This is a test function.
3 % Lu, Chia-Feng 2014.10.2
4
5 c=a+b;
6 d=a-b;
```

consecutive notation lines

- ▶ **open**
 - Open files by extension
 - .m file open program file in MATLAB Editor.

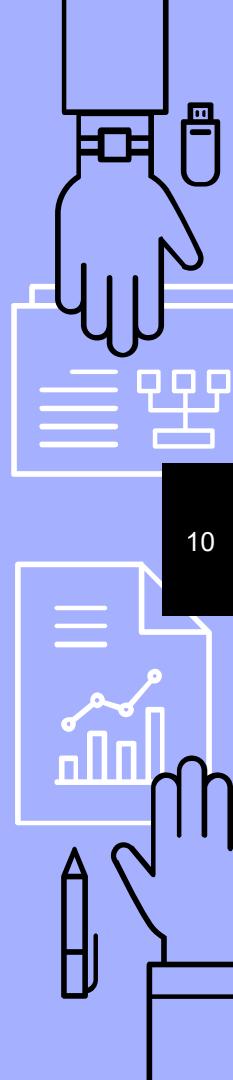
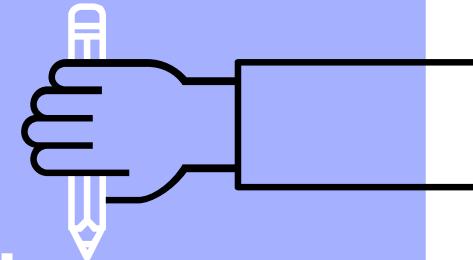
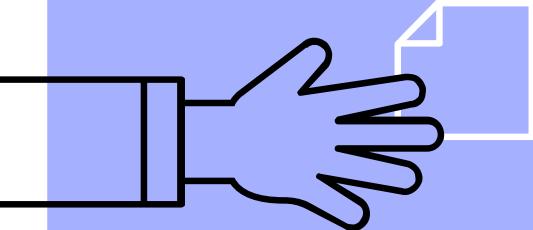
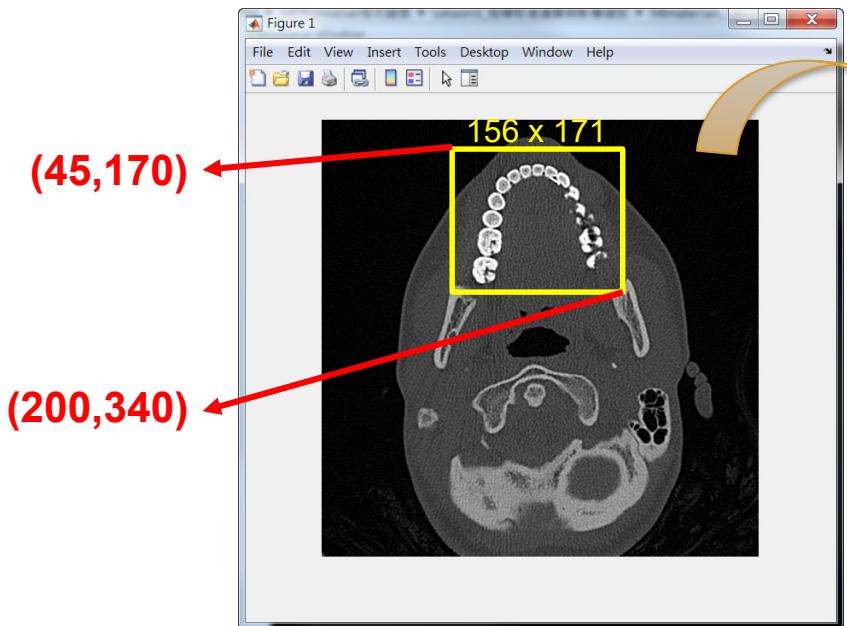


Image smoothing and edge detection

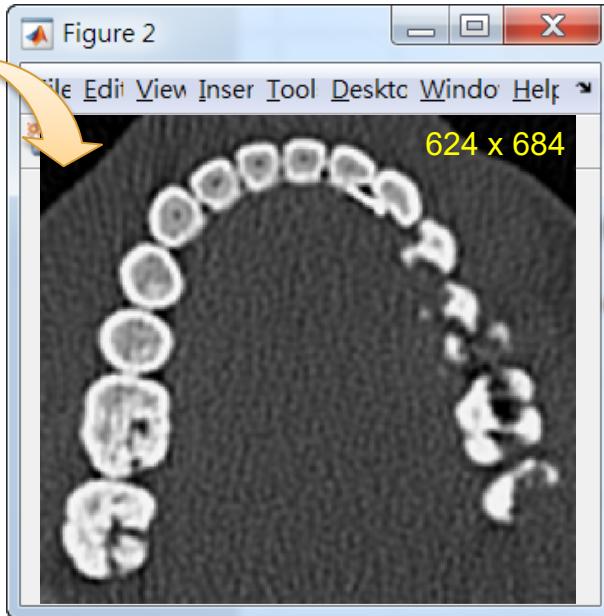


Step 1 – Import, Region, and Interpolation

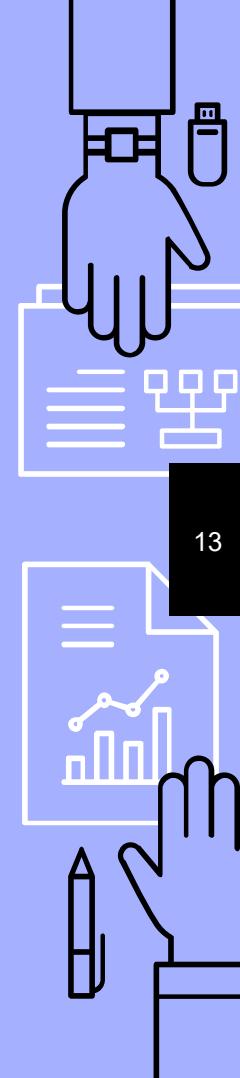
```
img=dicomread('IM-0001-0081.dcm');
```



Interpolated image (by a factor of 4)



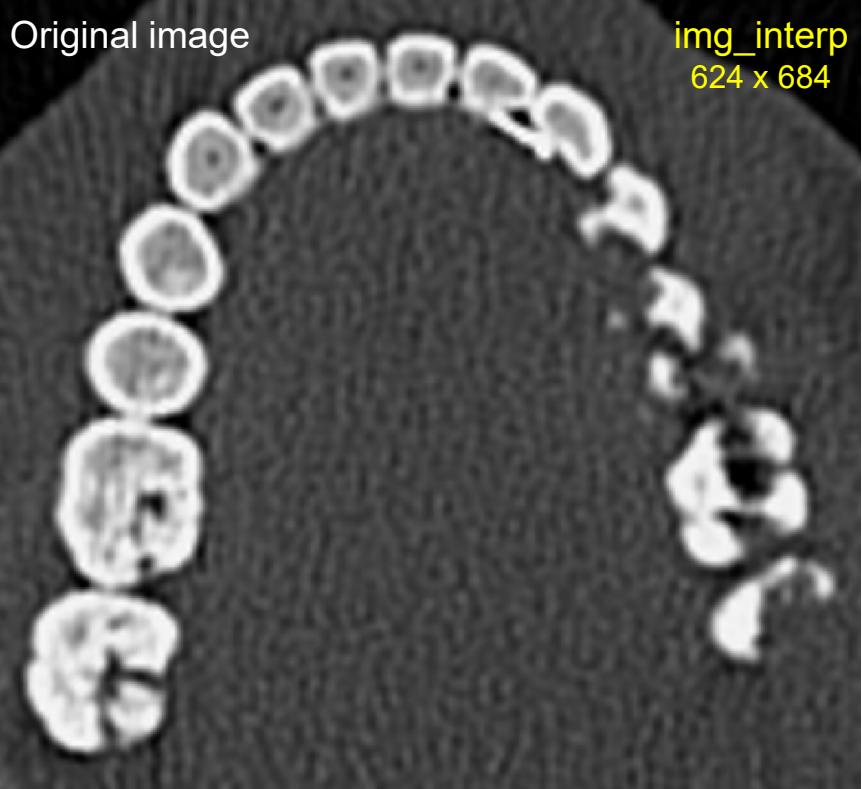
Please save as a Script File (*.m)~



Step 1 – Import, Region, and Interpolation

```
ImageEx01.m × +  
1 %% Perform Image Interpolation on local image region us  
2 - clear, close all  
3  
4 % get the image data of a DICOM file  
5 img=dicomread('IM-0001-0081.dcm');  
6  
7 % extract a local region, perform interpolation and display  
8 img_local=img(45:200,170:340);  
9 img_interp=imresize(img_local,4);  
10 figure, imshow(img_interp,[],'border','tight')
```

Step 2 – Image Smoothing



Step 2 – Image Smoothing

- 2D convolution

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

Original image

*

=

0	1	0
1	1	1
0	1	0

Kernel

1	2	4	3	2	0	0
0	2	3	5	3	2	0
0	0	2	4	5	2	1
0	0	2	4	3	2	0
0	2	3	3	2	0	0
2	3	3	2	0	0	0
2	3	2	0	0	0	0

Smoothed Image



Step 2 – Image Smoothing

▶ 2D convolution

x0	x1	x0					
x1	0 <i>x1</i>	1 <i>x1</i>	1	1 <i>x1</i>	0 <i>x1</i>	0 <i>x1</i>	0
x0	0 <i>x1</i>	0 <i>x0</i>	1	1 <i>x0</i>	1 <i>x1</i>	0 <i>x0</i>	0
0	0	0	1	1	1	1	0
0	0	0 <i>x0</i>	1 <i>x1</i>	1 <i>x0</i>	0	0	
0	0	1 <i>x1</i>	1 <i>x1</i>	0 <i>x1</i>	0	0	
0	1	1 <i>x0</i>	0 <i>x1</i>	0 <i>x0</i>	0	0	
1	1	0	0	0	0	0	0

Original image

$$\begin{matrix} * & = \end{matrix}$$

0	1	0
1	1	1
0	1	0

Kernel

1			2		
			3		

Smoothed Image



Step 2 – Image Smoothing

conv2 Two dimensional convolution.

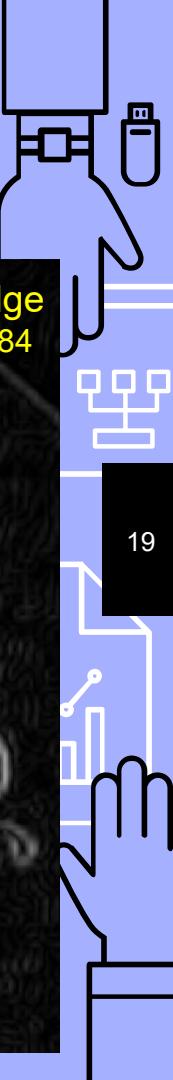
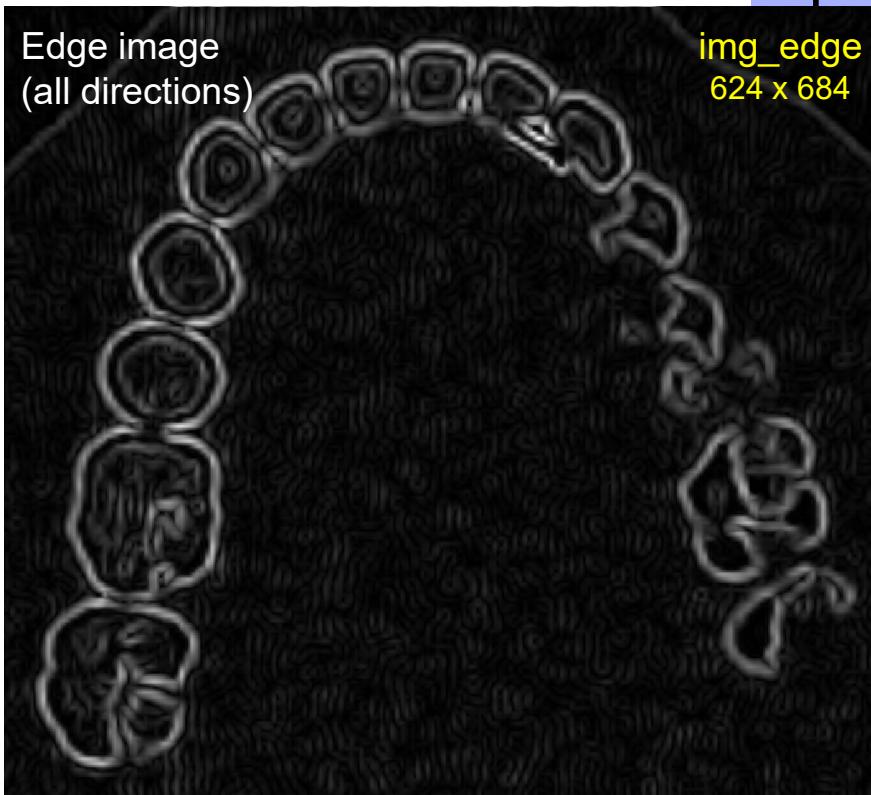
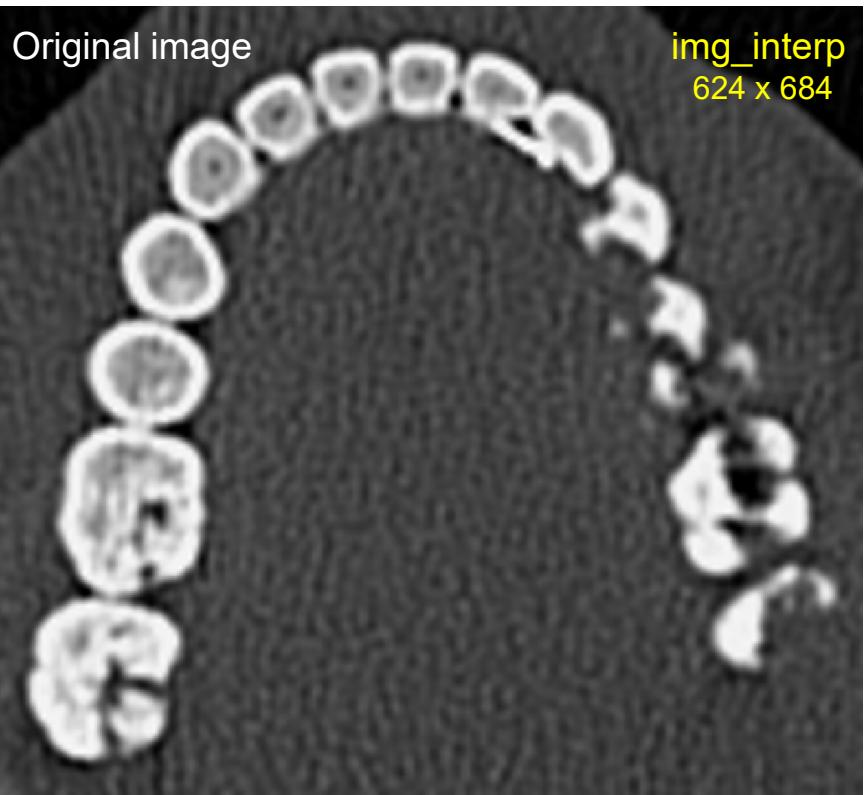
- ▷ `img_new=conv2(img,kernel,'same');`

```
12      %% perform image smoothing
13      kernal_sm=ones(15,15);
14      img_sm=conv2(double(img_interp),kernal_sm,'same');
15
16      figure, imshow(img_sm,[],'border','tight')
17
```

18



Step 3 – Edge Detection



Step 3 – Edge Detection

- 2D convolution

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

*

1	0	-1
1	0	-1
1	0	-1

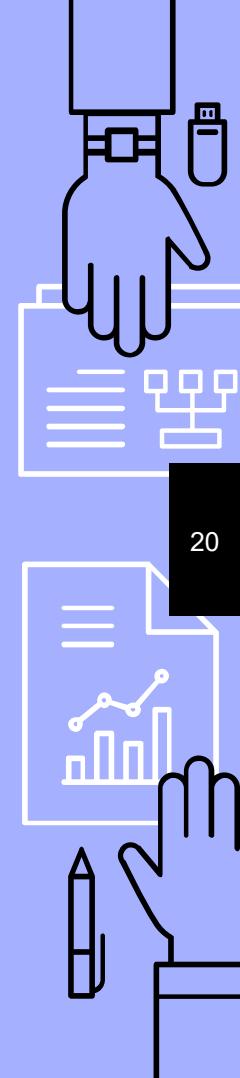
=

-1	-2	-1	1	2	1	0
-1	-2	-2	0	2	2	1
0	-1	-3	-2	2	3	1
0	-1	-3	-1	2	2	1
-1	-2	-1	1	2	1	0
-2	-1	1	2	1	0	0
-2	0	2	1	0	0	0

Original image

Kernel

Edge Image



Step 3 – Edge Detection

▶ 2D convolution

x1	x0	x(-1)		x1	x0	x(-1)				
x1	0	x0	1	1	x1	0	x0	x(-1)	0	
x1	0	x0	x(-1)	1	1	x1	1	x0	x(-1)	0
0	0	0	1	1	1	1	0	0	0	
0	0	0	x1	1	x0	x(-1)	0	0	0	
0	0	1	x1	1	x0	x(-1)	0	0	0	
0	1	1	x1	0	x0	x(-1)	0	0	0	
1	1	0	0	0	0	0	0	0	0	

Original image

*

1	0	-1
1	0	-1
1	0	-1

Kernel

=

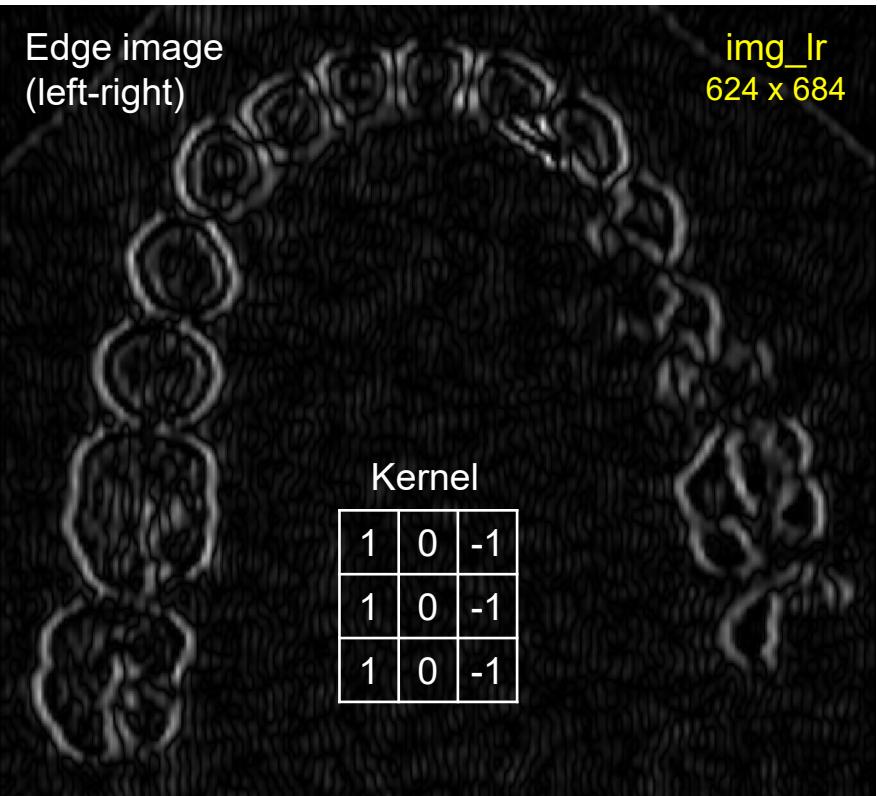
-1			2		
			1		

Edge Image

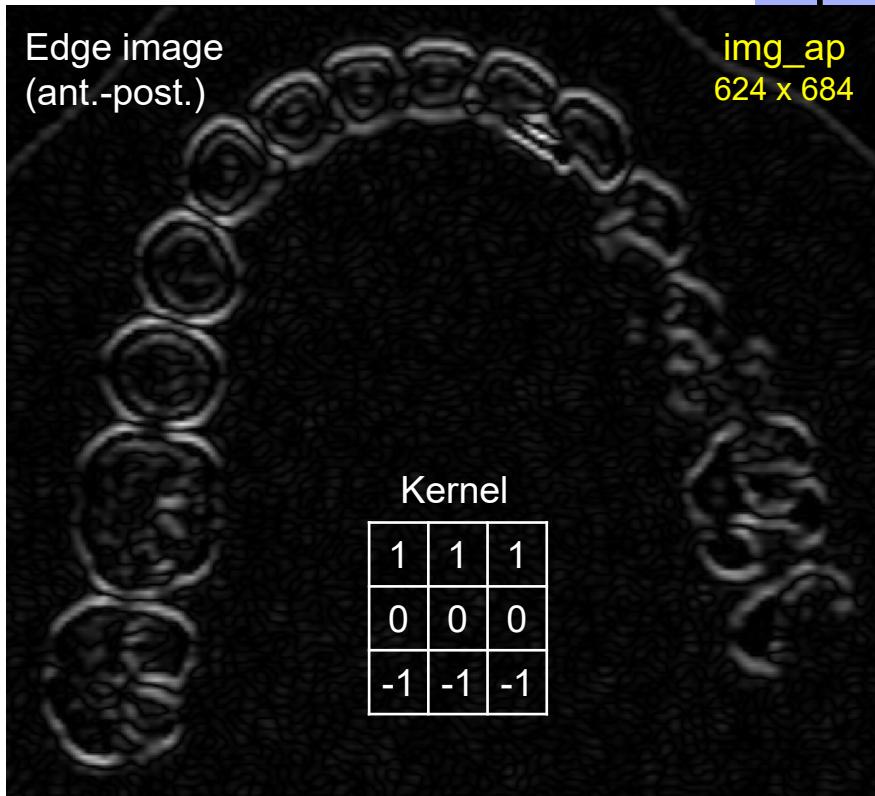


Step 3 – Edge Detection

Edge image
(left-right)

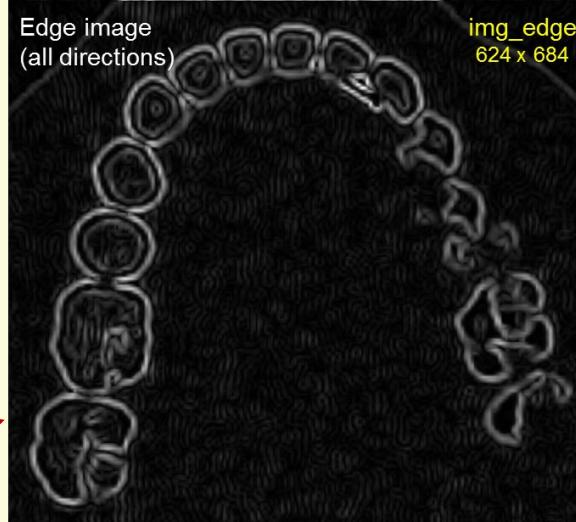


Edge image
(ant.-post.)



Step 3 – Edge Detection

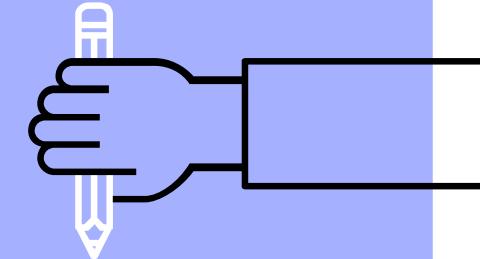
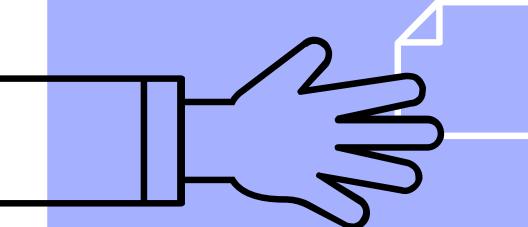
```
18 %% perform edge detection
19 kernal_lr=[1 0 -1;1 0 -1;1 0 -1];
20 img_lr=conv2(double(img_interp),kernal_lr,'same');
21 figure, imshow(abs(img_lr),[],'border','tight')
22
23 kernal_ap=[1 1 1;0 0 0;-1 -1 -1];
24 img_ap=conv2(double(img_interp),kernal_ap,'same');
25 figure, imshow(abs(img_ap),[],'border','tight')
26
27 img_edge=max(abs(img_lr),abs(img_ap));
28 figure, imshow(img_edge,[],'border','tight')
```



Homework

- ▶ Create a function that can...
 - extract local image **based on rowrange and colrange**,
 - perform interpolation (by a factor of 4),
 - image smoothing (15 pixels),
 - edge detection (3 pixels, L/R & A/P)
- ▶ **[img_sm,img_edge]=imgprocess(img,rowrange,colrange);**
 - Modify from **ImageEx04.m**
 - Do not load image within the function (**dicomread**)!
 - Do not display any image within the function (**imshow**)!





THE END

alvin4016@nycu.edu.tw