جَـامعة
المَـنارة
MANARA UNIVERSITY

معالجة الصور الرقمية - ميكاترونيكس

د. عيسى الغنام

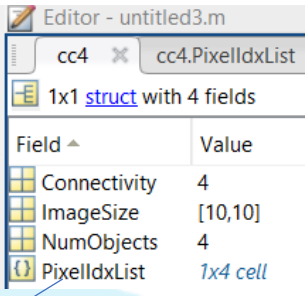
Example one: 4 and 8 connectivity

```
إنشاء صورة ثنائية (BW = Binary Image)
BW = false(10,10);
إنشاء مصفوفة 10x 10 من القيم المنطقية كلها false

إضافة بعض البكسلات لإنشاء كائنات مختلفة
BW(2,2) = true; BW(2,3) = true; % مكون أفقي
BW(3,2) = true; BW(4,2) = true; % مكون عمودي
BW(5,5) = true;
BW(6,6) = true; BW(7,7) = true;
مكونات متصلة قطريًا فقط (مهم للتمييز بين 4 و 8 اتصال)
figure;
imshow(BW);
title('Original Binary Image');
%%%%%%%%%%%%%%%%%%%%%%%%%
استخدام الاتصال الرباعي(4-connectivity)
cc4 = bwconncomp(BW, 4);
labeled4 = labelmatrix(cc4); % تحويل النتائج إلى صورة مُعنونة لتسهيل العرض
figure;
imshow(labeled4, []); % [] لضبط نطاق الألوان تلقائيًا
colormap jet; % استخدام خريطة ألوان لتوضيح المكونات المختلفة
بدونها تظهر الضورة رمادية بدرجات مختلفة حسب عدد المكونات
```

Original Binary Image

4-Connectivity (Components labeled)



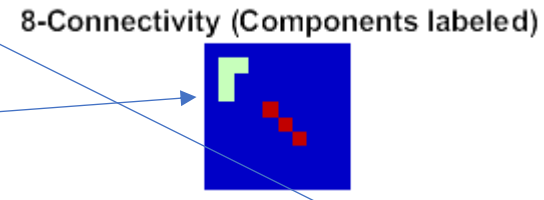
cc4.PixelIdxList

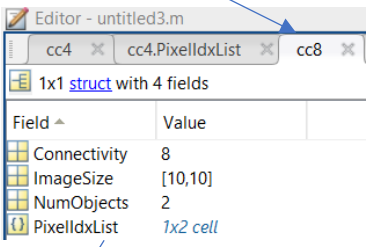
	1	2	3	4
1	[12;13;14;22]	45	56	67

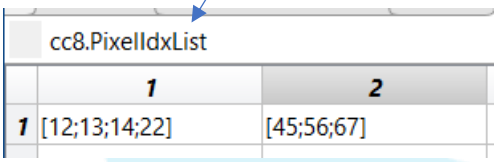
```
title('4-Connectivity (Components labeled)');
fprintf('Number of components (4-conn): %d\n', cc4.NumObjects);
%%%%%%%%%%%%%%%%%
                                    استخدام الاتصال الثماني(8-connectivity)
cc8 = bwconncomp(BW, 8);
labeled8 = labelmatrix(cc8);


figure;
imshow(labeled8, []);
colormap jet;
title('8-Connectivity (Components labeled)');
fprintf('Number of components (8-conn): %d\n', cc8.NumObjects);
```

Number of components (4-conn): 4







Number of components (8-conn): 2

Example Two: arithmatical operations

```
% Clear workspace, close figures, clear command window
clear; close all; clc;
original_img_uint8 = imread('cameraman.tif');
% a built-in MATLAB example image.  grayscale and uint8.
% If image is RGB, you would convert it to grayscale first.

figure;  imshow(original_img_uint8);  title('Original Grayscale Image (uint8)');

% --- Define the number to add ---
number_to_add = 50;

% --- Perform the addition (Crucial: Data Type Considerations) ---
% It's best practice to convert the image to a 'double' data type
% BEFORE performing arithmetic operations, such as addition.
% This prevents pixel values from "clipping" or "wrapping around" if
% they exceed the maximum value (e.g., 255 for uint8) during addition.
% For example, if a uint8 pixel is 230 and you add 50, it would become 255 (clipped).
% If it were double, 230 + 50 = 280.

img_double = double(original_img_uint8); % Convert to double. Values will still be 0-255 initially.
% Add the number to the double image
img_added_double = img_double + number_to_add;

% --- 4. Display the modified image ---
```

% Option A: Displaying the double image with automatic scaling

% Using imshow(image, []) tells MATLAB to scale the data in 'image'

% to the full display range (0 to 255 for grayscale) automatically.

% This is often preferred for visualizing processed double images,

% especially if values went above 255 or below 0.

figure;

imshow(img_added_double, []);

% without using [], imshow will display white image

title(['Image with ', num2str(number_to_add), ' Added (Double, auto-scaled)']);

% --- Explanation of the effect ---

% Adding a positive number makes the image brighter.

% Adding a negative number (e.g., -50) would make it darker.

% --- Option B: Convert back to uint8 if needed for saving or further uint8 operations ---

% When converting back to uint8, values outside the 0-255 range will be clipped.

% For example, 280 will become 255, and -20 will become 0.

img_added_uint8 = uint8(img_added_double);

% display this clipped version if you want to see the effect

figure;imshow(original_img_uint8);title('Original Image');

figure;imshow(img_added_uint8);title(['Image with ', num2str(number_to_add), ' Added (Clipped to uint8)']);

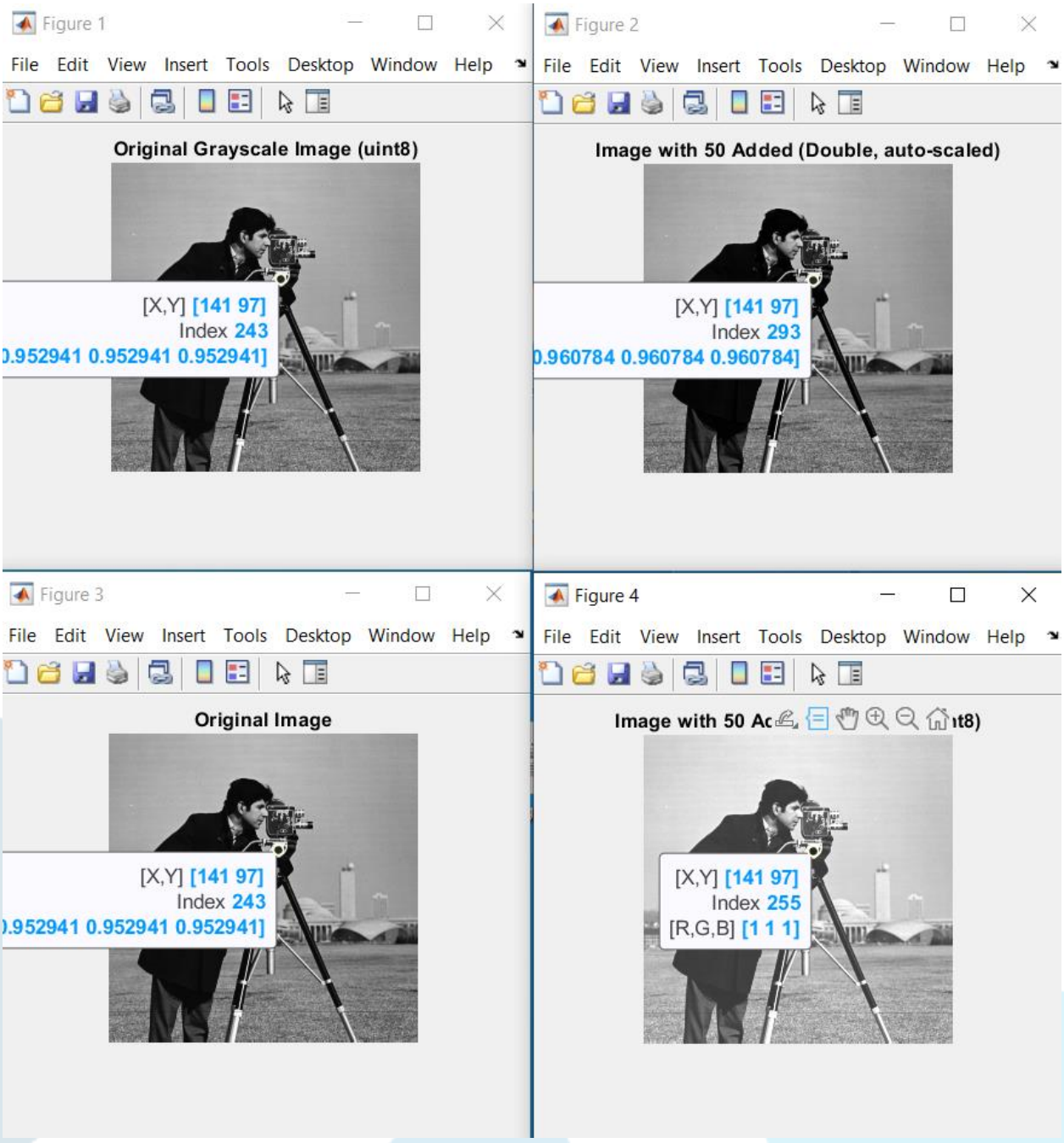


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Grayscale Image (uint8)
[X,Y] [141 97]
Index 243
0.952941 0.952941 0.952941]
Figure 2
File Edit View Insert Tools Desktop Window Help
Image with 50 Added (Double, auto-scaled)
[X,Y] [141 97]
Index 293
0.960784 0.960784 0.960784]
Figure 3
File Edit View Insert Tools Desktop Window Help
Original Image
[X,Y] [141 97]
Index 243
0.952941 0.952941 0.952941]
Figure 4
File Edit View Insert Tools Desktop Window Help
[X,Y] [141 97]
Index 255
[R,G,B] [1 1 1]

Arithmetic operations with Normalization

```
% Clear workspace, close figures, clear command window
clear;close all;clc;

% --- 1. Load a grayscale image ---
original_img_uint8 = imread('cameraman.tif'); % A grayscale image

figure('Name', 'Image Processing with Normalization');
subplot(2, 3, 1);
imshow(original_img_uint8);
title('1. Original Image (uint8)');
```


```
% --- 2. Convert to double and add a number ---
% IMPORTANT: Convert to double before arithmetic to avoid clipping
img_double = double(original_img_uint8);
number_to_add = 70; % Example: add a positive number to brighten
img_added = img_double + number_to_add;

subplot(2, 3, 2);
```

imshow(img_added, []); % Display with auto-scaling for current range

title(['2. After Adding ', num2str(number_to_add), ' (double, auto-scaled)']);

% Notice how this looks brighter. Its pixel values are now mostly > 255

% The min/max values have shifted.

% --- 3. Normalization Methods ---

% Get the new min and max values after addition

min_val_after_add = min(img_added(:));

max_val_after_add = max(img_added(:));

fprintf('Min pixel value after addition: %.2f\n', min_val_after_add);

fprintf('Max pixel value after addition: %.2f\n', max_val_after_add);

Min pixel value after addition: 77.00

Max pixel value after addition: 323.00

% --- Method A: Using mat2gray (Normalizes to [0, 1]) ---

```
% mat2gray scales the image to the range [0, 1], suitable for double images.
img_normalized_mat2gray = mat2gray(img_added);
subplot(2, 3, 3);
imshow(img_normalized_mat2gray); % imshow displays 0-1 double images correctly
title('3. Normalized to [0,1] (using mat2gray)');
```


```
fprintf('Min pixel value after mat2gray: %.2f\n', min(img_normalized_mat2gray(:)));
fprintf('Max pixel value after mat2gray: %.2f\n', max(img_normalized_mat2gray(:)));
```

```
Min pixel value after mat2gray: 0.00
Max pixel value after mat2gray: 1.00
```

% --- Method B: Manual Linear Normalization to [0, 255] ---

```
% This gives you full control and helps you understand the process.
% First, scale to 0-1:
img_normalized_0_1_manual = (img_added - min_val_after_add) / (max_val_after_add - min_val_after_add);
```

% Then, scale to 0-255 and convert to uint8:

img_normalized_0_255_manual = uint8(img_normalized_0_1_manual * 255);

subplot(2, 3, 4);

imshow(img_normalized_0_255_manual);

title('4. Normalized to [0,255] (Manual)');

fprintf('Min pixel value after manual 0-255: %d\n', min(img_normalized_0_255_manual(:)));

fprintf('Max pixel value after manual 0-255: %d\n', max(img_normalized_0_255_manual(:)));

```
Min pixel value after manual 0-255: 0
Max pixel value after manual 0-255: 255
```

% --- Method C: Using rescale (Modern and Flexible) ---

% rescale function allows specifying target range directly.

img_normalized_rescale_0_1 = rescale(img_added, 0, 1);

img_normalized_rescale_0_255 = rescale(img_added, 0, 255); % Still double

subplot(2, 3, 5);

imshow(img_normalized_rescale_0_1);

title('5. Normalized to [0,1] (using rescale)');


```
fprintf('Min pixel value after rescale 0-1: %.2f\n', min(img_normalized_rescale_0_1(:)));
fprintf('Max pixel value after rescale 0-1: %.2f\n', max(img_normalized_rescale_0_1(:)));
```

```
Min pixel value after rescale 0-1: 0.00
Max pixel value after rescale 0-1: 1.00
```

```
% Convert the 0-255 double result from rescale to uint8 for saving/display as integer type
img_normalized_rescale_uint8 = uint8(img_normalized_rescale_0_255);
subplot(2, 3, 6);
imshow(img_normalized_rescale_uint8);
title('6. Normalized to [0,255] (rescale, then uint8)');
```

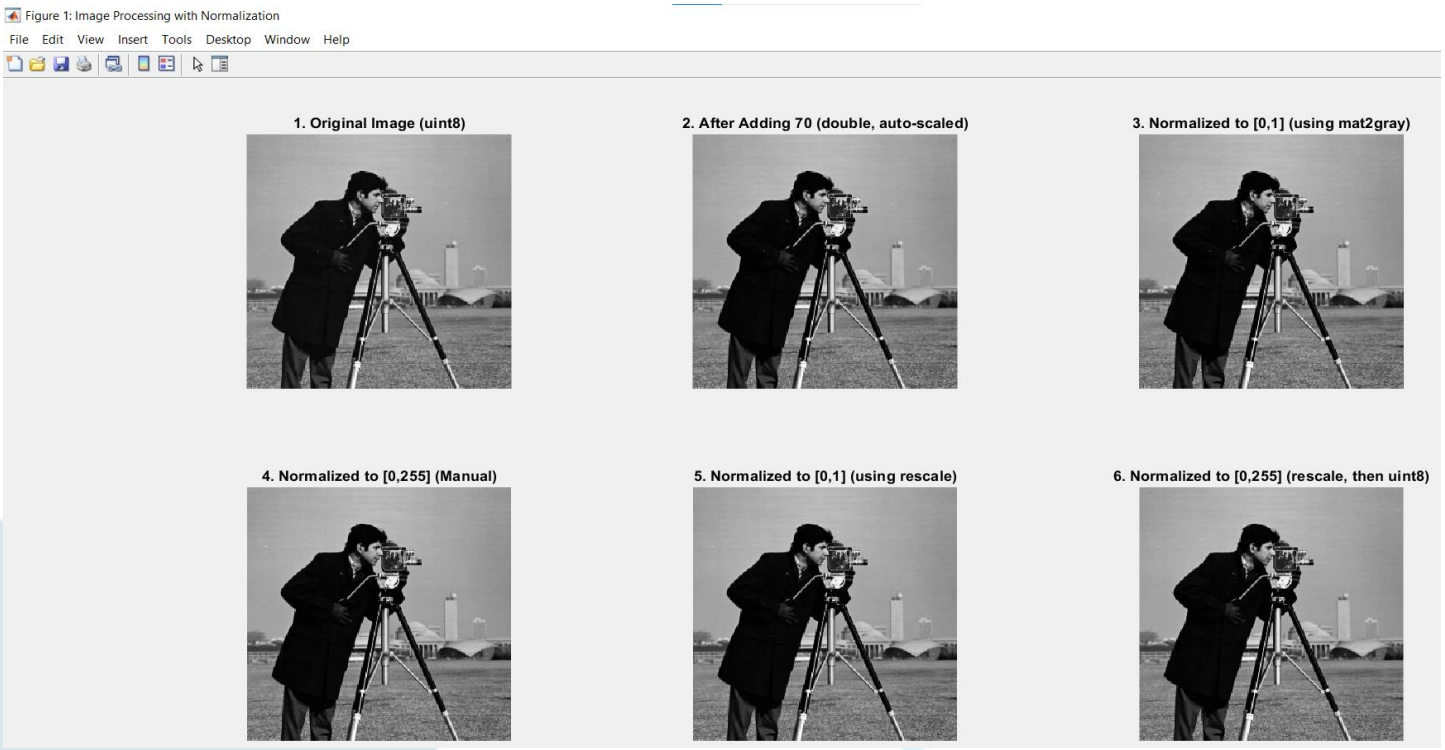
fprintf('Min pixel value after rescale 0-255 then uint8: %d\n', min(img_normalized_rescale_uint8(:)));

fprintf('Max pixel value after rescale 0-255 then uint8: %d\n', max(img_normalized_rescale_uint8(:)));

Min pixel value after rescale 0-255 then uint8: 0

Max pixel value after rescale 0-255 then uint8: 255

Cod output:



Min pixel value after addition: 77.00

Max pixel value after addition: 323.00

Min pixel value after mat2gray: 0.00

Max pixel value after mat2gray: 1.00

Min pixel value after manual 0-255: 0

Max pixel value after manual 0-255: 255

Min pixel value after rescale 0-1: 0.00

Max pixel value after rescale 0-1: 1.00

Min pixel value after rescale 0-255 then uint8: 0

Max pixel value after rescale 0-255 then uint8: 255

% --- Comparison with simple clipping (without normalization) ---

```
img_clipped_uint8 = uint8(img_added);
% Values > 255 are clipped to 255.
% This can lead to loss of detail in brighter areas.
figure('Name', 'Comparison: Normalization vs. Simple Clipping');
subplot(1, 2, 1);
imshow(img_normalized_0_255_manual);
title('Normalized (retains detail)');
subplot(1, 2, 2);
imshow(img_clipped_uint8);
title('Simply Clipped (some detail lost)');
```