Digital image processing using MATLAB

Image reading and display

- ".mat" files

```
clear all;
clf; % clear current figure window
load imagereading.mat % read the input image
figure(1);imshow(imagereading); % display the image
```

- picture files

```
clf;
img_read = imread('image1.jpg'); % read the input image
```

```
figure(2);imshow(img_read);
```

some helpful functions

- title()
- axis()

```
figure(3);imshow(img_read);
title('image'); % add a title to the image
axis([0 400 0 400]); % specify the range of the image display
```

- subplot()

```
% Using subplot to split figure .
% Hence, we can show more than one image in the same figure.
figure(4);
subplot(1,2,1);
imshow(imagereading);title('fig1');
subplot(1,2,2);
imshow(img_read);title('fig2');
```

fig1

fig2

Data type

- uint8 & double
- double()
- im2double()
- uint8()
- im2uint8()

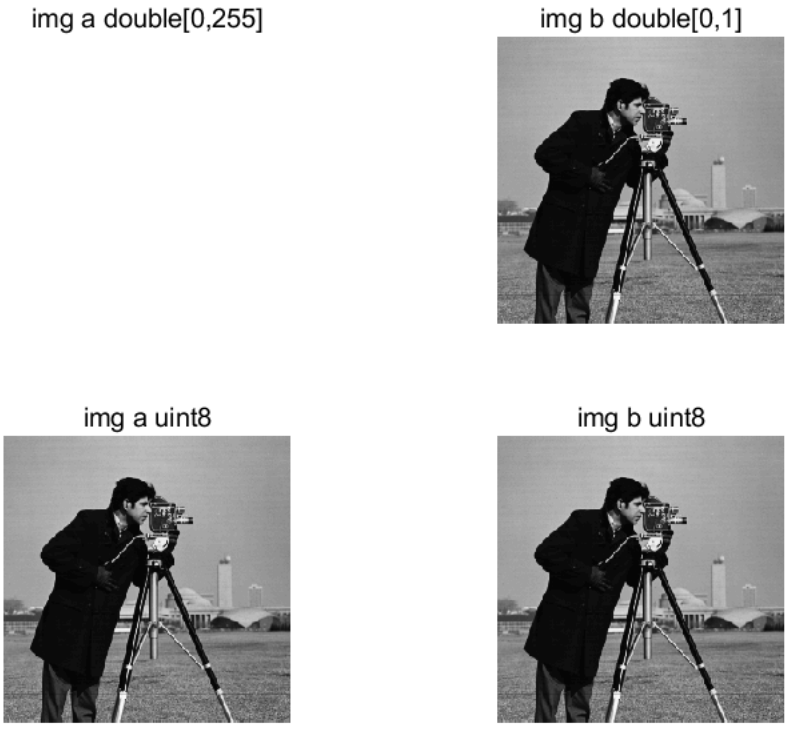
```
clf;
camaraman = imread('cameraman.tif'); % uint8 data
% both functions below turn 'uint8' type data into 'double' type data
img_a = double(camaraman); % double() keeps at a range of [0,255]
img_a
```

```
img_a = 256×256
   156   159   158   155   158   156   159   158   157   158   158   159   160 ···
   160   154   157   158   157   159   158   158   158   160   155   156   159
   156   159   158   155   158   156   159   158   157   158   158   159   160
   160   154   157   158   157   159   158   158   158   160   155   156   159
   156   153   155   159   159   155   156   155   155   157   155   154   154
   155   155   155   157   156   159   152   158   156   158   152   153   159
   156   153   157   156   153   155   154   155   157   156   155   156   155
   159   159   156   158   156   159   157   161   162   157   157   159   161
   158   155   158   154   156   160   162   155   159   161   156   161   160
   155   154   157   158   160   160   159   160   158   161   160   160   158
     ⋮
     ⋮
```

```
img_b = im2double(camaraman); % im2double() turn the range to [0,1]
img_b
```

```
img_b = 256×256
    0.6118    0.6235    0.6196    0.6078    0.6196    0.6118    0.6235    0.6196 ···
    0.6275    0.6039    0.6157    0.6196    0.6157    0.6235    0.6196    0.6196
    0.6118    0.6235    0.6196    0.6078    0.6196    0.6118    0.6235    0.6196
    0.6275    0.6039    0.6157    0.6196    0.6157    0.6235    0.6196    0.6196
    0.6118    0.6000    0.6078    0.6235    0.6235    0.6078    0.6118    0.6078
    0.6078    0.6078    0.6078    0.6157    0.6118    0.6235    0.5961    0.6196
    0.6118    0.6000    0.6157    0.6118    0.6000    0.6078    0.6039    0.6078
    0.6235    0.6235    0.6118    0.6196    0.6118    0.6235    0.6157    0.6314
    0.6196    0.6078    0.6196    0.6039    0.6118    0.6275    0.6353    0.6078
    0.6078    0.6039    0.6157    0.6196    0.6275    0.6275    0.6235    0.6275
      ⋮
      ⋮
```

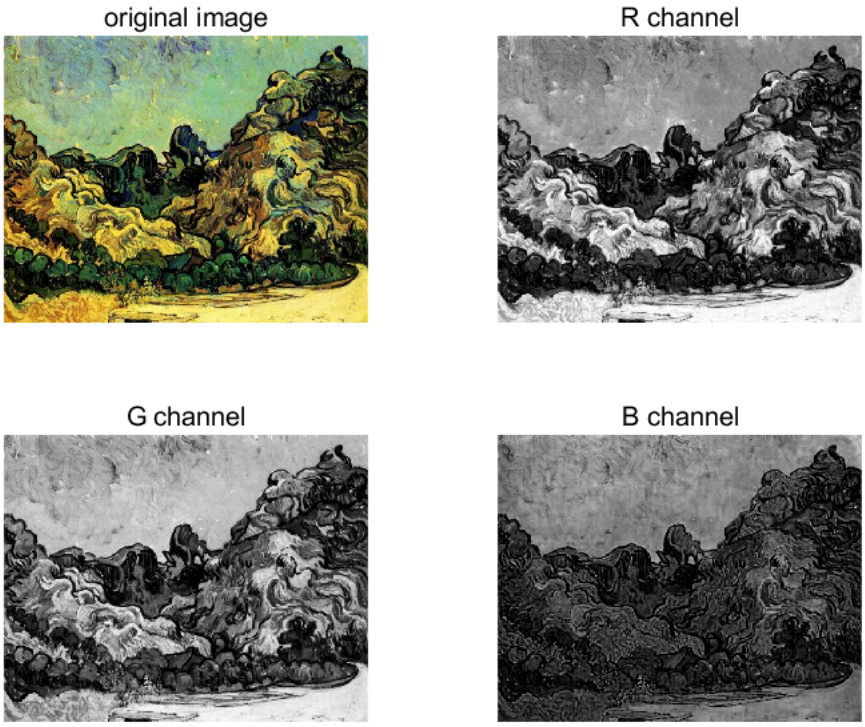
```
% When we use imshow to display the image, we must pay attention to the data
% type. If the image is 'double' type, then only when it's normalized, it
% can be shown by function 'imshow()', which means the 'double' type data
% must at a range of [0,1].
% If the data is at range [0,255], we must turn its type into 'uint8'.
% function'double()' ---invert---function'uint8()'
% function'im2double()'---invert---function'im2uint8()'
figure(5);
subplot(2,2,1);imshow(img_a);title('img a double[0,255]');
subplot(2,2,2);imshow(img_b);title('img b double[0,1]');
subplot(2,2,3);imshow(uint8(img_a));title('img a uint8');
subplot(2,2,4);imshow(im2uint8(img_b));title('img b uint8');
```



RGB image

- How to split three color channels?

```
clf;
original_img = imread('image2.jpg');% Read the input image, here the image is a 3-D matrix.
img_R = original_img(:,:,1);% Get the data of R channel. Note that img_R is a 2-D matrix.
img_G = original_img(:,:,2);% Get the data of G channel. Note that img_G is a 2-D matrix.
img_B = original_img(:,:,3);% Get the data of B channel. Note that img_B is a 2-D matrix.
figure(6);
subplot(2,2,1);imshow(original_img);title('original image');
subplot(2,2,2);imshow(img_R);title('R channel');
subplot(2,2,3);imshow(img_G);title('G channel');
subplot(2,2,4);imshow(img_B);title('B channel');
```

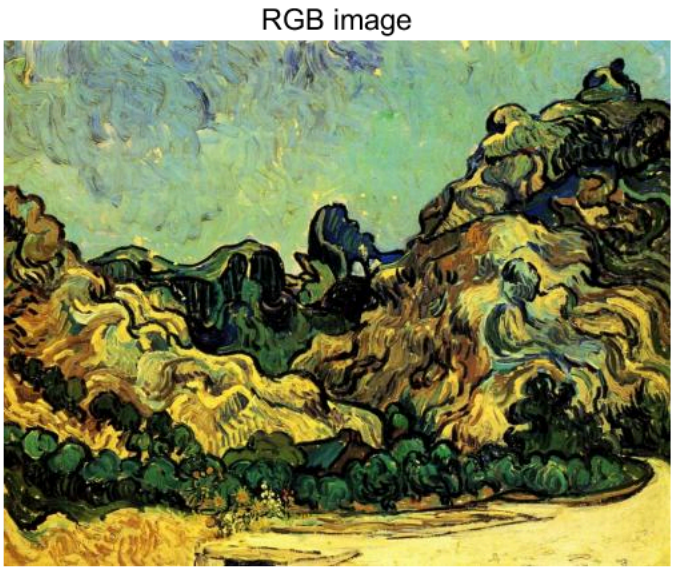


The R,G,B channel's images are all gray. For a single channel image, each pixel only records its intensity. There is no color imformation for a graylevel image. When we concatenate all the channels, each channel will be defined as a R/G/B channel,which means they have color imformation. That's why three gray images can become a color image.

- How to concatenate three color channels?
- cat()

```
clf;
result = cat(3,img_R,img_G,img_B);
```

```
figure(7);imshow(result);title('RGB image');
```



Some useful functions for this homework

- mean(): find the average value of input

```
A = [0,1,2;3,4,5;6,7,8] % define a 3*3 matrix
```

```
A = 3×3
     0     1     2
     3     4     5
     6     7     8
```

```
average_A_col = mean(A) % function 'mean()' gives out the average of each column
```

```
average_A_col = 1×3
     3     4     5
```

```
average_A_row = mean(A,2)% give dim=2 to calculate the average of each row
```

```
average_A_row = 3×1
     1
     4
     7
```

```
average_A = mean(mean(A)) % If we want the average value of matrix A, we can use 'mean()' two t
```

```
average_A = 4
```

```
average_a = mean(A,"all") % Another way to get the average value of matrix A.
```

```
average_a = 4
```

- sort():the arrangement of data in a prescribed sequence

```
B = [0,7,6;3,5,4;1,8,2] % define a 3*3 matrix
```

```
B = 3×3
     0     7     6
     3     5     4
     1     8     2
```

```
new_B_col = sort(B) % sort matrix B in each column (dim=1)(defalt dimension is 1)
```

```
new_B_col = 3×3
     0     5     2
     1     7     4
     3     8     6
```

```
new_B_row = sort(B,2) % sort matrix B in each row (dim=2)
```

```
new_B_row = 3×3
     0     6     7
     3     4     5
     1     2     8
```

```
new_B_all = sort(B(:)) % sort all the elements in matrix B
```

```
new_B_all = 9×1
     0
     1
     2
     3
     4
     5
     6
     7
     8
```

```
new_B_all' % the symbol ' is used to find the transpose of matrix new_B_all
```

```
ans = 1×9
     0     1     2     3     4     5     6     7     8
```

```
new_B_h2l = sort(B(:),'descend'); % 'descend' defines the sort direction (high to low)
new_B_h2l'
```

```
ans = 1×9
     8     7     6     5     4     3     2     1     0
```

- round():find the nearest integer

```
C = 1.33;D = 6.67;
round_C = round(C)
```

```
round_C = 1
```

```
round_D = round(D)
```

```
round_D = 7
```

```
E = [2.11 3.5; -3.5 0.78];
round_E = round(E)
```

```
round_E = 2×2
     2     4
    -4     1
```

- cumsum():cumulative sum

```
F = [1,4,7;2,5,8;3,6,9] % define a 3*3 matrix F
```

```
F = 3×3
     1     4     7
     2     5     8
     3     6     9
```

```
cumsum_F_col = cumsum(F) % do the cumulative sum of each column
```

```
cumsum_F_col = 3×3
     1     4     7
     3     9    15
     6    15    24
```

```
cumsum_F_row = cumsum(F,2) % do the cumulative sum of each row
```

```
cumsum_F_row = 3×3
     1     5    12
     2     7    15
     3     9    18
```

```
cumsum_F_all = cumsum(F(:));% do the cumulative sum of all the elements in F
cumsum_F_all'
```

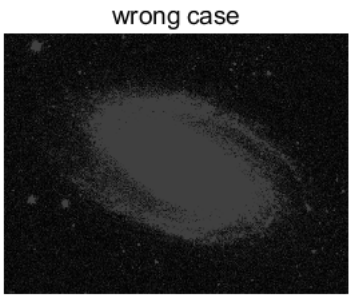
```
ans = 1×9
     1     3     6    10    15    21    28    36    45
```

Examples of digital image processing

- Addition of images

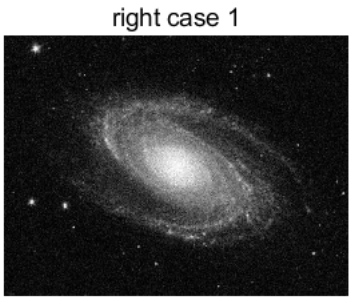
```
clear all;clf;clc;
Noise1 = imread('Noisy_image1.jpg');
Noise2 = imread('Noisy_image2.jpg');
Noise3 = imread('Noisy_image3.jpg');
Noise4 = imread('Noisy_image4.jpg');

% Wrong Case!
wrong_case = (Noise1 + Noise2 + Noise3 + Noise4)/4;
figure(8);imshow(wrong_case);title('wrong case');
```



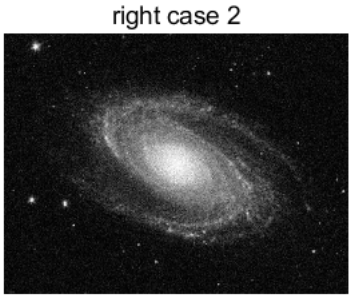
Four original images are all uint8 data type. When we add them, some of the addition result are larger than 255. However, the value of the intensity cannot larger than 255. Hence, we have to do division first and then add them.

```
% Right Case!
right_case1 = Noise1/4 + Noise2/4 + Noise3/4 + Noise4/4;
figure(9);imshow(right_case1);title('right case 1');
```



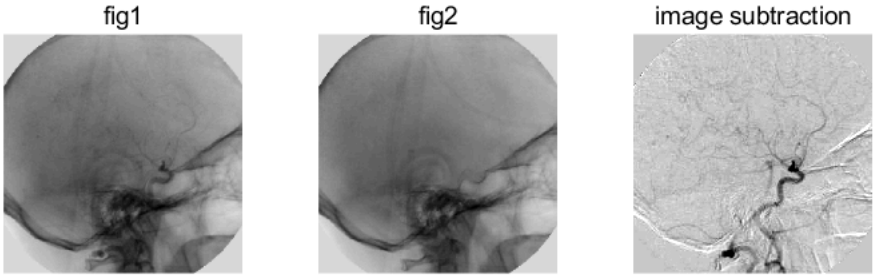
Here gives another way to do addition by changing data type.

```
right_case2 = (double(Noise1) + double(Noise2) + double(Noise3) + double(Noise4))/4;
figure(10);imshow(uint8(right_case2));title('right case 2');
```



- Subtraction of images

```
clear all;clf;
fig1 = imread("Fig1.tif");
fig2 = imread("Fig2.tif");
details = double(fig1)-double(fig2); % Using subtraction to find details in two similar images
n = 4; % Gain of details. (Larger value for higher contrast details)
background = 200; % intensity of the image background
result = background+n*(details); % Synthesize a new image
figure(11);
subplot(1,3,1);imshow(fig1);title('fig1');
subplot(1,3,2);imshow(fig2);title('fig2');
subplot(1,3,3);imshow(uint8(result));title('image subtraction')
```

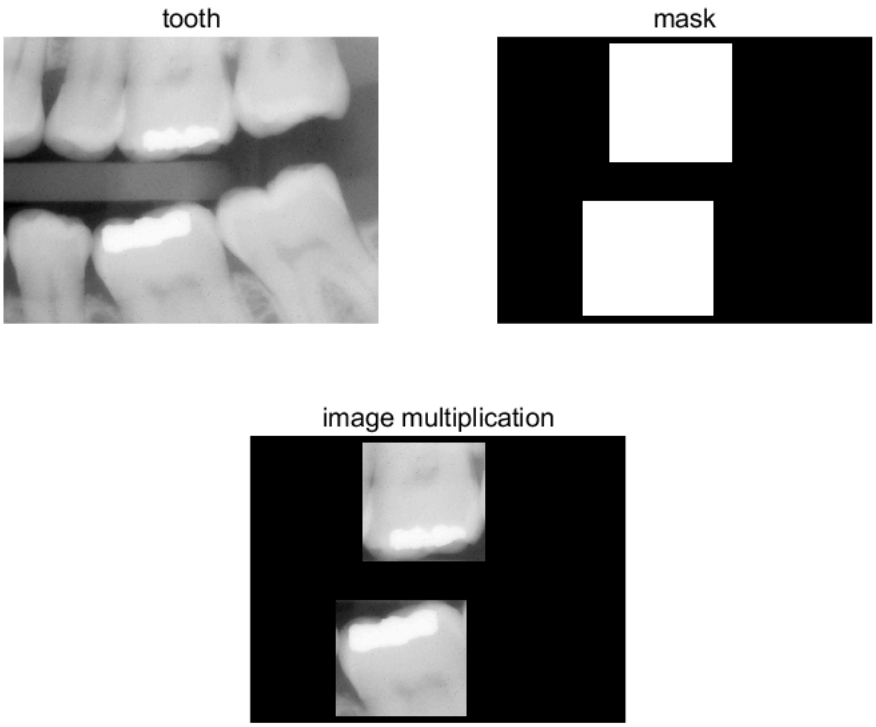


- Multiplication of images

```
clear all;clf;
tooth = imread('tooth.tif');
mask = imread('mask.tif');

% Similarly as the addtion part, we cannot do multiplication directly.
% Here only shows one of the method to do multiplication.
% You can try the method of changing data type.

normalized_mask = mask/255; % Do the normalization of the mask.
result = tooth.*normalized_mask; % Then do the multiplication.
figure(12);
subplot(2,2,1);imshow(tooth);title('tooth');
subplot(2,2,2);imshow(mask);title('mask');
subplot(2,2,[3,4]);imshow(result);title('image multiplication');
```



- Division of images

```
clear all;clf;
fiber = imread('fiber.tif');
background = imread('bkg.tif');

result = double(fiber)./double(background); % Do the division in 'double' data type.
figure(13);
subplot(1,3,1);imshow(fiber);title('fiber');
subplot(1,3,2);imshow(background);title('background');
subplot(1,3,3);imshow(result);title('image division');
```

- Background removal

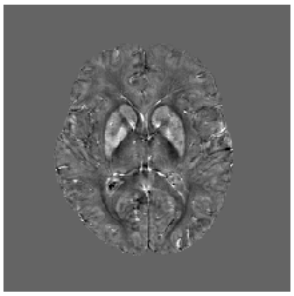
```
clear all;clf;
load slice.mat slice
load mask.mat mask

% There are also many ways to achieve this result.
% Here only shows one method to remove background.
% For different input data, the method of processing may also be different.

n = 5; % Gain of the input details to reach higher contrast.
background = 0.4; % Since the input slice data is not all positive, we add a background to the
new_slice = n*slice + background;
removal_bkg = new_slice.*double(mask); % Using mask to remove the useless background.

figure(14);% note the data type & data range,we should use function 'im2uint8()' here.
subplot(2,2,1);imshow(im2uint8(new_slice));title('slice');
subplot(2,2,2);imshow(mask*255);title('mask');
subplot(2,2,[3,4]);imshow(im2uint8(removal_bkg));title('background removal');
```

slice



mask

background removal