Lab 3 Log Book

Task 1 - Contrast enhancement with function imadjust

Importing an image

```
clear all
imfinfo('assets/breastXray.tif') %Retrieve image file information
f = imread('assets/breastXray.tif');
imshow(f)
```

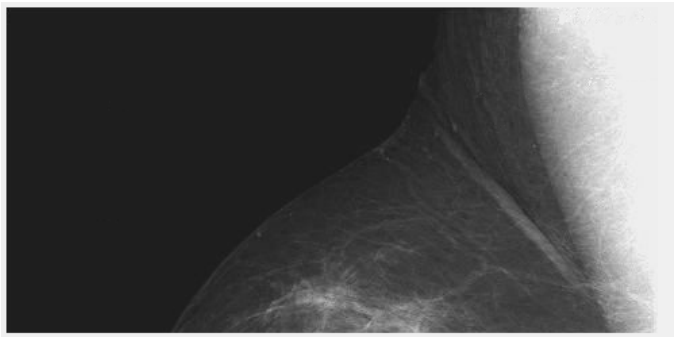
Output:

```
f(3,10)            % print the intensity of pixel(3,10)
imshow(f(1:241,:))  % display only top half of the image
```

Output: ans = 28

f(3,10): Access the pixel value at the 3rd row, 10th column of matrix f, and print it in the command window.

imshow(f(1:241,:)): Crop and display the upper half of the image, showing only the pixels of the first 241 lines while keeping the number of columns unchanged. Select the first 1 to 241 rows of f and retain all columns.



To find the maximum and minimum intensity values of the image, do this:

```
[fmin, fmax] = bounds(f(:))
```

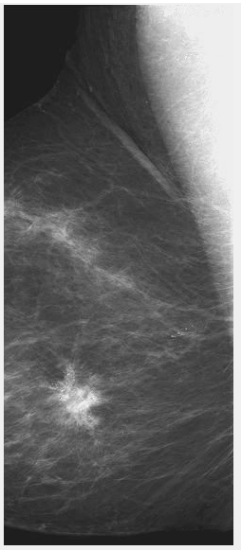
Output：fmin = 21， fmax = 255

bounds returns the maximum and minimum values in the entire image f. The index (:) means every columns. If this is not specified, Matlab will return the max and min values for each column as 2 row vectors.

Since the data type for *f* is *uint8*, the full intensity range is [0 255]. Is the intensity of *f* close to the full range? Answer： Yes the range[21 255] is close to the full range.

Display the right half of the image. Capture it for your logbook.

```
figure
imshow(f(:,241:482));
```



Output: Matlab uses rows and columns to represent matrices, so to retain the right half, it is necessary to ensure all rows and the columns of the right half. Therefore, the first colon indicates reading all rows, and 241:482 indicates reading the columns of the right half.

Negative image

To compute the negative image and display both the original and the negative image side-by-side, do this:

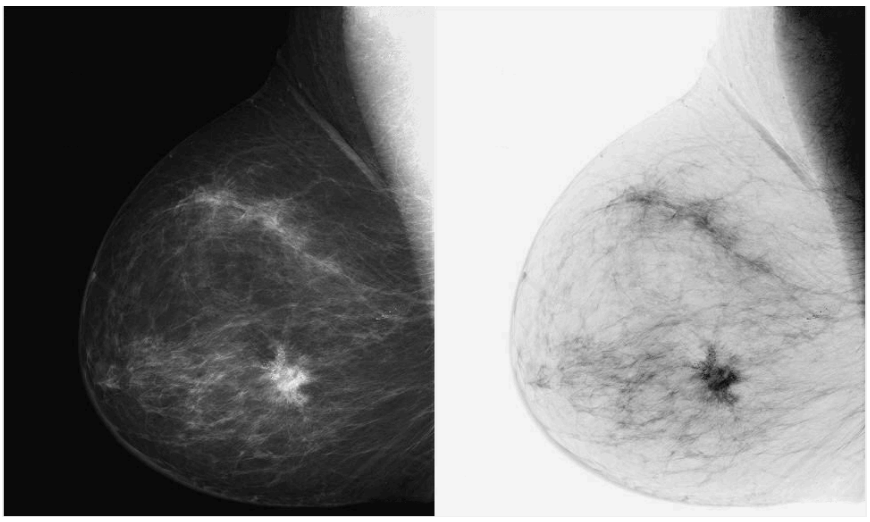
```
g1 = imadjust(f, [0 1], [1 0])  % Perform contrast inversion
figure                          % open a new figure window
imshowpair(f, g1, 'montage')
```

Output: imadjust(f, [low_in high_in], [low_out high_out])

[0 1] Representing from darkest to brightest

[1 0] Indicate pixel value inversion

If the pixel value of a position is 0.2, after flipping it becomes 1 - 0.2 = 0.8.



> The 2nd parameter of imadjust is in the form of [low_in high_in], where the values are between 0 and 1. [0 1] means that the input image is to be adjusted to within 1% of the bottom and top pixel values.

> The 3rd parameter is also in the form of [low_out high_out]. It specifies how the input range is mapped to output range. So,

> [1 0] means that the lowest pixel intensity of the input is now mapped to highest pixel intensity at the output and vice versa. This of course means that all intensities are inverted, producing the negative image. The same thing can be achieved using function imcomplement.

Gamma correction

```
g2 = imadjust(f, [0.5 0.75], [0 1]); %Adjust the contrast so that the range from
g3 = imadjust(f, [ ], [ ], 2); %Perform gamma correction, gamma = 2
figure
montage({g2,g3})
```

g2 has the gray scale range between 0.5 and 0.75 mapped to the full range.

g3 uses gamma correct with gamma = 2.0 as shown in the diagram below. [ ] is the same as [0 1] by default.

Output:

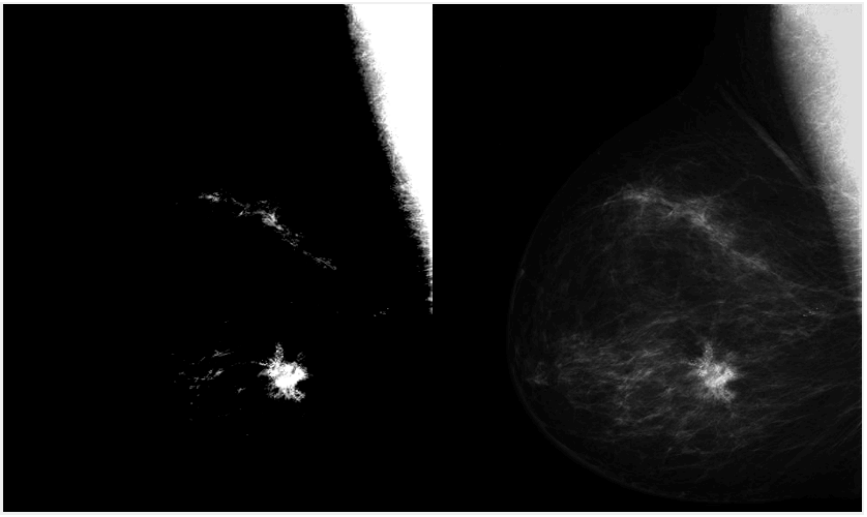
g2 Enhance the contrast of medium brightness areas (0.5 0.75) without affecting the darkest or brightest areas. The parts of the original image with pixel values between 0.5 and 0.75 will be stretched to the new range [0 1].

g3 Perform Gamma correction, enhance dark area details, and reduce brightness contrast.

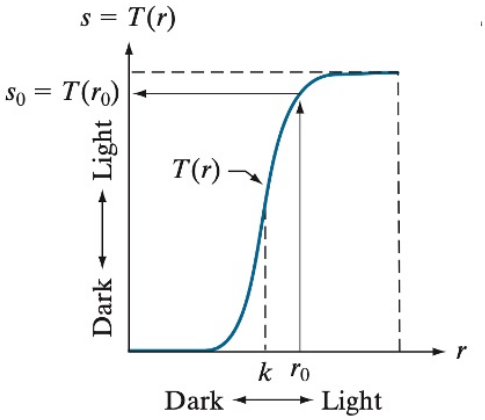
If γ > 1, the image becomes brighter, and the details in the dark areas are enhanced.

If γ < 1, the image becomes darker, and the details in the bright areas are enhanced.

g2 and g3 perform similar, but gamma correction keep more details.



Task 2: Contrast-stretching transformation



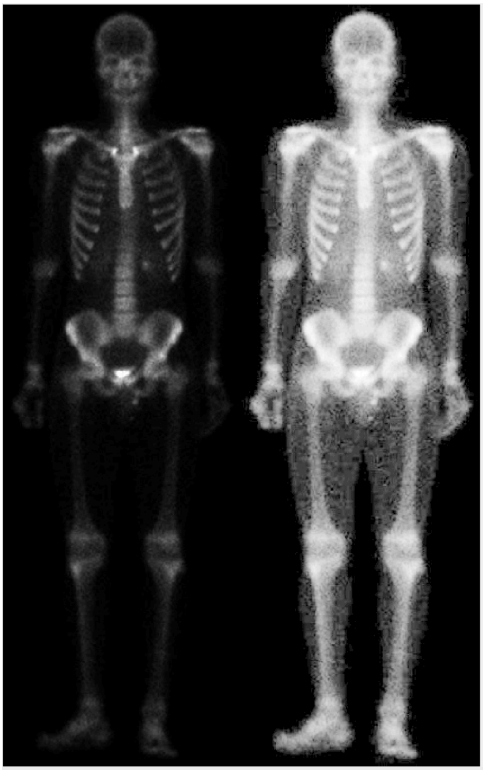
The equation of this function is:

$$s = T(r) = \frac{1}{1 + (k/r)^E}$$

where *k* is often set to the average intensity level and E determines steepness of the function

```
clear all       % clear all variables
close all       % close all figure windows
f = imread('assets/bonescan-front.tif');
r = double(f);  % uint8 to double conversion
k = mean2(r);   % find mean intensity of image
E = 0.9;
s = 1 ./ (1.0 + (k ./ (r + eps)) .^ E);  % Contrast enhancement transformation
% The intensity values of s are normalized to the range of [0.0 1.0] and is in ty
g = uint8(255*s);  %Normalize the channel [0 255], and convert to uint8
imshowpair(f, g, "montage")
```

Output：This image processing method is based on an S-type (sigmoid-like) transformation to adjust the pixel values of the image.



The right image (processed) has significantly enhanced contrast compared to the left image (original). Bones, especially the ribs, pelvis, and skull, appear **brighter and more distinguishable** in the processed image. Soft tissues that were barely visible in the original image have become more pronounced

The sigmoid-like transformation with exponent E=0.9 ensures that mid-intensity regions (such as bones) are enhanced without oversaturating bright areas.

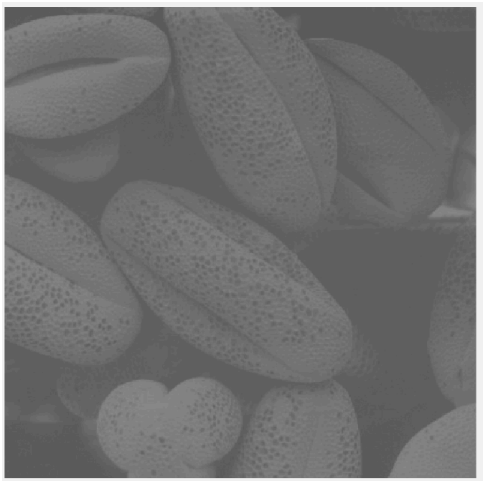
The processed image retains more gradual intensity variations, making it more readable for medical analysis.

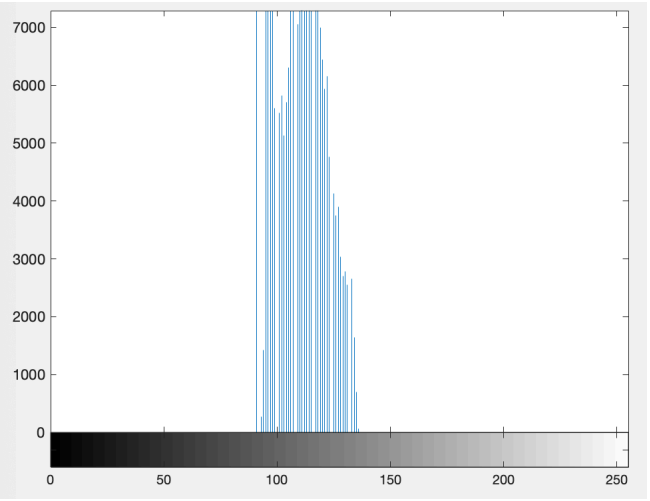
Task 3: Contrast Enhancement using Histogram

PLotting the histogram of an image

```
clear all       % clear all variable in workspace
close all       % close all figure windows
f=imread('assets/pollen.tif');
imshow(f)
figure          % open a new figure window
imhist(f);      % calculate and plot the histogram
```

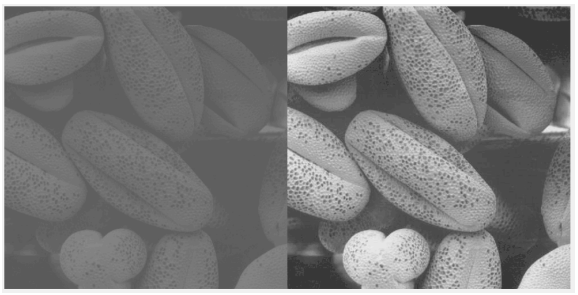
Output: The histogram shows the distribution of pixels between 0-255. It is clear that the intesity level of this image is very much squashed up between 70 to 140.

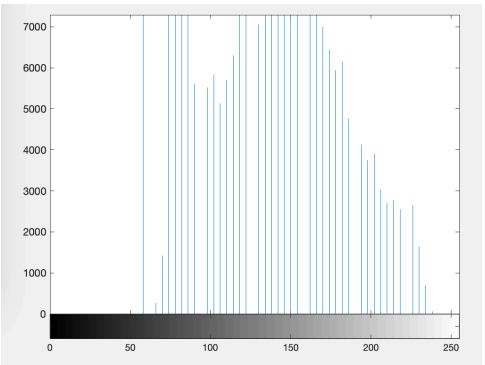





```
close all
g=imadjust(f,[0.3 0.55]); % stretch the intensity between 0.3 and 0.55 of full s
montage({f, g})     % display list of images side-by-side
figure
imhist(g);
```

Output：The histogram of the adjusted image is more spread out. The image after stretching enhances the contrast of the image, making it appear clearer.

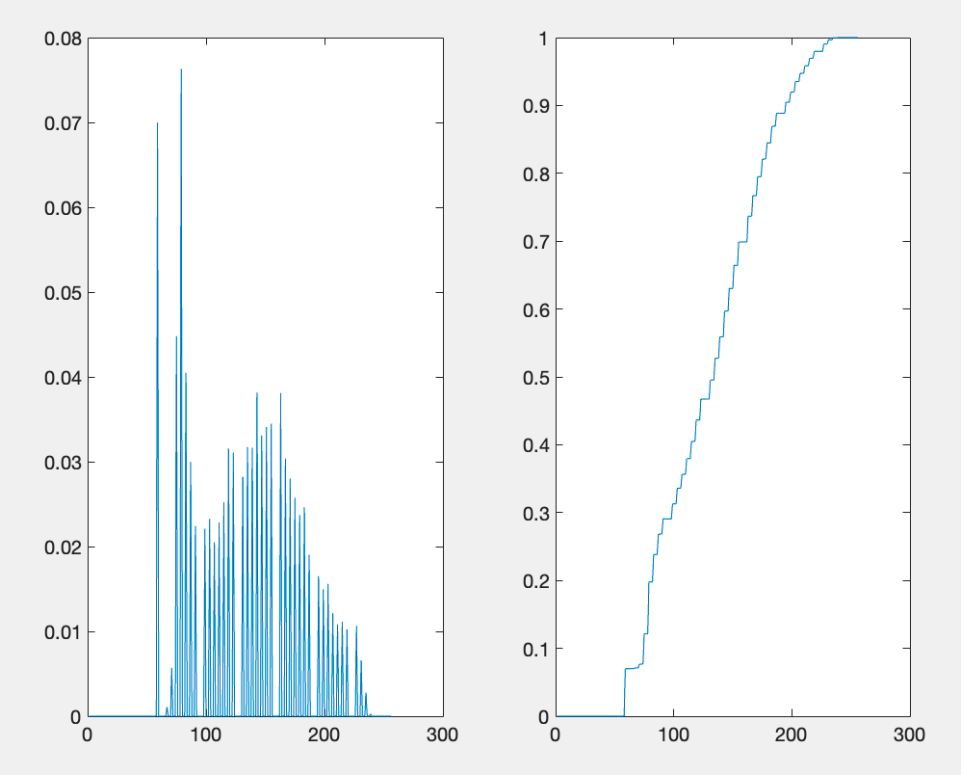




Histogram, PDF and CDF

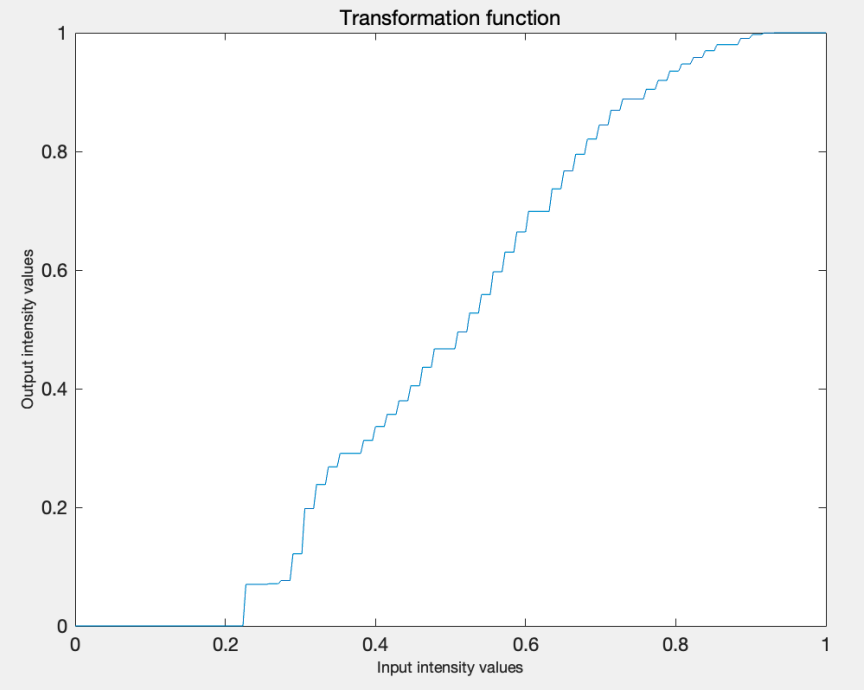
```
g_pdf = imhist(g) ./ numel(g);  % compute PDF
g_cdf = cumsum(g_pdf);          % compute CDF
close all                       % close all figure windows
```

```
imshow(g);
subplot(1,2,1)              % plot 1 in a 1×2 subplot
plot(g_pdf)
subplot(1,2,2)               % plot 2 in a 1×2 subplot
plot(g_cdf)
```



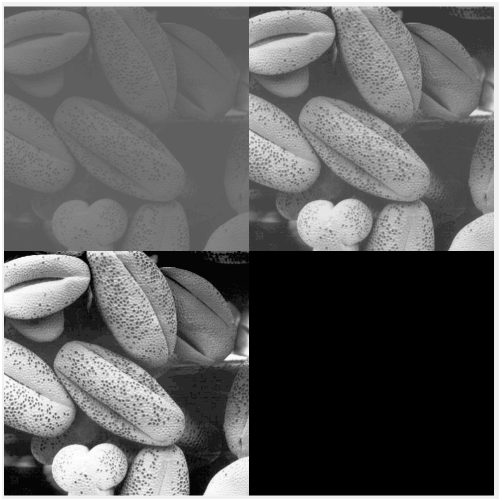
Histogram Equalization

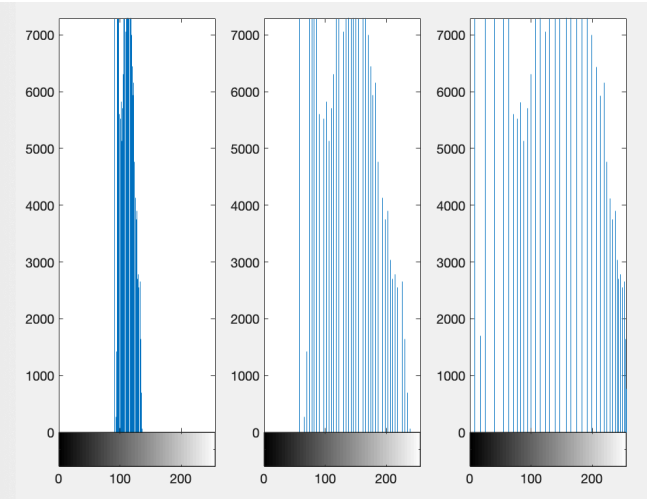
```
x = linspace(0, 1, 256);    % x has 256 values equally spaced
%  .... between 0 and 1
figure
plot(x, g_cdf)
axis([0 1 0 1])             % graph x and y range is 0 to 1
set(gca, 'xtick', 0:0.2:1)  % x tick marks are in steps of 0.2
set(gca, 'ytick', 0:0.2:1)
xlabel('Input intensity values', 'fontsize', 9)
ylabel('Output intensity values', 'fontsize', 9)
title('Transformation function', 'fontsize', 12)
```




```
h = histeq(g,256);          % histogram equalize g
close all
montage({f, g, h})
figure;
subplot(1,3,1); imhist(f);
subplot(1,3,2); imhist(g);
subplot(1,3,3); imhist(h);
```

Output: h is the histogram equalization image, which makes the overall grayscale range of the image more uniform and enhances the contrast. Compared to g (which only enhances the grayscale values of the original image f in the range of 0.3-0.55), the distribution of h is more uniform and the effect is better.



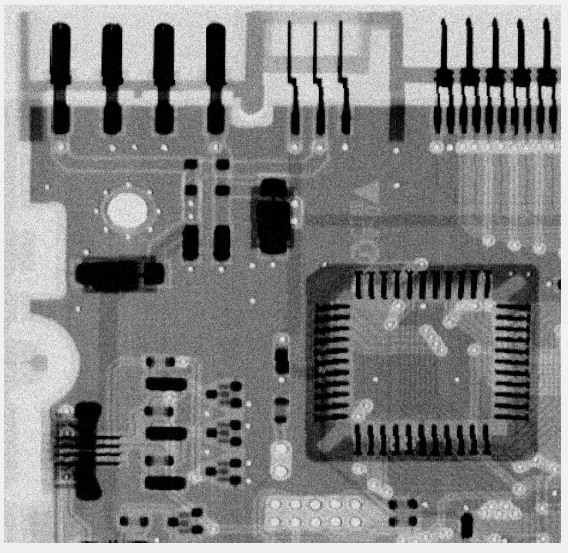


Task 4 - Noise reduction with lowpass filter

Matlab provides a function called *fspecial*, which returns different types of filter kernels. The table below shows the types of kernels that can generated.

Type	Syntax and Parameters
`'average'`	`fspecial('average', [r c])`. A rectangular averaging filter of size `r` × `c`. The default is 3 × 3. A single number instead of `[r c]` specifies a square filter.
`'disk'`	`fspecial('disk', r)`. A circular averaging filter (within a square of size `2r + 1`) with radius `r`. The default radius is 5.
`'gaussian'`	`fspecial('gaussian', [r c], sig)`. A Gaussian lowpass filter of size `r` × `c` and standard deviation `sig` (positive). The defaults are 3 × 3 and 0.5. A single number instead of `[r c]` specifies a square filter.
`'laplacian'`	`fspecial('laplacian', alpha)`. A 3 × 3 Laplacian filter whose shape is specified by `alpha`, a number in the range [0, 1]. The default value for `alpha` is 0.5.
`'log'`	`fspecial('log', [r c], sig)`. Laplacian of a Gaussian (LoG) filter of size `r` × `c` and standard deviation `sig` (positive). The defaults are 5 × 5 and 0.5. A single number instead of `[r c]` specifies a square filter.
`'motion'`	`fspecial('motion', len, theta)`. Outputs a filter that, when convolved with an image, approximates linear motion (of a camera with respect to the image) of `len` pixels. The direction of motion is `theta`, measured in degrees, counterclockwise from the horizontal. The defaults are 9 and 0, which represents a motion of 9 pixels in the horizontal direction.
`'prewitt'`	`fspecial('prewitt')`. Outputs a 3 × 3 Prewitt mask, `wv`, that approximates a vertical gradient. A mask for the horizontal gradient is obtained by transposing the result: `wh = wv'`.
`'sobel'`	`fspecial('sobel')`. Outputs a 3 × 3 Sobel mask, `sv`, that approximates a vertical gradient. A mask for the horizontal gradient is obtained by transposing the result: `sh = sv'`.
`'unsharp'`	`fspecial('unsharp', alpha)`. Outputs a 3 × 3 unsharp filter. Parameter `alpha` controls the shape; it must be greater than 0 and less than or equal to 1.0; the default is 0.2.

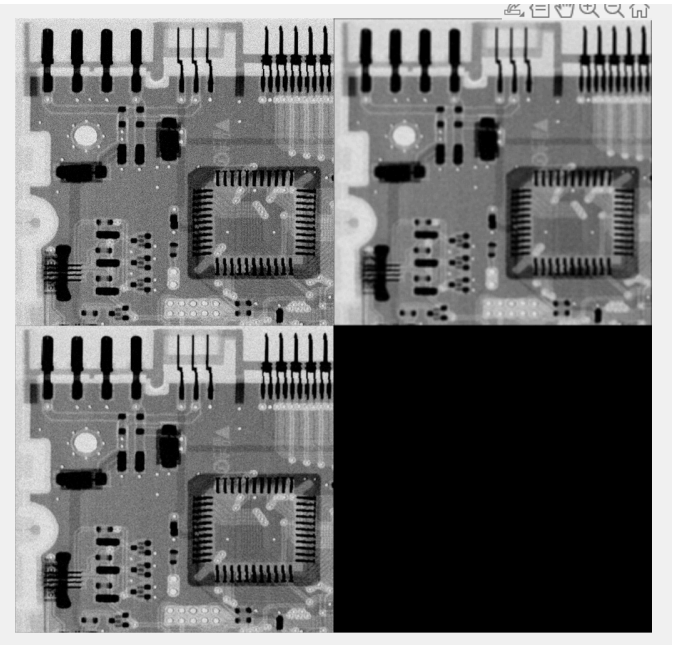
```
clear all
close all
f = imread('assets/noisyPCB.jpg');
imshow(f)
```



```
w_box = fspecial('average', [9 9])  % Generate a 9×9 mean filter (average filte
w_gauss = fspecial('Gaussian', [7 7], 1.0) %Generate a 7×7 Gaussian filter witl
g_box = imfilter(f, w_box, 0);
g_gauss = imfilter(f, w_gauss, 0);
figure
montage({f, g_box, g_gauss})
```

Output：g_box: Appears **blurrier**, noise is reduced but fine details are also lost.

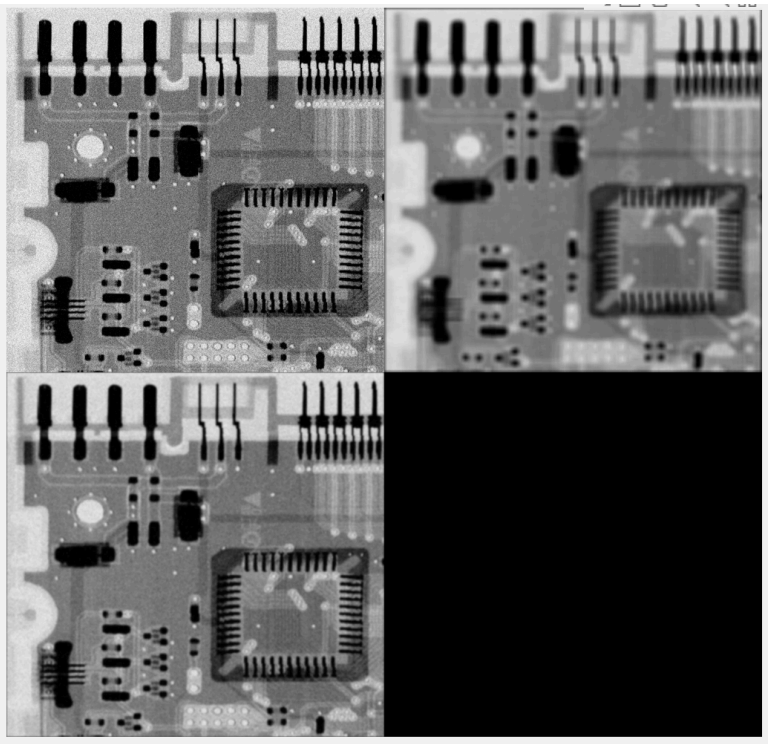
g_gauss: Shows **smoother reduction in noise**, but edges and structures are preserved better than the box filter.



```
w_box = fspecial('average', [15 15])
w_gauss = fspecial('Gaussian',  [11 11], 2.0)
g_box = imfilter(f, w_box, 0);
g_gauss = imfilter(f, w_gauss, 0);
figure
montage({f, g_box, g_gauss})
```

Output：Using a larger average filter [15 15], the image becomes significantly more blurred, the noise reduction effect is enhanced, but the loss of edge details is more severe.

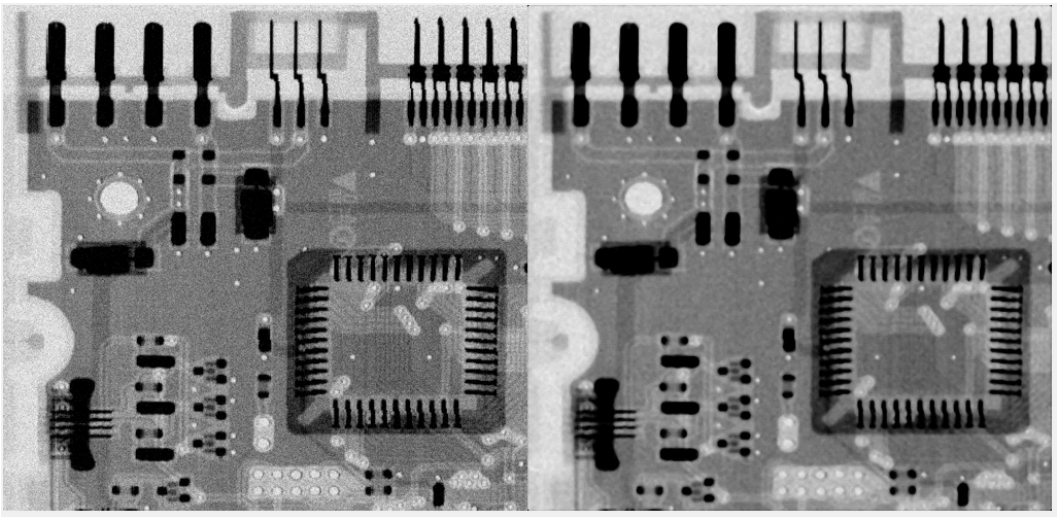
After using gauss[11 11], σ=2.0, the noise reduction effect is stronger than before, but it still retains some details. Due to σ=2.0, the Gaussian filtering smooths more, but retains more edge information than the mean filter.



Task 5 - Median Filtering

```
g_median = medfilt2(f, [7 7], 'zero'); %Apply a 7×7 median filter
figure; montage({f, g_median})
```

Output：The median filter uses a 7×7 window, calculates the median value of all pixels within the window, and then uses this value to replace the central pixel. Compared to the box filter, it more clearly preserves the edges and removes noise.



Task 6 - Sharpening the image with Laplacian, Sobel and Unsharp filters

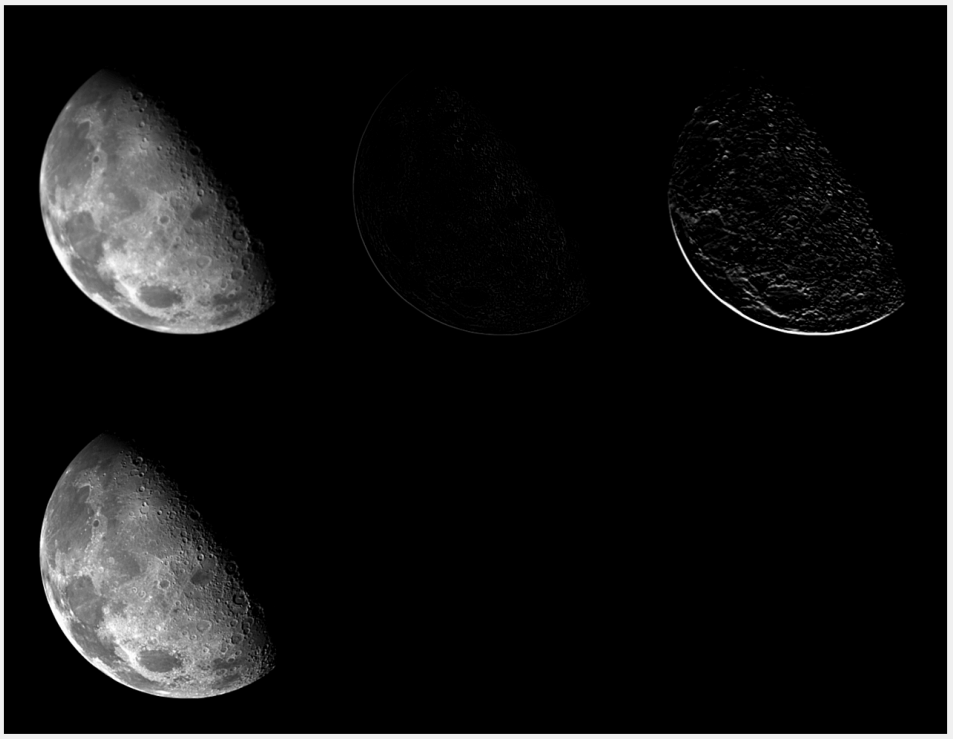
```
clear all
close all
f = imread('assets/moon.tif');

imshow(f)    % Display the original image
% Define different filters
w_laplacian = fspecial('laplacian', 0.5); % Create a Laplacian filter with alpha =
w_sobel = fspecial('sobel');              % Create a Sobel filter for edge detection
w_unsharp = fspecial('unsharp', 0.5);     % Create an Unsharp filter to enhance
% Apply the filters to the image using imfilter()
g_laplacian = imfilter(f, w_laplacian, 0);  % Apply Laplacian filter for edge dete
g_sobel = imfilter(f, w_sobel, 0);          % Apply Sobel filter for directional edge
g_unsharp = imfilter(f, w_unsharp, 0);      % Apply Unsharp mask filter for shar

figure
montage({f, g_laplacian, g_sobel, g_unsharp})
```

Output：

g_laplacian: Produces an edge-enhanced version where bright edges appear against a dark background. Captures high-frequency details (moon edges). The edges are highlighted symmetrically in all directions.

g_sobel: The moon's left edge is strongly highlighted, indicating horizontal edge detection. The edge detection in the vertical direction is not obvious, but compared to the Laplacian, the details of the meteor crater are preserved, and the edges are clearer.

g_unsharp: Enhances the sharpness of the image by increasing local contrast. The craters and textures appear more pronounced, making the details more visible. Unlike Laplacian and Sobel, this filter does not just detect edges but enhances the entire image’s sharpness.



Task 7 - Test yourself Challenges

- Improve the contrast of a lake and tree image store in file *lake&tree.png* use any technique you have learned in this lab. Compare your results with others in the class.

```
clear all
close all
f = imread('assets/lake&tree.png');  % Read the input image
f_gray = im2gray(f);  % Convert to grayscale

% 1. Box Filter (Average Filter for Smoothing)
w_box = fspecial('average', [5 5]);  % 5×5 average filter
g_box = imfilter(f_gray, w_box, 'replicate');  % Apply smoothing

% 2. Sobel Edge Detection
g_sobel = edge(f_gray, 'sobel');  % Apply Sobel edge detection

% 3. Unsharp Masking (Sharpening)
w_unsharp = fspecial('unsharp', 0.4);  % Define the unsharp filter
g_unsharp = imfilter(f_gray, w_unsharp, 'replicate');  % Apply sharpening
```

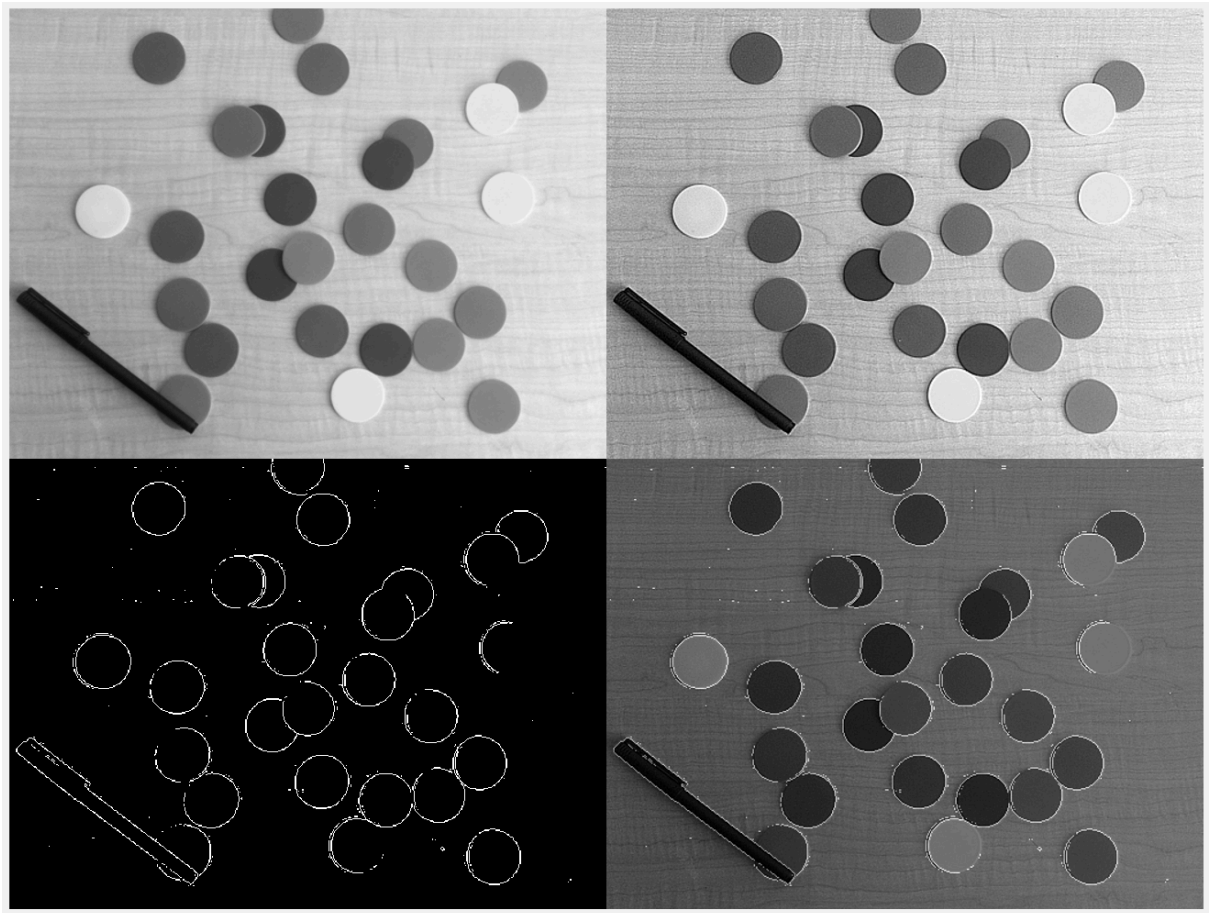
```
% 4. Combined Enhancement (Merging Sobel, Box Filter, Unsharp Masking, ar
g_combined = imfuse(g_unsharp, g_sobel, 'blend');  % Sharpening + Sobel ed
g_combined = imfuse(g_combined, g_box, 'blend');   % Blend with Box filter fo
g_final = imfuse(g_combined, histeq(f_gray), 'blend');  % Merge with Histogra

% 5. Display results using montage
figure;
montage({f_gray, g_sobel, g_box, g_unsharp, g_final}, 'Size', [2 3]);
```

Output：

- Use the Sobel filter in combination with any other techniques, find the edge of the circles in the image file *circles.tif*. You are encouraged to discuss and work with your classmates and compare results.

```
clear all
close all

f = imread('assets/circles.tif');
f_gray = im2gray(f);

% 1. Apply Unsharp Masking (Enhancing Edge Details)
w_unsharp = fspecial('unsharp', 0.5);  % Strength of sharpening
g_unsharp = imfilter(f_gray, w_unsharp, 'replicate');

% 2. Apply Sobel Edge Detection on the Sharpened Image
g_sobel = edge(g_unsharp, 'sobel');

% 3. Combine Sobel Edge Detection with Unsharp Masked Image
g_final = imfuse(g_sobel, g_unsharp, 'blend');

figure;
montage({f_gray, g_unsharp, g_sobel, g_final}, 'Size', [2 2]);
```

Output：



- *office.jpg* is a colour photograph taken of an office with badd exposure. Use whatever means at your disposal to improve the lighting and colour of this photo.

```
clear all
close all

f = imread('assets/office.jpg');
% Gamma correction (overall brightness enhancement)
gamma = 1.4;  %gamma > 1, increase brightness while maintaining contrast
f_gamma = imadjust(f, [], [], gamma);

% Color Enhancement (Increase Saturation)
f_hsv = rgb2hsv(f_gamma);
f_hsv(:,:,2) = imadjust(f_hsv(:,:,2), [], [], 1.2);  % Increase Saturation
f_final = hsv2rgb(f_hsv);
```

```
figure;
montage({f, f_gamma, f_final}, 'Size', [2 2]);
```

Output：