% Histogram Equalization in MATLAB

```
% Step 1: Read the grayscale image
original_img = imread('your_image.jpg');  % Replace with your image file
gray_img = rgb2gray(original_img);
% Convert to grayscale if needed
% Step 2: Perform histogram equalization
equalized_img = histeq(gray_img);
% Step 3: Display original and equalized images
figure;
subplot(2,2,1);
imshow(gray_img);
title('Original Grayscale Image')
subplot(2,2,2);
imhist(gray_img);
title('Original Histogram');
subplot(2,2,3);
imshow(equalized_img);
title('Histogram Equalized Image');
subplot(2,2,4);
imhist(equalized_img);
title('Equalized Histogram');
```

% Read, Transform, and Rotate an Image in MATLAB

```
% Step 1: Read the image
original_img = imread('your_image.jpg');  % Replace with your actual image file
% Step 2: Convert to grayscale if it's a color image
if size(original_img, 3) == 3
   gray_img = rgb2gray(original_img);
else
   gray_img = original_img;
end
% Step 3: Apply image transformation (e.g., scaling by 0.5)
scale_factor = 0.5;
transformed_img = imresize(gray_img, scale_factor);
% Step 4: Apply rotation (e.g., 45 degrees)
rotated_img = imrotate(transformed_img, 45)
% Step 5: Display the images
figure;
subplot(1, 3, 1);
imshow(gray_img);
title('Original Grayscale Image')
subplot(1, 3, 2);
imshow(transformed_img);
title('Scaled Image (50%)')
subplot(1, 3, 3)
imshow(rotated_img);
title('Rotated Image (45°)');
```

Spatial + Frequency Domain Filtering

```
% Read and convert image
img = im2double(imread('your_image.jpg'));
if size(img,3) == 3, img = rgb2gray(img); end

% Spatial Filtering (Averaging)
spatial_filtered = imfilter(img, fspecial('average', [5 5]), 'replicate');

% Frequency Domain Filtering (Ideal Low-pass)
[M,N] = size(img);
F = fftshift(fft2(img));
[u,v] = meshgrid(-N/2:N/2-1, -M/2:M/2-1);
H = double(sqrt(u.^2 + v.^2) <= 50);
F_filtered = ifft2(ifftshift(F .* H));

% Display
figure;
subplot(1,3,1), imshow(img), title('Original');
subplot(1,3,2), imshow(spatial_filtered, []), title('Spatial Filtered');
subplot(1,3,3), imshow(real(F_filtered), []), title('Freq. Domain Filtered');
```

Linear Filtering using Convolution

```
% Read and convert image
img = im2double(imread('your_image.jpg'));  % Replace with your image
if size(img,3) == 3, img = rgb2gray(img); end

% Define a linear filter kernel (e.g., 3x3 averaging filter)
kernel = ones(3,3) / 9;

% Apply linear filtering using convolution
filtered_img = conv2(img, kernel, 'same');

% Display results
figure;
subplot(1,2,1), imshow(img), title('Original Image');
subplot(1,2,2), imshow(filtered_img), title('Filtered Image (Convolution)');
```

Smoothing in Spatial Domain

```
% Read and convert image
img = im2double(imread('your_image.jpg'));  % Replace with your image
if size(img,3) == 3, img = rgb2gray(img); end

% Create a smoothing (averaging) filter
h = fspecial('average', [5 5]);  % 5x5 averaging kernel

% Apply the filter
smoothed_img = imfilter(img, h, 'replicate');

% Display original and smoothed images
figure;
subplot(1,2,1), imshow(img), title('Original Image');
subplot(1,2,2), imshow(smoothed_img), title('Smoothed Image');
```

Image Type Conversion

```
% Read the image
img = imread('your_image.jpg');  % Replace with your image
if size(img,3) == 3
   gray_img = rgb2gray(img);    % Convert to grayscale if it's a color image
else
   gray_img = img;
end
% Convert to double
img_double = im2double(gray_img);
% Convert to uint8
img_uint8 = im2uint8(img_double);
% Convert to logical (binary image using threshold)
img_logical = imbinarize(img_double);
% Convert grayscale to RGB
img_rgb = cat(3, gray_img, gray_img, gray_img);
% Display all types
figure;
subplot(2,3,1), imshow(gray_img), title('Original Grayscale (uint8)');
subplot(2,3,2), imshow(img_double), title('Converted to Double');
subplot(2,3,3), imshow(img_uint8), title('Converted back to Uint8');
subplot(2,3,4), imshow(img_logical), title('Converted to Logical');
subplot(2,3,5), imshow(img_rgb), title('Grayscale to RGB');
```

Negative of an Image

```
% Read the image
img = imread('your_image.jpg');  % Replace with your image file

% Convert to grayscale if it's a color image
if size(img, 3) == 3
   img = rgb2gray(img);
end

% Create the negative image
negative_img = 255 - img;

% Display the original and negative images
figure;
subplot(1,2,1), imshow(img), title('Original Image');
subplot(1,2,2), imshow(negative_img), title('Negative Image');
```

Cropping an Image

```
% Step 1: Read the image
img = imread('your_image.jpg');  % Replace with your image file
% Step 2: Define the crop region (rectangular section of the image)
% The format for the crop region is [x, y, width, height]
% Example: Crop a region starting at (100, 50) with a width of 200 and height of 150
crop_region = [100, 50, 200, 150];  % Modify these values as needed
% Step 3: Perform the cropping
cropped_img = imcrop(img, crop_region)
% Step 4: Display the original and cropped images using subplots
figure;
% Display the original image
subplot(1, 2, 1);
imshow(img);
title('Original Image');
% Display the cropped image
subplot(1, 2, 2);
imshow(cropped_img);
title('Cropped Image');
```

Zooming into an Image

```
% Step 1: Read the image
img = imread('your_image.jpg');  % Replace with your image file
% Step 2: Define the region to zoom into (e.g., a specific rectangular region)
% The format for the zoom region is [x, y, width, height]
% Example: Zoom into a region starting at (100, 100) with width 150 and height 150
zoom_region = [100, 100, 150, 150];  % Modify these values as needed
% Step 3: Crop the region to zoom into
zoomed_img = imcrop(img, zoom_region);
% Step 4: Enlarge the cropped image (zooming effect)
zoomed_in_img = imresize(zoomed_img, 2);  % Resize by a factor of 2 (zoom in)
% Step 5: Display the original and zoomed-in images using subplots
figure;
% Display the original image
subplot(1, 2, 1);
imshow(img);
title('Original Image');
% Display the zoomed-in image
subplot(1, 2, 2);
imshow(zoomed_in_img);
title('Zoomed-in Image');
```

Mirror Image and Flips with Titles and Subplots

```
% Step 1: Read the image
img = imread('your_image.jpg');  % Replace with your image file
% Step 2: Create the mirror image (horizontal flipping)
mirror_img = fliplr(img);
% Step 3: Create Horizontal Flip
horizontal_flip = fliplr(img);
% Step 4: Create Vertical Flip
vertical_flip = flipud(img);
% Step 5: Create Horizontal and Vertical Flip (180-degree flip)
both_flip = flipud(fliplr(img));
% Step 6: Display the results using subplots
figure;
% Original Image
subplot(2, 3, 1);  % Position of the subplot
imshow(img);  % Display the original image
title('Original Image');  % Title for the original image
% Mirror Image (Horizontal Flip)
subplot(2, 3, 2);  % Position of the subplot
imshow(mirror_img);  % Display the mirror image
title('Mirror Image (Horizontal Flip)');  % Title for mirror image
% Horizontal Flip
subplot(2, 3, 3);  % Position of the subplot
imshow(horizontal_flip);  % Display horizontal flip
title('Horizontal Flip');  % Title for horizontal flip
% Vertical Flip
subplot(2, 3, 4);  % Position of the subplot
imshow(vertical_flip);  % Display vertical flip
title('Vertical Flip');  % Title for vertical flip
% Horizontal and Vertical Flip (180-degree flip)
subplot(2, 3, 5);  % Position of the subplot
imshow(both_flip);  % Display horizontal and vertical flip
title('Horizontal + Vertical Flip');  % Title for both flips
% Add a blank subplot for spacing
subplot(2, 3, 6);
axis off;  % Hide axis
title('');  % Empty title for spacing
```

Morphological Operations

```
% Step 1: Read the image
img = imread('your_image.jpg');  % Replace with your image file
if size(img, 3) == 3
    img = rgb2gray(img);  % Convert to grayscale if it's a color image
end
% Convert the image to binary for better morphological operations
binary_img = imbinarize(img);
% Step 2: Define a structuring element (e.g., a 3x3 square)
se = strel('square', 3);  % Square structuring element with size 3x3
% Step 3: Perform morphological operations
erosion_img = imerode(binary_img, se);    % Erosion
dilation_img = imdilate(binary_img, se);  % Dilation
opening_img = imopen(binary_img, se);     % Opening (erosion followed by dilation)
closing_img = imclose(binary_img, se);    % Closing (dilation followed by erosion)
boundary_img = bwperim(binary_img);       % Boundary extraction
% Step 4: Display the results using subplots
figure;
% Display the original image
subplot(2, 3, 1);
imshow(binary_img);
title('Original Binary Image')
% Display the erosion result
subplot(2, 3, 2);
imshow(erosion_img);
title('Erosion')
% Display the dilation result
subplot(2, 3, 3);
imshow(dilation_img);
title('Dilation')
% Display the opening result
subplot(2, 3, 4);
imshow(opening_img);
title('Opening');
% Display the closing result
subplot(2, 3, 5);
imshow(closing_img);
title('Closing');
% Display the boundary extraction result
subplot(2, 3, 6);
imshow(boundary_img);
title('Boundary Extraction');
```

Shrinking an Image

```
% Step 1: Read the image
img = imread('your_image.jpg');  % Replace with your image file
% Step 2: Define the shrink factor
shrink_factor = 0.5;  % Shrinking by 50% (use a value between 0 and 1)
% Step 3: Shrink the image by resizing it
shrunk_img = imresize(img, shrink_factor);
% Step 4: Display the original and shrunk images using subplots
figure;
% Display the original image
subplot(1, 2, 1);
imshow(img);
title('Original Image');
% Display the shrunk image
subplot(1, 2, 2);
imshow(shrunk_img);
title('Shrunk Image');
```
