|  |  |  |
| --- | --- | --- |
| % 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');** |  |  |
|  |  |  |