VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE, MUMBAI

**Yash Rajput**
**211060042**
**Third Year**
**Electronics Engineering**

# IMAGE AND VIDEO PROCESSING LAB - EXPERIMENT 4

Date: 11th Mar, 2024

## To Perform Morphology in Image Processing

## AIM

To perform morphology in image processing :

- Closing
- Opening
- Thickening
- Thinning
- Dilation and Erosion Comparison with MATLAB
- Dilation and Erosion of a 3x3 Matrix

## APPARATUS:

- MATLAB 2023B
- Image in tiff format

## THEORY:

In **Image Processing**, morphology refers to a set of operations used to analyze and manipulate the shape and structure of objects within images. It involves techniques such as dilation, erosion, opening, and closing, which modify the pixels in an image based on their spatial arrangement. Morphological operations are commonly used for tasks like noise reduction, object detection, and image segmentation.

## THEORY:

**Closing** is a morphological operation in image processing that is used to close small gaps and smooth the boundaries of objects in binary images. It involves performing dilation followed by erosion, which helps to fill in small holes and gaps in objects, resulting in smoother and more connected shapes. Closing is often used to preprocess images before further analysis or to improve the results of segmentation algorithms.

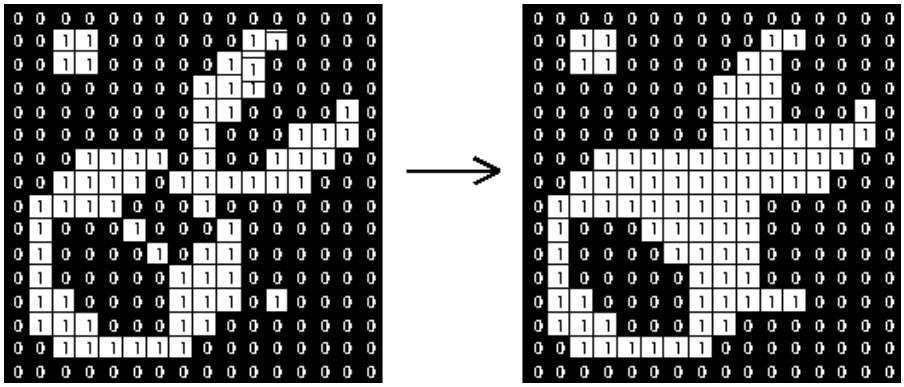

Figure 1 : Visual Representation of Closing

**Opening** is a morphological operation in image processing that involves performing erosion followed by dilation on a binary image. It helps to remove small objects and smooth out the boundaries of larger objects by eliminating small protrusions and thin connections. Opening is commonly used to remove noise and refine the segmentation of objects in images.


Figure 2 : Visual Representation of Opening

## THEORY:

**Thickening** is a morphological operation in image processing that involves expanding the boundaries of objects in a binary image. It is the opposite of erosion and aims to make objects larger by adding pixels to their edges. Thickening is often used to fill in gaps, connect disjointed regions, or enhance the representation of objects in images.

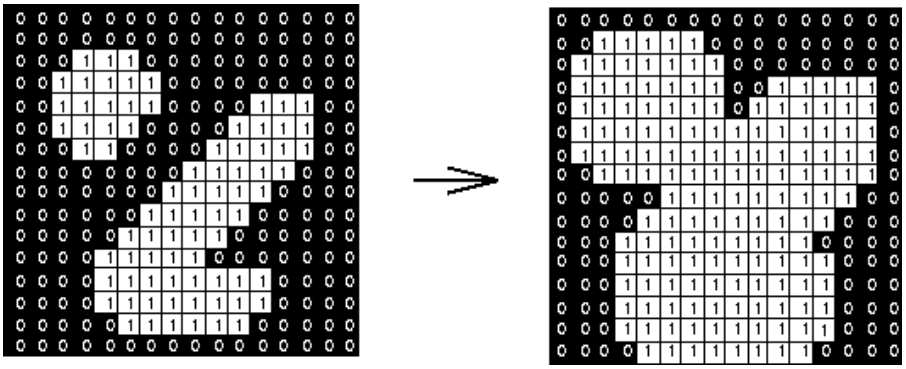

Figure 3 : Visual Representation of Thickening

**Thinning** is a morphological operation in image processing that involves reducing the width of objects in a binary image to a single-pixel thickness while preserving their essential structure and connectivity. It is the opposite of thickening and aims to simplify object boundaries for tasks like skeletonization, pattern recognition, and feature extraction in images. Thinning is commonly used to reduce computational complexity and improve the efficiency of subsequent image analysis algorithms.

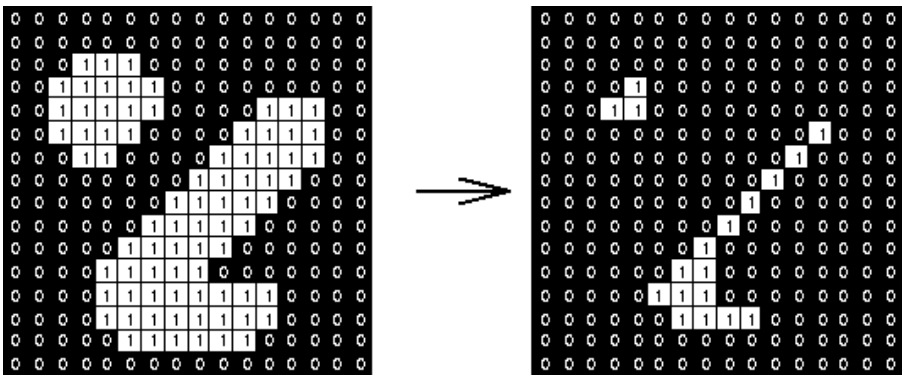

Figure 4 : Visual Representation of Thinning

## THEORY:

**Erosion** is a morphological operation in image processing where the boundaries of objects in a binary image are eroded or shrunk by removing pixels at the object's edges. It helps to separate connected objects and remove small protrusions or details, making objects appear smaller and thinner. Erosion is often used for tasks like noise reduction, segmentation, and feature extraction in images.

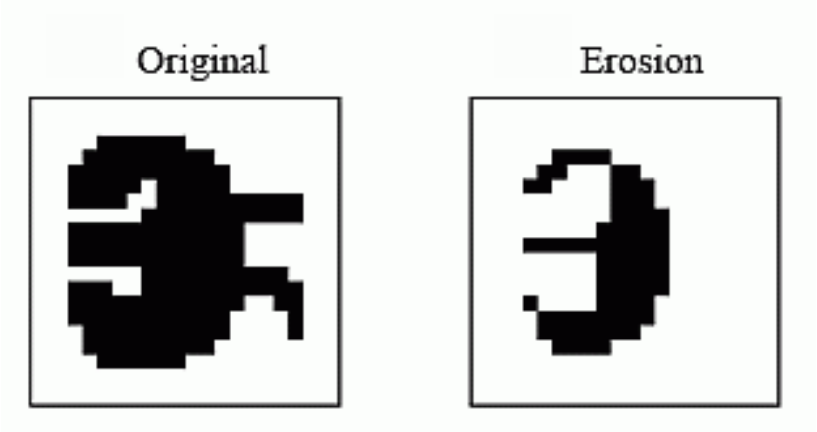


Figure 5 : Visual Representation of Erosion

**Dilation** is a morphological operation in image processing that expands or thickens the boundaries of objects in a binary image. It involves replacing each pixel in the image with the maximum pixel value in its neighborhood, effectively making objects larger and filling in small gaps or holes within them. Dilation is often used to merge nearby objects, enhance features, or perform tasks like object detection and image segmentation.

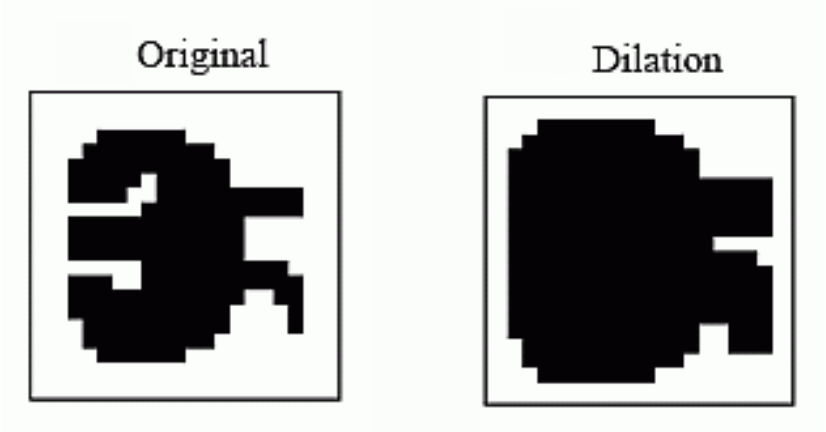


Figure 6 : Visual Representation of Dilation

## CODE:

Opening

```
% Program for Opening
% Opening means Erode first and Dilate later

% Clearing previous variables and command window
clear all
clc
% Read the input image
a = imread('bot.png');
a = double(a);
% Define the size of the image
p = size(a);
% Define the structuring element
w = [1 1 1; 1 1 1; 1 1 1];
% Perform Erosion
for x = 2:p(1)-1
    for y = 2:p(2)-1
        % Extract a 3x3 neighborhood
        neighborhood = [w(1)*a(x-1,y-1) w(2)*a(x-1,y) w(3)*a(x-1,y+1) ...
                        w(4)*a(x,y-1)   w(5)*a(x,y)   w(6)*a(x,y+1) ...
                        w(7)*a(x+1,y-1) w(8)*a(x+1,y) w(9)*a(x+1,y+1)];
        % Perform Erosion (minimum value in the neighborhood)
        A1(x,y) = min(neighborhood);
    end
    % Display the result after Erosion
    subplot(221)
    imshow(a)
    title('Original Image')

    subplot(222)
    imshow(A1)
    title('Image After Erosion')
end
% Define the size of the eroded image
q = size(A1);
% Perform Dilation
for x = 2:q(1)-1
    for y = 2:q(2)-1
        % Extract a 3x3 neighborhood from the eroded image
        neighborhood_dilation = [w(1)*A1(x-1,y-1) w(2)*A1(x-1,y) w(3)*A1(x-1,y+1) ...
                                 w(4)*A1(x,y-1)   w(5)*A1(x,y)   w(6)*A1(x,y+1) ...
                                 w(7)*A1(x+1,y-1) w(8)*A1(x+1,y) w(9)*A1(x+1,y+1)];
        % Perform Dilation (maximum value in the neighborhood)
        A2(x,y) = max(neighborhood_dilation);
    end
end
% Display the result after Dilation
subplot(223)
imshow(A2)
title('Image After Dilation')
```

## CODE:

Closing

```
% Program for Closing
% Closing means Dilate first and Erode later

clear all
clc

a = imread('coins.png');
a = double(a);
p = size(a);
w = [1 1 1; 1 1 1; 1 1 1]; % Structuring element

% Perform Dilation
for x = 2:p(1)-1
    for y = 2:p(2)-1
        % Extract a 3x3 neighborhood
        a1 = [w(1)*a(x-1,y-1) w(2)*a(x-1,y) w(3)*a(x-1,y+1) ...
              w(4)*a(x,y-1)   w(5)*a(x,y)   w(6)*a(x,y+1) ...
              w(7)*a(x+1,y-1) w(8)*a(x+1,y) w(9)*a(x+1,y+1)];
        % Perform Dilation (maximum value in the neighborhood)
        A1(x,y) = max(a1);
    end
end

% Display the result after Dilation
subplot(131)
imshow(uint8(a))
title('Original Image')

subplot(132)
imshow(uint8(A1))
title('Image After Dilation')

% Perform Erosion
q = size(A1);
for x = 2:q(1)-1
    for y = 2:q(2)-1
        % Extract a 3x3 neighborhood from the dilated image
        a11 = [w(1)*A1(x-1,y-1) w(2)*A1(x-1,y) w(3)*A1(x-1,y+1) ...
               w(4)*A1(x,y-1)   w(5)*A1(x,y)   w(6)*A1(x,y+1) ...
               w(7)*A1(x+1,y-1) w(8)*A1(x+1,y) w(9)*A1(x+1,y+1)];
        % Perform Erosion (minimum value in the neighborhood)
        A2(x,y) = min(a11);
    end
end

% Display the result after Erosion
subplot(133)
imshow(uint8(A2))
title('Image After Erosion')
```

## CODE:

### Thickening

```
clear all
clc

% Define a 2D array with zeros
a = zeros(9,10);
a(4:4,1:10) = 1; % Create a horizontal line at the 4th row

subplot(221)
imshow(a)
title('Original Image')

% Apply the 'thicken' morphological operation
b = bwmorph(a, 'thicken', 1);

subplot(222)
imshow(b)
title('Thicken Image')
```

### Thinning

```
% Thinning
% Get the image skeleton.
BW = imread('circles.png');

subplot(131)
imshow(BW);
title('Original Image')

% Remove interior pixels to leave an outline of the shapes.
BW_outline = bwmorph(BW, 'remove');
subplot(132)
imshow(BW_outline);
title('Image Outline')

% Get the skeleton of the image.
BW_skeleton = bwmorph(BW, 'skel', Inf);
subplot(133)
imshow(BW_skeleton);
title('Image Skeleton')
```

## CODE:

Comparison of MATLAB in built functions with code

```
clear all
clc

% Load the image
a = imread('bot.png');
p = size(a);
% Using MATLAB's built-in functions for comparison
s = strel('square', 3);
dilated_matlab = imdilate(a, s);
eroded_matlab = imerode(a, s);
% Implementing dilation and erosion manually
w = [1 1 1; 1 1 1; 1 1 1]; % Structuring element
for x = 2:1:p(1)-1
    for y = 2:1:p(2)-1
        al = [w(1) * a(x-1, y-1) w(2) * a(x-1, y) w(3) * a(x-1, y+1) ...
              w(4) * a(x, y-1)   w(5) * a(x, y)   w(6) * a(x, y+1) ...
              w(7) * a(x+1, y-1) w(8) * a(x+1, y) w(9) * a(x+1, y+1)];

        % Dilation
        A1(x, y) = max(al);

        % Erosion
        A2(x, y) = min(al);
    end
end

% Plotting
subplot(231)
imshow(a)
title('Original Image')
subplot(232)
imshow(dilated_matlab)
title('Dilation using MATLAB')
subplot(233)
imshow(eroded_matlab)
title('Erosion using MATLAB')
subplot(234)
imshow(A1)
title('Dilation using program')
subplot(235)
imshow(A2)
title('Erosion using program')
```

## CODE:

Dilation and Erosion of a 3x3 Matrix

```
clear all
clc

% Define a new 3x3 binary matrix
originalMatrix = [0 0 0;
                  0 1 0;
                  0 0 0];

structuringElement = ones(3); % Structuring element for neighborhood processing

% Initialize result matrices for dilation and erosion
dilatedMatrix = ones(size(originalMatrix)); % Initialize to all white
erodedMatrix = zeros(size(originalMatrix));

for x = 2:size(originalMatrix, 1) - 1
    for y = 2:size(originalMatrix, 2) - 1
        % Extract the neighborhood using the structuring element
        neighborhood = originalMatrix(x-1:x+1, y-1:y+1);

        % Apply dilation
        dilatedMatrix(x, y) = max(neighborhood(:));

        % Apply erosion
        erodedMatrix(x, y) = min(neighborhood(:));
    end
end

% Display original, dilation, and erosion results in a 1x3 subplot
subplot(1, 3, 1)
imshow(originalMatrix)
title('Original')

subplot(1, 3, 2)
imshow(dilatedMatrix)
title('Dilation')

subplot(1, 3, 3)
imshow(erodedMatrix)
title('Erosion')

% Display matrices
disp('Original Matrix:')
disp(originalMatrix)

disp('Dilated Matrix:')
disp(dilatedMatrix)

disp('Eroded Matrix:')
disp(erodedMatrix)
```

# OUTPUT:

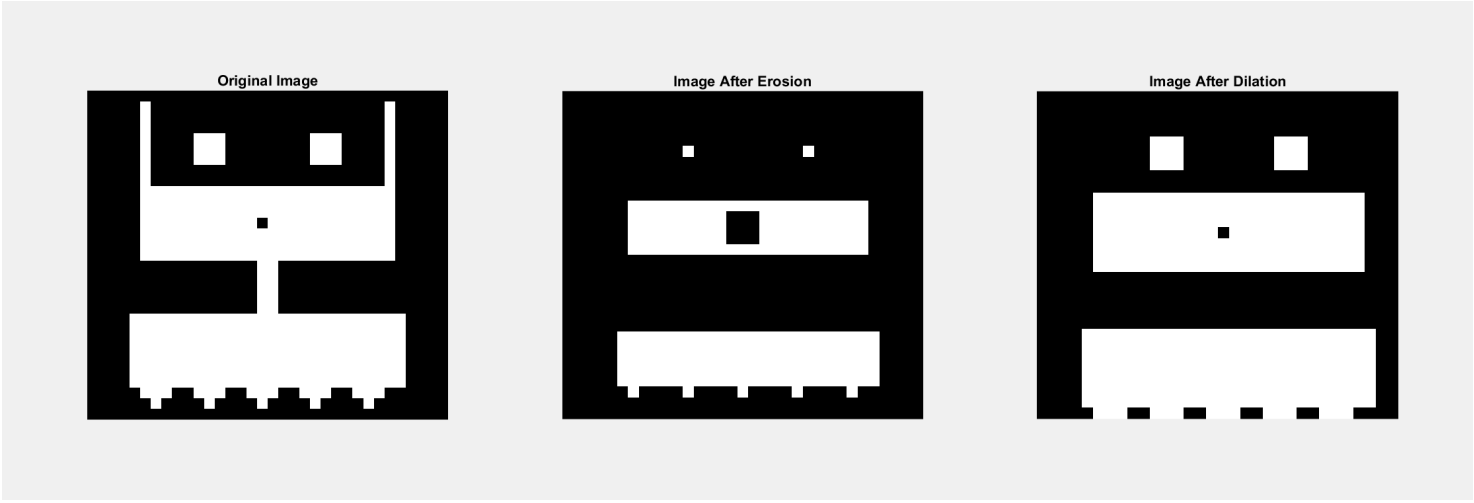


Figure 7 : MATLAB Output of Opening

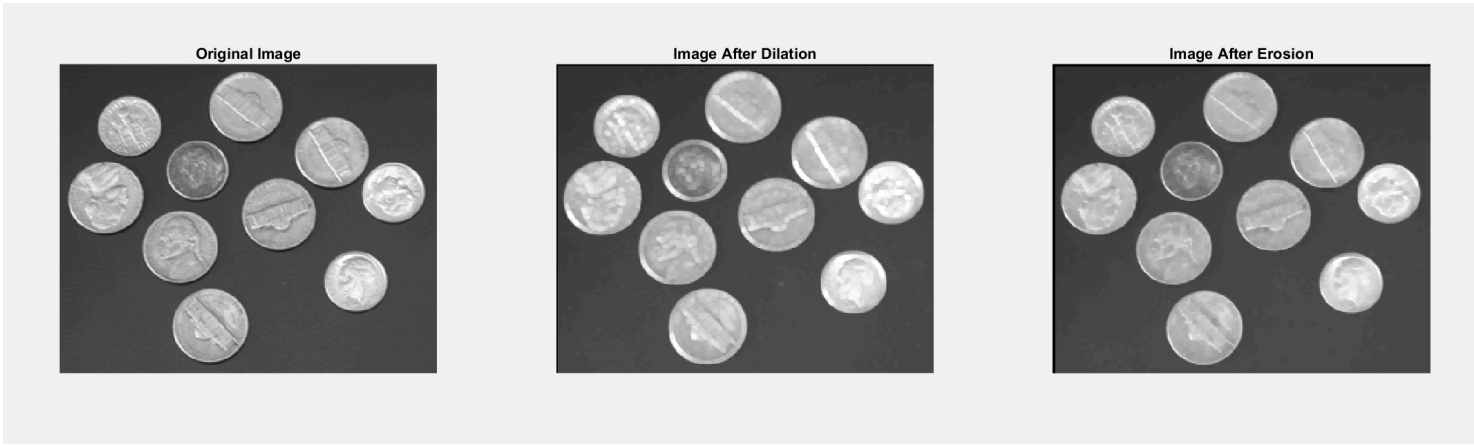


Figure 8 : MATLAB Output of Closing


Figure 9 : MATLAB Output of Thickening

# OUTPUT:

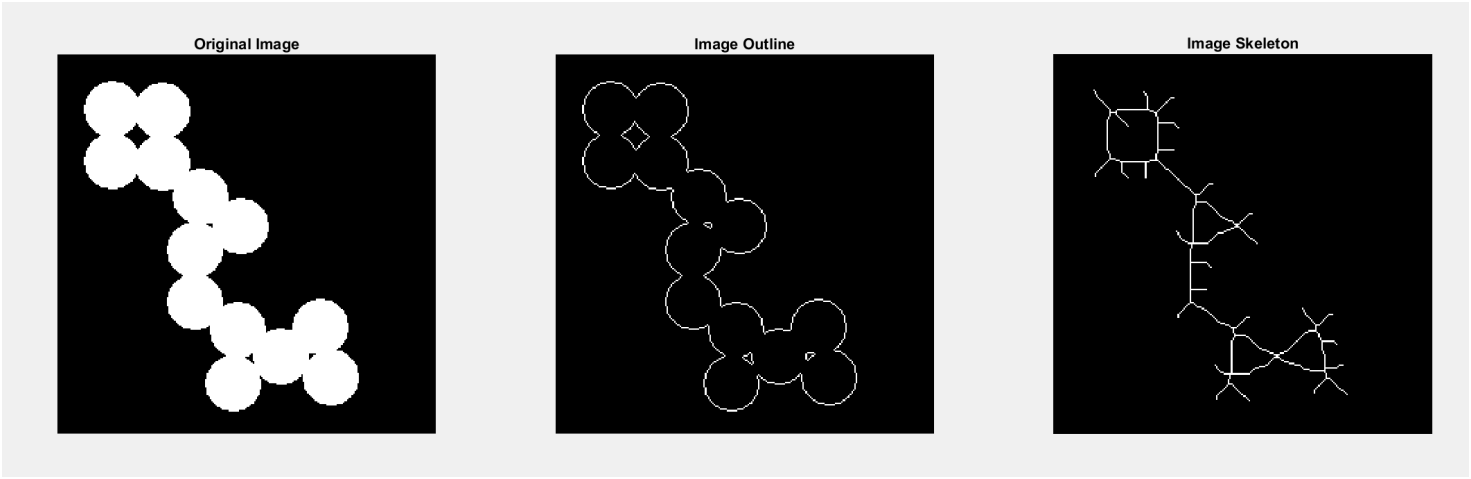


Figure 10 : MATLAB Output of Thinning

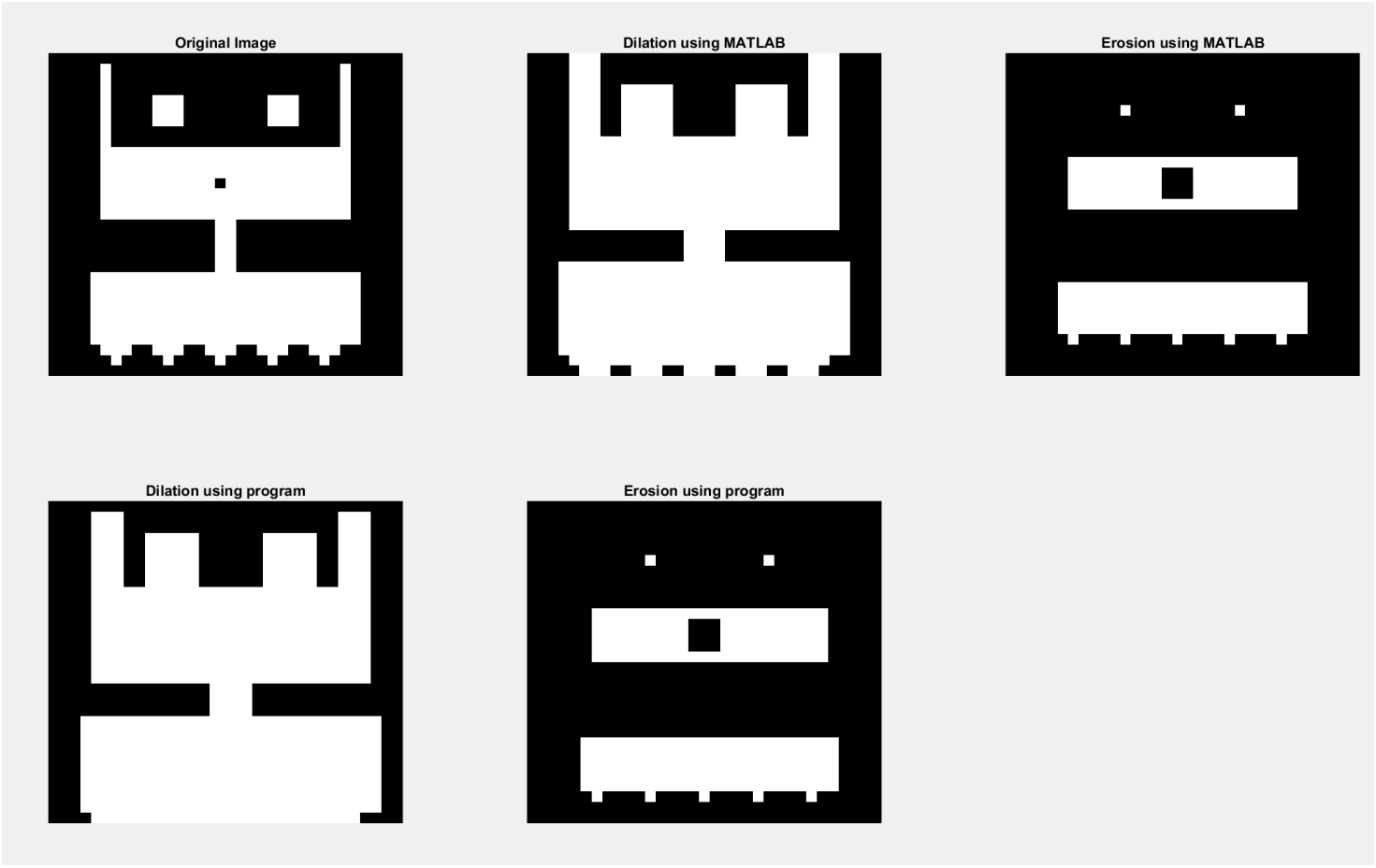


Figure 11 : MATLAB Output of Dilation and Erosion Comparison with MATLAB

# OUTPUT:

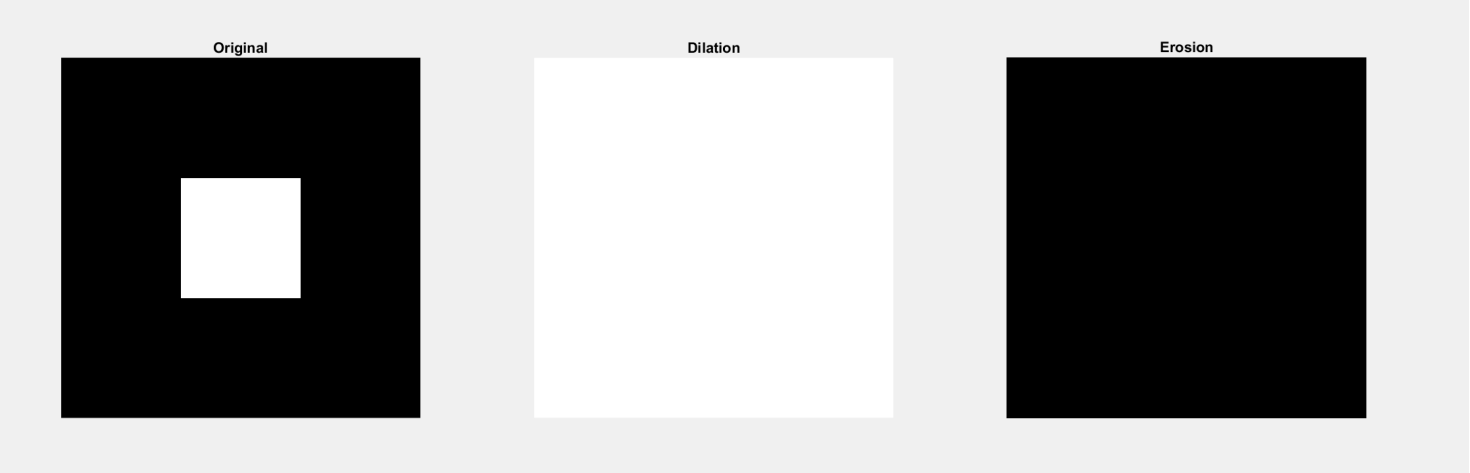


Figure 12 : MATLAB Output of Thinning

```
Command Window
  Original Matrix:
       0     0     0
       0     1     0
       0     0     0

  Dilated Matrix:
       1     1     1
       1     1     1
       1     1     1

  Eroded Matrix:
       0     0     0
       0     0     0
       0     0     0
```

Figure 11 : MATLAB Output of Command Window

## CONCLUSION:

After running various MATLAB codes for morphological operations like Closing, Opening, Thickening, and Thinning, and comparing Dilation and Erosion results with MATLAB's built-in functions, as well as performing these operations on a 3x3 matrix, several conclusions can be drawn:

1. **Closing and Opening**: These operations are effective for smoothing and refining binary images, with Closing filling gaps and Opening and removing small objects.

2. **Thickening and Thinning**: Used for shape manipulation, Thickening adds pixels to object boundaries while Thinning removes them, aiding in skeletonization.

3. **Comparison with MATLAB**: Custom implementations yielded results comparable to MATLAB's functions, affirming their correctness and MATLAB's utility for image processing tasks.

4. **Dilation and Erosion**: Fundamental morphological operations, expand or shrink object boundaries, serving key roles in noise reduction, edge detection, and feature extraction.

5. **Conclusion**: Morphological operations, with their versatility and effectiveness, are essential tools in image processing, offering tailored solutions for various tasks. MATLAB's functions provide convenience, while custom implementations offer insight into algorithms, collectively enhancing image analysis and manipulation capabilities.