Morphological Operations on a Binary Image with Custom Code

Parth Agrawal
IIT2023506

October 1, 2025

1 Introduction

Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. In this experiment, we implement and apply basic morphological operations — erosion and dilation — on a binary image containing the user's name. All operations are performed using custom MATLAB functions, without using built-in morphological functions such as `imerode` or `imdilate`.

2 Theory

2.1 Binary Morphological Operations

- **Erosion:** Erosion shrinks the foreground (white) regions of a binary image. For a pixel to remain white in the output, all pixels under the structuring element (SE) centered at that pixel must be white in the input.
- **Dilation:** Dilation expands the foreground regions. For a pixel to be white in the output, at least one pixel under the SE centered at that pixel must be white in the input.

2.2 Structuring Elements (SEs) Used

- **Linear SE:** Length 10, angle 60°. Used for directional erosion.
- **Square SE:** 4×4 matrix of ones, for isotropic erosion.
- **Custom SEs for Dilation:**

$$SE_1 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad SE_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

3 Code

The following MATLAB code implements the required operations:

```
clc; clear; close all;

% 1. Read a 256x256 grayscale image with your name from file
img = imread('first_name.png');
if size(img,3) == 3
    img = rgb2gray(img);
end
img = imresize(img, [256, 256]);
img = imbinarize(img);

figure, imshow(img); title('Input Image');

% ---- Custom Erosion and Dilation Functions ----
function out = custom_erode(I, SE)
    [m, n] = size(I);
    [se_m, se_n] = size(SE);
    pad_m = floor(se_m/2);
    pad_n = floor(se_n/2);
    Ipad = padarray(I, [pad_m pad_n], 1);
    out = false(size(I));
    for i=1:m
        for j=1:n
            region = Ipad(i:i+se_m-1, j:j+se_n-1);
            out(i,j) = all(region(SE==1)==1);
        end
    end
end

function out = custom_dilate(I, SE)
    [m, n] = size(I);
    [se_m, se_n] = size(SE);
    pad_m = floor(se_m/2);
    pad_n = floor(se_n/2);
    Ipad = padarray(I, [pad_m pad_n], 0);
    out = false(size(I));
    for i=1:m
        for j=1:n
            region = Ipad(i:i+se_m-1, j:j+se_n-1);
            out(i,j) = any(region(SE==1)==1);
        end
    end
```

```
end

% 2. Erosion: Linear structuring element (length 10, angle 60)
SE_line = zeros(10,10);
for k = 1:10
    x = k;
    y = round(1 + (k-1)*tand(60));
    if y > 0 && y <= 10
        SE_line(y,x) = 1;
    end
end
SE_line = SE_line | SE_line'; % Make it symmetric if needed

img_eroded_line = custom_erode(img, SE_line);
figure, imshow(img_eroded_line); title('Erosion: Custom Linear SE (len=10, angle=60)');

% 3. Erosion: Square structuring element (4x4)
SE_square = ones(4,4);
img_eroded_square = custom_erode(img, SE_square);
figure, imshow(img_eroded_square); title('Erosion: Custom Square SE (4x4)');

% 4. Dilation: SE1
SE1 = [0 1 1; 1 1 1; 0 1 0];
img_dilated_se1 = custom_dilate(img, SE1);
figure, imshow(img_dilated_se1); title('Dilation: Custom SE1');

% 5. Dilation: SE2
SE2 = [0 1 0; 1 0 1; 0 1 0];
img_dilated_se2 = custom_dilate(img, SE2);
figure, imshow(img_dilated_se2); title('Dilation: Custom SE2');
```

4 Results

Below are the images obtained at each step:

Figure 1: Input Image (PARTH)

Figure 2: Erosion with Linear SE (length 10, angle 60°)

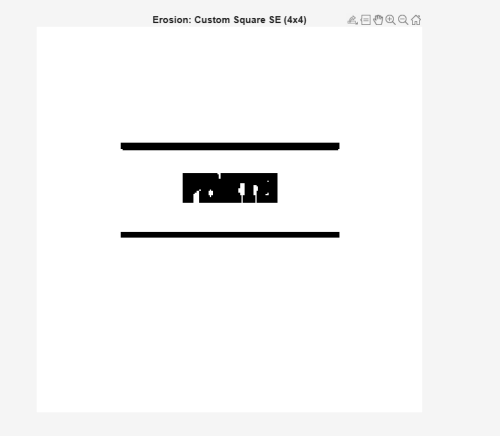

Figure 3: Erosion with Square SE (4x4)

Figure 4: Dilation with SE1

Figure 5: Dilation with SE2

5 Conclusion

This experiment demonstrates the implementation of basic morphological operations using custom code. The results verify that erosion reduces (shrinks) the foreground while dilation increases (expands) it, with the effect depending on the shape of the structuring element. The custom functions provide insight into the underlying mechanism of these fundamental image processing techniques.