Name: Vishal Rajesh Mahajan
Class: TE IT A

Q.1 Explain the concept of Erosion and Dilation in image processing.

Answer:

a. Definition:

Erosion and Dilation are fundamental morphological operations in image processing, particularly used for analyzing and processing binary or grayscale images. These operations are applied using a structuring element, which defines the neighborhood for processing each pixel.

- Erosion removes pixels on object boundaries.
- Dilation adds pixels to the boundaries of objects.

b. Concept with Example

- **Erosion:** Imagine a binary image where white pixels represent the object and black pixels represent the background. Erosion works by placing the structuring element on each white pixel. If all pixels under the structuring element are white, the central pixel remains white; otherwise, it is turned to black.
  For example, in a 3x3 square structuring element, if the neighborhood contains even one black pixel, the center white pixel is eroded (turned black).
- **Dilation:** In dilation, if at least one pixel under the structuring element is white, the central pixel becomes white. This causes white regions (objects) to grow in size.

c. Effects of Erosion and Dilation

Operation	Effect on Image
Erosion	Shrinks object boundaries; removes small white noises; breaks apart thin connections.
Dilation	Expands object boundaries; fills small black holes; connects nearby objects.

Erosion and dilation are often used sequentially (opening and closing) to achieve noise removal or object smoothing.

d. Applications of Erosion and Dilation

1. **Noise Removal:** Erosion removes small white noise; dilation removes small black noise.
2. **Object Separation:** Erosion is used to separate connected components.
3. **Hole Filling:** Dilation can be used to fill small holes within an object.

4. **Pre-processing for Feature Extraction:** Used to highlight or suppress specific structures before edge detection or object recognition.
5. **Shape Analysis:** Helps analyze object morphology and simplify structures for easier analysis.

e. Comparison between Erosion and Dilation

Aspect	Erosion	Dilation
Definition	Removes pixels from object boundaries	Adds pixels to object boundaries
Effect	Shrinks white regions	Enlarges white regions
Noise Handling	Removes small white noises	Fills small black gaps
Connectivity	May disconnect nearby objects	May connect nearby objects
Common Usage	Used before dilation in 'Opening' operation	Used after erosion in 'Closing' operation

Q2] Write a code in Python/MATLAB to illustrate the concept of Erosion and Dilation.

Answer:

Python Code

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load a color image (RGB)
image_rgb = cv2.imread('sample_rgb_image.jpg')

# Convert to Grayscale
image_gray = cv2.cvtColor(image_rgb, cv2.COLOR_BGR2GRAY)

# Convert to Binary using thresholding
_, binary_image = cv2.threshold(image_gray, 127, 255,
cv2.THRESH_BINARY)

# Define a 3x3 square structuring element
kernel = np.ones((3, 3), np.uint8)

# Apply Erosion
eroded_image = cv2.erode(binary_image, kernel, iterations=1)

# Apply Dilation
dilated_image = cv2.dilate(binary_image, kernel, iterations=1)

# Display images
titles = ['Original RGB Image', 'Grayscale Image', 'Binary
Image', 'Eroded Image', 'Dilated Image']
images = [image_rgb, image_gray, binary_image,
eroded_image, dilated_image]

plt.figure(figsize=(12, 10))
for i in range(5):
    plt.subplot(3, 2, i+1)
    if i == 0:
        plt.imshow(cv2.cvtColor(images[i],
cv2.COLOR_BGR2RGB))  # Convert BGR to RGB for
display
    else:
        plt.imshow(images[i], cmap='gray')
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

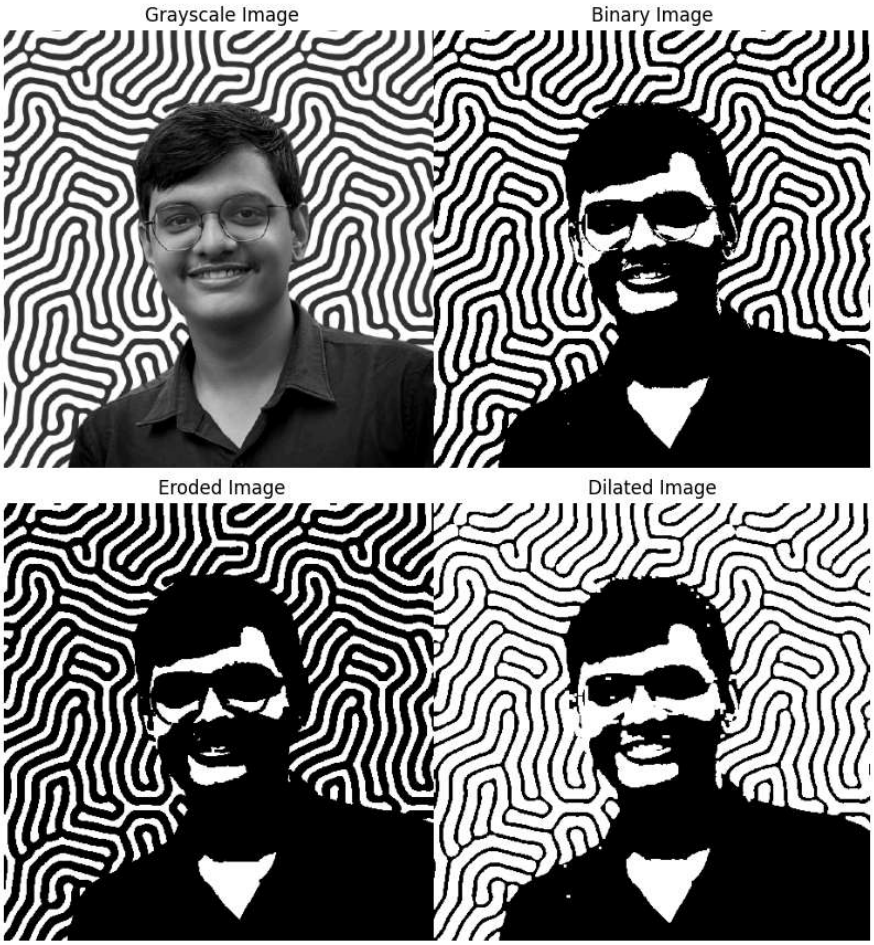

Fig1. Generated Image from Above Python Code

MATLAB CODE

```
% Read an RGB image
rgbImage = imread('sample_rgb_image.jpg');

% Convert RGB to Grayscale
grayImage = rgb2gray(rgbImage);

% Convert Grayscale to Binary using automatic thresholding
binaryImage = imbinarize(grayImage);  % You can also use: binaryImage = grayImage > 127;

% Define a 3x3 square structuring element
se = strel('square', 3);

% Apply Erosion
erodedImage = imerode(binaryImage, se);

% Apply Dilation
dilatedImage = imdilate(binaryImage, se);

% Display only the processed images
subplot(2,2,1), imshow(grayImage), title('Grayscale Image');
subplot(2,2,2), imshow(binaryImage), title('Binary Image');
subplot(2,2,3), imshow(erodedImage), title('Eroded Image');
subplot(2,2,4), imshow(dilatedImage), title('Dilated Image');
```

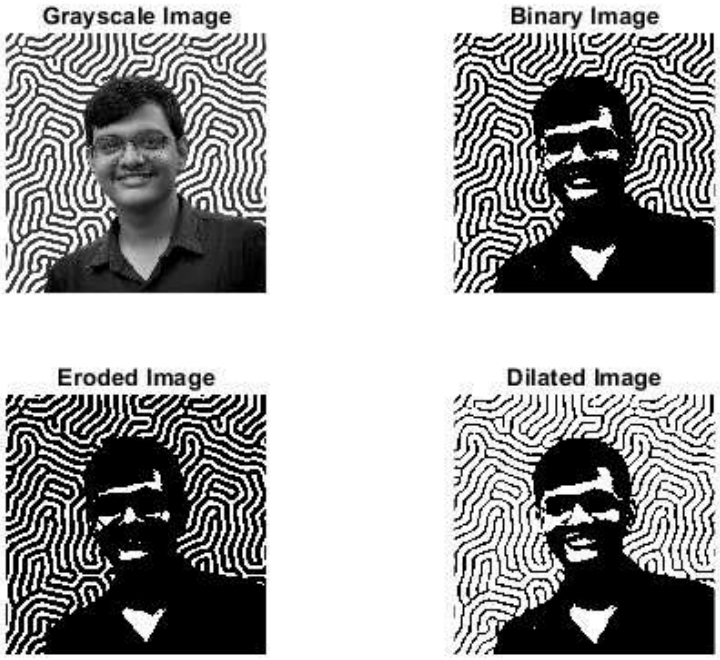

Fig2. Generated Image from Above MATLAB Code

Q3] Explain the concept of Hit and Miss Transform in image processing.

Answer:

a. Concept of Hit and Miss Transform

The **Hit and Miss Transform (HMT)** is a fundamental morphological operation used to detect specific patterns or shapes within a binary image. It is particularly useful for shape recognition, thinning, and skeletonization. Unlike dilation and erosion, which are general-purpose operators, HMT is a template-matching technique that searches for exact configurations of foreground (object) and background (non-object) pixels.

b. Mathematical Representation

The Hit and Miss Transform is defined using two structuring elements:

- One for the foreground (J)
- One for the background (K)

Let A be a binary image, then the Hit and Miss Transform is defined as:

$$A \otimes (J, K) = (A \ominus J) \cap (A^c \ominus K)$$

Where:

- $\ominus$ denotes erosion
- $\cap$ denotes intersection
- A^c is the complement of image A
- J represents the structuring element for the object
- K represents the structuring element for the background

This transform identifies locations in the image where the foreground matches J and the background matches K simultaneously.

c. Important Steps to Achieve Hit and Miss Transform

1. Prepare the binary image where the object is represented by 1s and the background by 0s.
2. Define two structuring elements (J and K):
    - J for the pattern to be matched in the foreground (object)
    - K for the pattern to be matched in the background
3. Erode the image A using structuring element J.
4. Complement the image and erode using structuring element K.
5. Intersect the two eroded images to obtain the final result.

d. Application of Hit and Miss Transform

- Pattern recognition in binary images
- Corner detection in objects
- Skeletonization of shapes
- Thinning operations to reduce binary shapes to lines
- Detection of specific configurations such as T-junctions, endpoints, and branches in images

Q4] Write a Python/MATLAB code to implement the concept of Hit and Miss Transform

Python Code

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage

# Step 1: Create a simple 5x5 binary image with a 3x3 square of 1's in the center
binary_image = np.zeros((5, 5), dtype=np.uint8)
binary_image[1:4, 1:4] = 1

# Step 2: Define structuring elements J (foreground) and K (background)
J = np.ones((3, 3), dtype=np.uint8)
K = np.zeros((3, 3), dtype=np.uint8)

# Step 3: Erode the image by J (foreground must match)
eroded_foreground = ndimage.binary_erosion(binary_image, structure=J)

# Step 4: Erode the complement by K (background must match)
eroded_background = ndimage.binary_erosion(1 - binary_image, structure=K)

# Step 5: Logical AND => Hit-and-Miss result
hit_miss_result = np.logical_and(eroded_foreground, eroded_background)

plt.figure(figsize=(9, 3))
plt.subplot(1, 3, 1)
plt.imshow(binary_image, cmap='gray')
plt.title('Binary Input')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(hit_miss_result, cmap='gray')
plt.title('Hit-and-Miss (Boolean)')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(hit_miss_result.astype(np.uint8)*255, cmap='gray')
plt.title('White Dot = Match')
plt.axis('off')

plt.tight_layout()
plt.show()
```

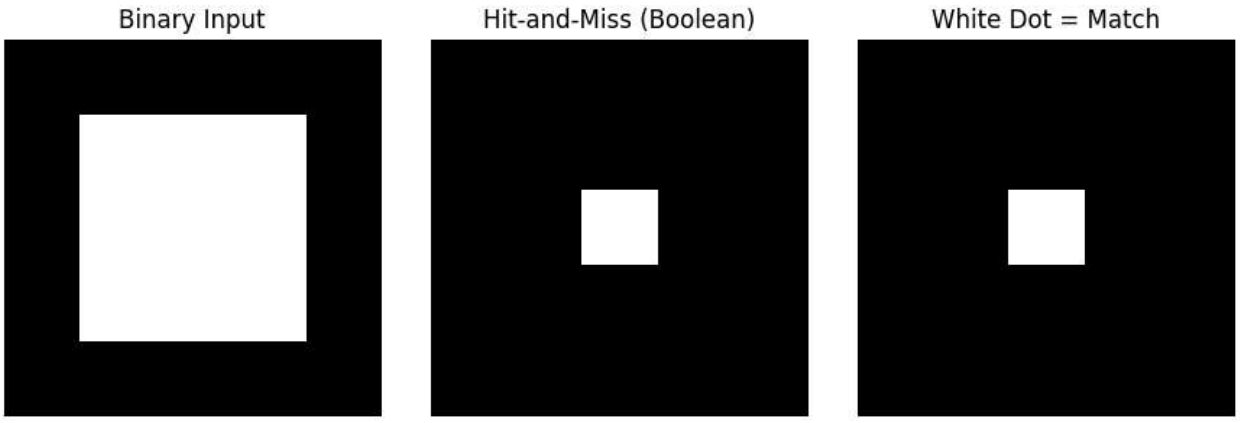

Fig3. Generated Image from Above Python Code

MATLAB CODE

```
% Step 1: Create a simple 5x5 binary image with a 3x3 white square in the center
binaryImage = zeros(5, 5);
binaryImage(2:4, 2:4) = 1;
binaryImage = logical(binaryImage);

% Step 2: Define the structuring elements:
seForeground = true(3, 3);
seBackground = false(3, 3);

% Step 3: Apply the Hit and Miss Transform
hitMissResult = bwhitmiss(binaryImage, seForeground, seBackground);

% Step 4: Display the results
figure;
subplot(1,3,1);
imshow(binaryImage);
title('Binary Input');
subplot(1,3,2);
imshow(hitMissResult);
title('Hit and Miss Output');
subplot(1,3,3);
imshow(hitMissResult, []);
title('White Dot = Match');
```

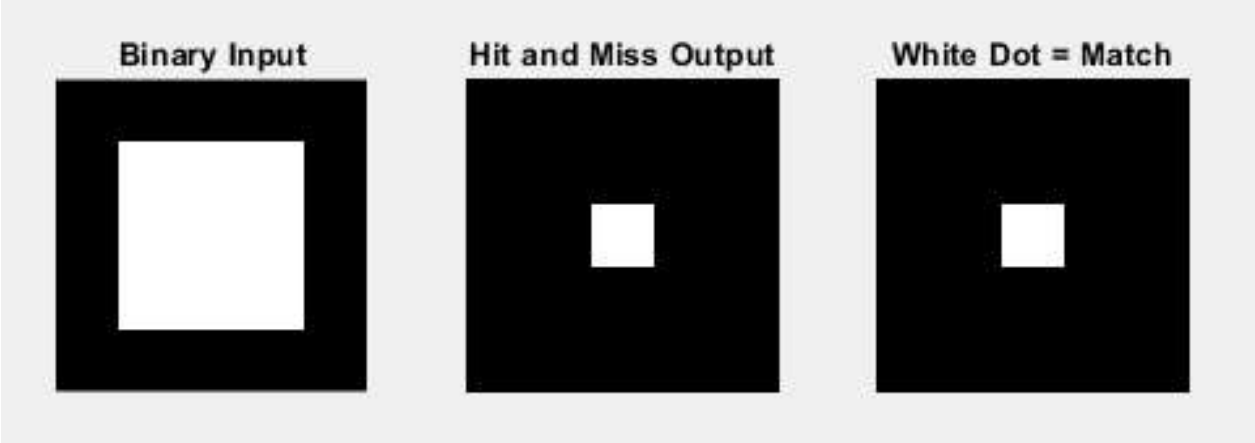

Fig4. Generated Image from Above MATLAB Code