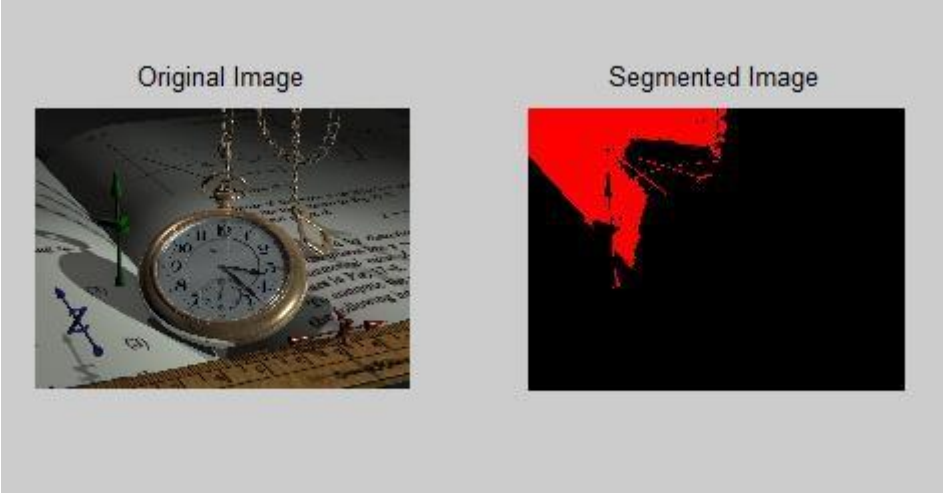
OUTPUT:



RESULT:

REVIEW QUESTIONS:

1. What is the primary objective of region-based image segmentation?
2. Describe the key steps involved in region-based segmentation.
3. What are some preprocessing techniques that can be applied to improve segmentation results?
4. How does region growing differ from split-and-merge in image segmentation?
5. What postprocessing steps can be employed to refine the segmented image?

Ex. No.	BASIC MORPHOLOGICAL OPERATIONS	Date

AIM:

The aim is to understand and implement basic morphological operations, including dilation, erosion, opening, and closing, using digital images.

SOFTWARE REQUIRED:

MATLAB 2013b

THEORY:

Morphological operations are image processing techniques based on the shape and structure of objects within an image. They are often applied to binary or grayscale images and rely on a structuring element (also known as a kernel) to modify the pixels in the image.

Dilation is used to enlarge the boundaries of objects in a binary image. It involves sliding the structuring element over the image and replacing the center pixel with the maximum value found in the neighborhood defined by the structuring element.

Erosion is used to shrink the boundaries of objects in a binary image. It involves sliding the structuring element over the image and replacing the center pixel with the minimum value found in the neighborhood defined by the structuring element.

Opening is a combination of erosion followed by dilation. It is used to remove small noise in binary images or separate objects that are close to each other.

Closing is a combination of dilation followed by erosion. It is used to close small gaps in binary images or to connect objects that are almost touching.

PROCEDURE:

1. Import the digital image onto which you want to apply morphological operations.
2. Convert the image to a binary format if it's not already binary. This is often done by thresholding.
3. Define a structuring element (a small matrix) that specifies the neighborhood for the morphological operation. The size and shape of the structuring element depend on the specific operation and the desired effect.
4. Apply dilation to the binary image using the chosen structuring element.
5. Update the pixel values according to the dilation operation.
6. Apply erosion to the binary image using the chosen structuring element.
7. Update the pixel values according to the erosion operation.
8. Apply an opening operation, which consists of an erosion followed by dilation. This is used for noise removal and object separation.
9. Apply a closing operation, which consists of a dilation followed by erosion. This is used for gap-filling and object connection.
10. Visualize or save the processed image.
11. Experiment with different structuring elements and parameters to achieve the desired image enhancement or feature extraction.
12. Summarize the results and the impact of the morphological operations on the image.

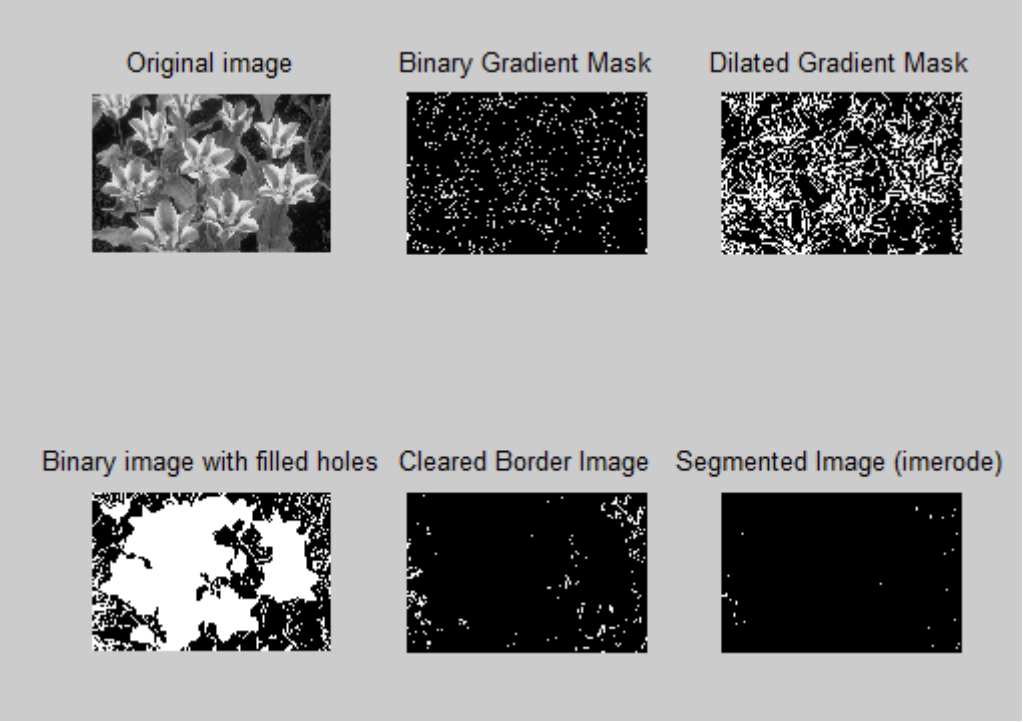
PROGRAM:

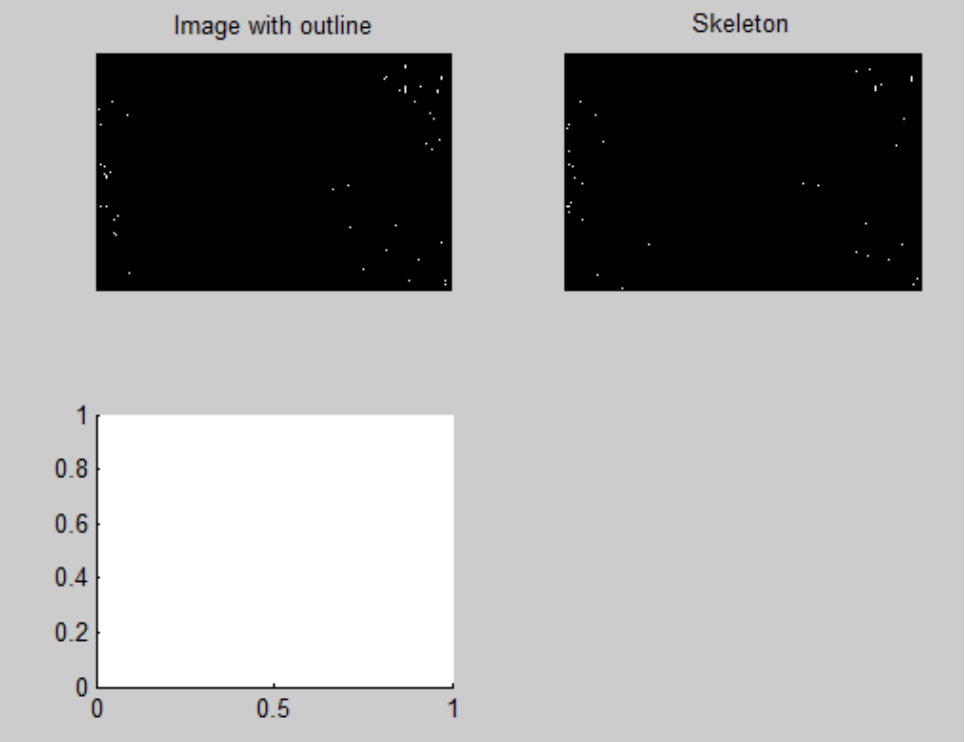
```
clc
clear all
close all
I = rgb2gray(imread('image.jpg'));
subplot(2, 3, 1)
imshow(I)
title('Original image')
%Edge detection and binary image
[~, threshold] = edge(I, 'sobel');
fudgeFactor = 0.5;
BWs = edge(I, 'sobel', threshold *fudgeFactor);
%Display the resulting binary gradient mask
subplot(2, 3, 2)
imshow(BWs)
title('Binary Gradient Mask')
%imdilate
se90 = strel('line', 3, 90);
se0 = strel('line', 3, 0);
BWsdil = imdilate(BWs, [se90, se0]);
subplot(2, 3, 3)
imshow(BWsdil)
title('Dilated Gradient Mask')
```

```
BWdfill = imfill(BWsdil, 'holes');
subplot(2, 3, 4)
imshow(BWdfill)
title('Binary image with filled holes')
BWnobord = imclearborder(BWdfill, 4);
subplot(2, 3, 5)
imshow(BWnobord)
title('Cleared Border Image')
seD = strel('diamond', 1);
BWfinal = imerode(BWnobord, seD);
BWfinal = imerode(BWfinal, seD);
subplot(2, 3, 6)
imshow(BWfinal)
title('Segmented Image (imerode)')
%Remove inner elements
BW2 = bwmorph(BWfinal, 'remove');
figure;
subplot(2, 2, 1)
imshow(BW2)
title('Image with outline')
%Skeleton image
BW3 = bwmorph(BWfinal, 'skel', Inf);
subplot(2, 2, 2)
imshow(BW3)
title('Skeleton')
subplot(2, 2, 3) imshow(labeloverlay(I,
BWfinal)) title('Mask over original image')
BWoutline = bwperim(BWfinal); Segout = I;
Segout(BWoutline) = 255;
subplot(2, 2, 4)
imshow(Segout)
title('Outlined original image')
%Opening and closing
originalBW = I; figure;
subplot(2, 2, 1)
imshow(originalBW)
title('Original')
se = strel('disk', 10);
closeBW = imclose(originalBW, se);
subplot(2, 2, 2)
imshow(closeBW, [])
title('imclose')
original = I;
subplot(2, 2, 3)
imshow(original)
title('Original')
se = strel('disk', 20);
afterOpening = imopen(original, se);
subplot(2, 2, 4)
```

```
imshow(afterOpening, [])
title('imopen')
```

OUTPUT:





RESULT:

REVIEW QUESTIONS:

1. What are morphological operations in image processing, and what is their primary purpose?
2. Explain the fundamental morphological operations: dilation and erosion. How do they work, and what are their typical applications?
3. What is the role of a structuring element in morphological operations? How does its size and shape affect the results?
4. Describe the differences between opening and closing operations. When would you use one over the other?
5. How can morphological operations be used for noise reduction or feature enhancement in images? Provide examples of scenarios where morphological operations are beneficial.

Ex. No.	IMPLEMENTATION OF IMAGE COMPRESSION TECHNIQUES	Date

AIM:

To perform JPEG compression using DCT to an image usingMATLAB.

SOFTWARE REQUIRED:

MATLAB 2013b

THEORY:

JPEG (Joint Photographic Experts Group) compression is a widely used image compression technique that employs the Discrete Cosine Transform (DCT) for reducing the size of digital images while maintaining reasonable image quality. JPEG compression begins with dividing the image into small blocks, typically 8x8 pixels. These blocks are then subjected to the DCT transformation, which converts spatial domain pixel values into frequency domain coefficients. After the DCT, quantization is applied to the frequency domain coefficients. Quantization reduces the precision of the coefficients by dividing them by a quantization matrix. The quantized coefficients are scanned in a zigzag pattern to group the high-frequency components together and make them more compressible. The zigzag-scanned coefficients are then run-length encoded, which replaces sequences of repeated values with a code that represents the value and the number of repetitions. This is a form of lossless compression. The encoded data is further compressed using Huffman encoding, which assigns shorter codes to more frequently occurring sequences and longer codes to less frequent sequences.

PROCEDURE:

1. Load the input image.
2. Divide the image into 8x8 pixel blocks.
3. Apply the DCT transformation to each block.
4. Quantize the DCT coefficients using a quantization matrix.
5. Perform zigzag scanning on the quantized coefficients.
6. Run-length encode the zigzag-scanned coefficients.
7. Apply Huffman encoding to further compress the data.
8. Save the compressed data and encoding tables for later decoding.
9. To decompress, reverse the process using Huffman decoding, run-length decoding, de-zigzag scanning, de-quantization, and inverse DCT transformation.
10. Display or save the decompressed image.

PROGRAM:

```
%nearer values are retained apart  from DC component
clc
clear
all
close
all
a = rgb2gray(imread('image.jpg'));
figure;
subplot(1, 2, 1)
imshow(a)
title('Original image')
[m, n] = size(a);
bs = 8;
Q = 20 * ones(bs);
Q(1, 1) = 10;
Rec_a = zeros(size(a));
for i=1:bs:(m-bs+1)
        for j=1:bs:(n-bs+1)
                block = a(i:i+bs-1, j:j+bs-1);
                block_D = dct2(block);
                block_Q = round(block_D./Q);
                d = block_Q.*Q;
                Rec_a(i:i+bs-1, j:j+bs-1) = idct2(d);
        end
end
subplot(1, 2, 2)
imshow(uint8(Rec_a))
title('Reconstructed image')
MSE = sum(sum((uint8(a)-uint8(Rec_a)).^2))/(m*n)
% set all components to zero except the DC component
clc
clear
all
close
all
a = rgb2gray(imread('D:\College\DIP\Lab\Lab8\image.jpg'));
figure;
subplot(1, 2, 1)
imshow(a)
title('Original image')
[m, n] = size(a);
bs = 8;
Q = 20 * ones(bs);
Q(1, 1) = 10;
Rec_a = zeros(size(a));
for i=1:bs:(m-bs+1)
        for j=1:bs:(n-bs+1)
                block = a(i:i+bs-1, j:j+bs-1);
```

```
                block_D = dct2(block);
                d = zeros(size(block_D));
                d(1, 1) = block_D(1, 1);
                Rec_a(i:i+bs-1, j:j+bs-1) = idct2(d);
        end
 end
subplot(1, 2, 2)
imshow(uint8(Rec_a))
title('Reconstructed image')
MSE = sum(sum((uint8(a)-uint8(Rec_a)).^2))/(m*n)
% specify how many elements are to be retained
clc
clear all
close all
a = rgb2gray(imread('D:\College\DIP\Lab\Lab8\image.jpg'));
k = input("Enter the number of values to be retained:");
figure; subplot(1,
2, 1)imshow(a)
title('Original image');
[m, n]=size(a);
bs = 8;
Rec_a = zeros(size(a));for
i=1:bs:(m-bs+1)
        for j=1:bs:(n-bs+1)
                block = a(i:i+bs-1, j:j+bs-1);
                block_D = dct2(block);
                d = zeros(size(block_D));
                d(1:k, 1:k) = block_D(1:k, 1:k);
                Rec_a(i:i+bs-1, j:j+bs-1) = idct2(d);
        end
end
subplot(1, 2, 2)
imshow(uint8(Rec_a))
title('Reconstructed image');
MSE = sum(sum((uint8(a)-uint8(Rec_a)).^2))/(m*n)
```

OUTPUT:

Fig 1.

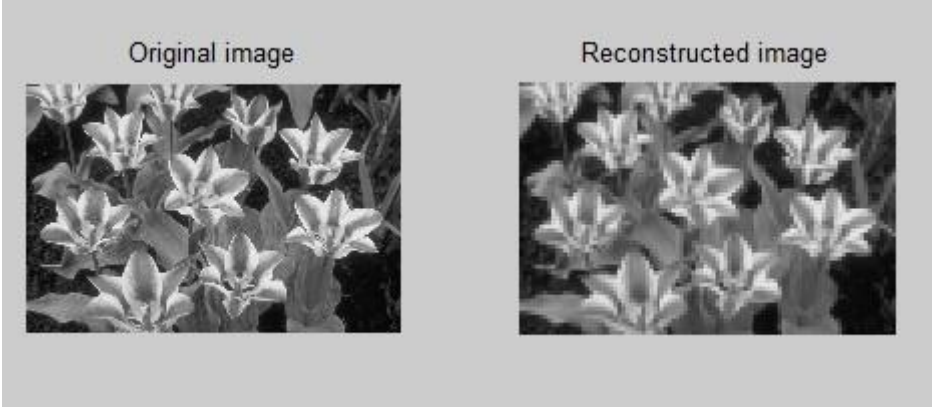


Fig.2.

Fig.3.

Enter the number of values to be retained: 5
MSE = 4.9197

RESULT:

REVIEW QUESTIONS:

1. What is the fundamental purpose of JPEG compression, and how does it achieve this purpose?
2. How does the Discrete Cosine Transform (DCT) contribute to JPEG compression?
3. Explain quantization in JPEG compression. How does the choice of quantization matrix affect image quality?
4. Describe the significance of zigzag scanning in JPEG compression. How does it impact the encoding process?
5. Why are both Run-Length Encoding (RLE) and Huffman Encoding used in JPEG compression?