REPUBLIC OF CAMEROON
PEACE-WORK-FATHERLAND
UNIVERSITY OF BUEA
BUEA, SOUTH-WEST REGION
P.O BOX 63.

RÉPUBLIQUE DU CAMEROUN
PAIX-TRAVAIL-PATRIE
UNIVERSITÉ DE BUEA
BUEA, RÉGION DU SUD-OUEST
B.P. 63.

FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING
SOFTWARE ENGINEERING
DIGITAL IMAGE PROCESSING

IMAGE FILTERING AND SEGMENTATION

Presented by:

NAMES	MATRICULE
QUINUEL TABOT NDIP-AGBOR	FE21A300
SIRRI THERESIA ANYE	FE21A306
Metagne Kamga Maiv	FE21A237
KAMCHE YANN ARNAUD	FE21A208
Nkwi Cyril Akinimbom	FE21A281

Course Facilitator:
Dr. SITAMZE BERNARD, PhD

Date Issued:
28th May 2024

Table of Content

1. INTRODUCTION

Image segmentation refers to the process of dividing an image into multiple meaningful and homogeneous regions or segments. The goal of image segmentation is to partition an image such that each resulting segment corresponds to a specific object or region of interest within the image. Image segmentation is a fundamental step in many computer vision and image processing tasks, such as object recognition, object tracking, and scene understanding.

There are various techniques used for image segmentation, including:

1. Thresholding: This technique involves setting a threshold value and classifying pixels based on their intensity values. Pixels above the threshold are assigned to one segment, while pixels below the threshold are assigned to another segment.
2. Region-based segmentation: This approach groups pixels based on similarities in color, texture, or other features. It starts with seed points and grows regions by incorporating neighboring pixels that satisfy certain similarity criteria.
3. Edge detection: This technique aims to identify boundaries or edges between different regions in an image. Edges are areas of rapid intensity transitions, and detecting them can help separate objects or regions.

Image filtering, on the other hand, involves applying various filters or operations to an image to enhance or modify its visual characteristics. Filtering can be used for tasks such as noise reduction, image enhancement, feature extraction, and image transformation.
Common types of filters used in image processing include:

1. Gaussian filter: This filter is used for smoothing or blurring an image, reducing noise, and extracting important features by convolving the image with a Gaussian function.
2. Median filter: It is used to remove noise from an image by replacing each pixel's value with the median value of its neighboring pixels.
3. Sobel filter: This filter is often used for edge detection by calculating the gradient magnitude of the image. It highlights areas of rapid intensity changes.
4. Laplacian filter: This filter is used for edge detection and enhancing the edges in an image by highlighting areas of rapid intensity changes.

2. IMPLEMENTATION STEPS

Thresholding is a popular technique used for converting an image to binary. It involves selecting a threshold value and classifying pixels based on their intensity or color values. The threshold value serves as a dividing point, where pixels with values below the threshold are assigned the value 0 (black), and pixels with values above or equal to the threshold are assigned the value 1 (white).

PROBLEM STATEMENT: L(m,n)=max(|R1(m,n)|,|R2(m,n)|,|R3(m,n)|,|R4(m,n)|).
I(m,n)>T (Discontinuity).
Apply the kernel on all pixel value, take the maximum value of the pixel value. Do this on all pixel values of the image.
Convert from 0-255 to 0-1, convert it from normal scale to zero scale and then do the image segmentation with a chosen threshold value .
Please explain to me dataily how to answer this digital image processing question. For the codes, using Matlab

The steps used in solving the problem were:

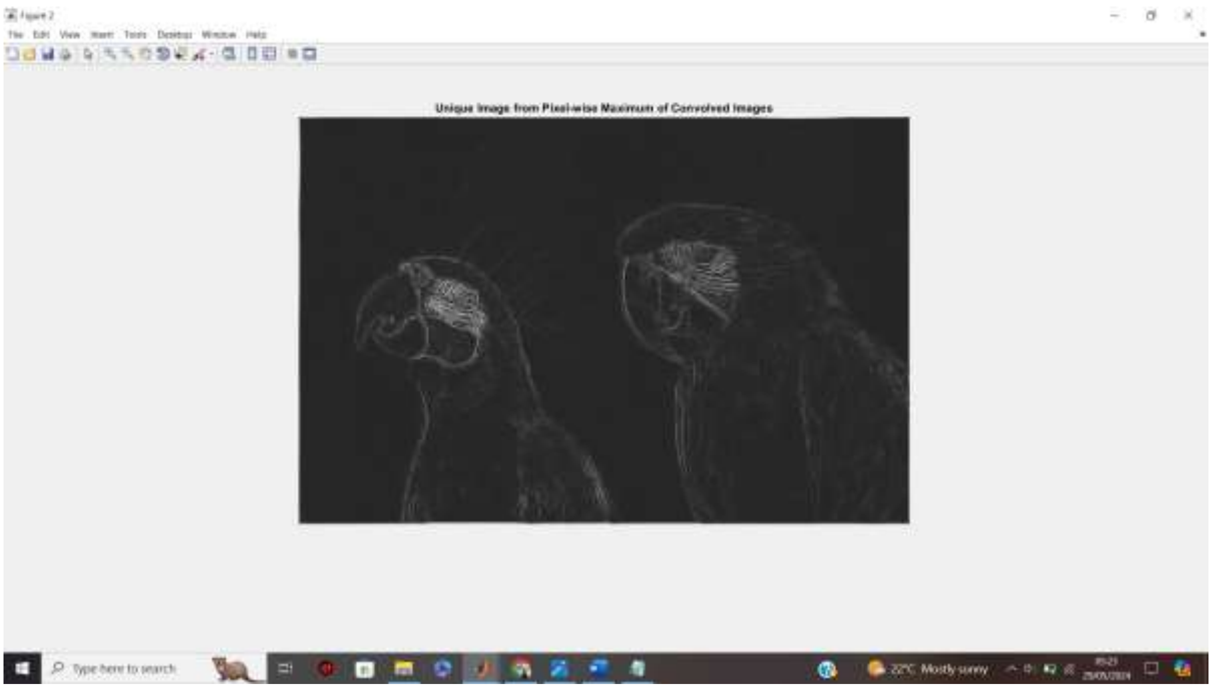
1. Read the image using the **imread** function in MATLAB.
2. Convert the pixel values from the 0-255 scale to the 0-1 scale using the double function.
3. Create four copies of the image, each representing one of the four directions, For example: R1 (right), R2 (left), R3 (down), and R4 (up).
4. Apply the kernel to each of the four copies of the image using the i**mfilter** function. The kernel should be a 3x3 matrix with the values 1, -1, and 1.
5. Take the maximum absolute value of the pixel values in each corresponding position of the four filtered images using the max function.
6. Calculate the maximum value across all four directions for each pixel position.
7. Check if the maximum value is greater than a chosen threshold value, T. If it is, then the pixel value is considered a discontinuity.
8. Convert the pixel values from the 0-1 scale to the normal scale using the uint8 function.
9. Apply the chosen threshold value to the image using the **imbinarize** function. This will convert the image to a binary image, where the pixel values above the threshold are set to 1, and the pixel values below the threshold are set to 0.

3. IMPLEMENTATION CODES

```
% Read the image
image = imread('parrots.jpeg');
grayImage = rgb2gray(image); % Convert to grayscale if
the image is in color

% Define the kernels
kernel1 = [-1, -1, -1; 2, 2, 2; -1, -1, -1];
kernel2 = [-1, 2, -1; -1, 2, -1; -1, 2, -1];
kernel3 = [-1, -1, 2; -1, 2, -1; 2, -1, -1];
kernel4 = [2, -1, -1; -1, 2, -1; -1, -1, 2];

% Apply convolution using each kernel
convolved1 = conv2(double(grayImage), kernel1, 'same');
convolved2 = conv2(double(grayImage), kernel2, 'same');
convolved3 = conv2(double(grayImage), kernel3, 'same');
convolved4 = conv2(double(grayImage), kernel4, 'same');

% Compute the pixel-wise maximum across all convolved
images
combinedConvolved = max(cat(3, convolved1, convolved2,
convolved3, convolved4), [], 3);

% Normalize the combined convolved image to the range 0
to 255
combinedConvolved = uint8(255 *
mat2gray(combinedConvolved));

% Display the unique image
figure;
imshow(combinedConvolved);
title('Unique Image from Pixel-wise Maximum of Convolved
Images');

% Optional: Apply binary thresholding
thresholdValue = 50;
binaryImage = combinedConvolved > thresholdValue;



% Display results
figure;
subplot(2, 2, 1);
imshow(grayImage);
```

```
title('Original Image');
```

4. RESULTS OF IMPLEMENTED CODES



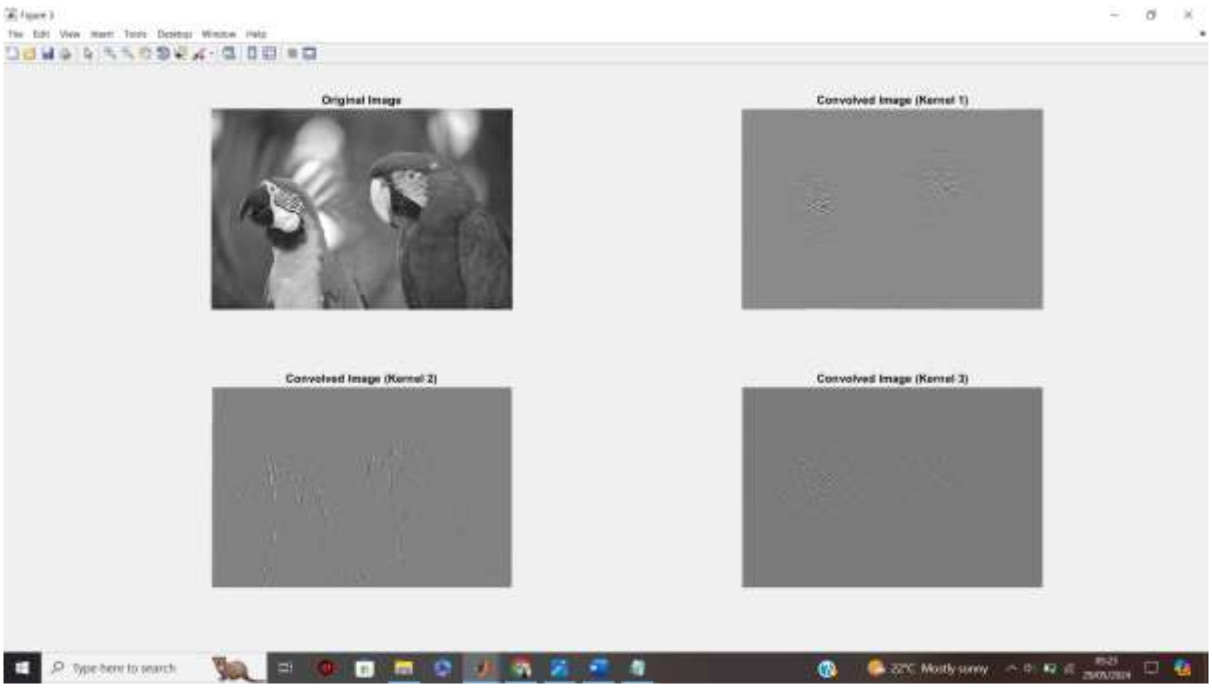


Figure 3
Original Image
Convolved Image (Kernel 1)
Convolved Image (Kernel 2)
Convolved Image (Kernel 3)
Type here to search
22°C Mostly sunny

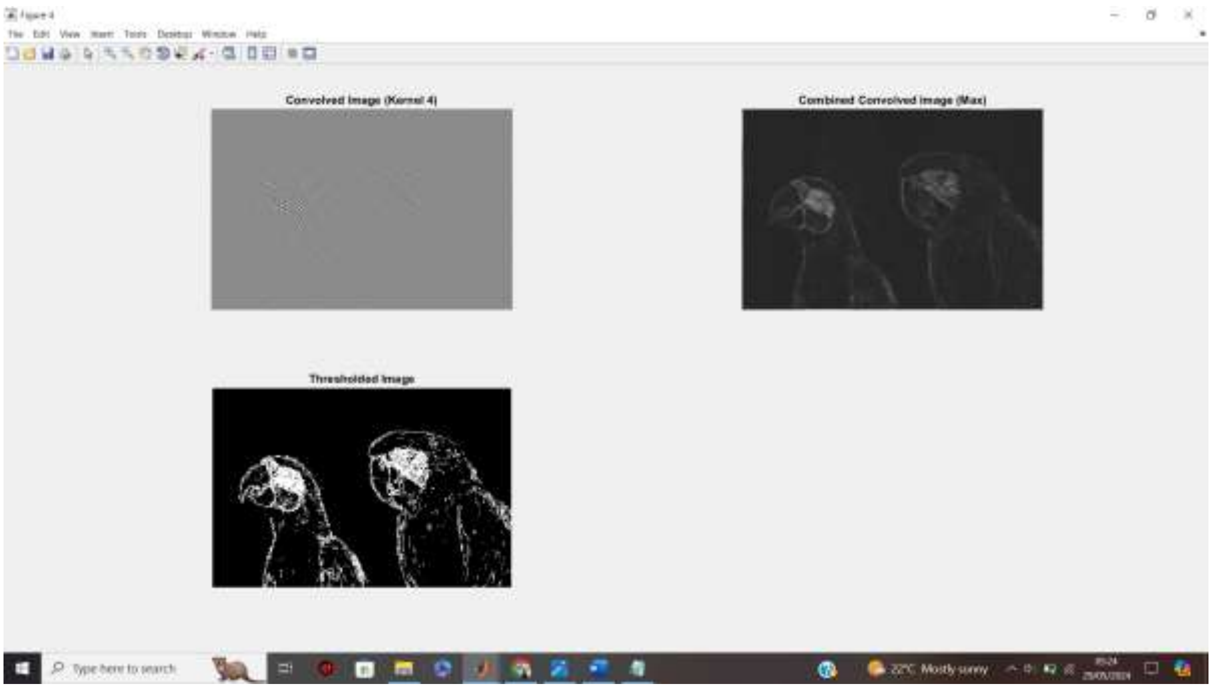


Figure 4
File Edit View Insert Tools Desktop Window Help
Convolved Image (Kernel 4)
Combined Convolved Image (Max)
Thresholded Image

5. CONCLUSION

The provided code demonstrates a series of image processing operations applied to the input image 'parrots.jpeg'. The objective is to generate a unique image based on the pixel-wise maximum of the convolved images using predefined kernels. The process involves the following steps:

The input image is read and converted to grayscale using the 'rgb2gray' function.
Four kernels, namely kernel1, kernel2, kernel3, and kernel4, are defined for convolution operations.
Convolution is applied to the grayscale image using each kernel, producing four convolved images: convolved1, convolved2, convolved3, and convolved4. The 'conv2' function is used for the convolution operation.
The pixel-wise maximum across all convolved images is computed using the 'max' function, resulting in a combined convolved image.
The combined convolved image is normalized to the range 0 to 255 using 'mat2gray' and 'uint8' functions.
The resulting unique image, obtained from the pixel-wise maximum operation, is displayed using the 'imshow' function.

Optionally, a binary thresholding step is applied to the combined convolved image using a threshold value of 50. This step creates a binary image where pixel values above the threshold are set to 1, while those below are set to 0.
The original grayscale image is also displayed for comparison.
The code snippet showcases a technique for creating a unique image by combining convolved images and demonstrates the potential of convolution and pixel-wise maximum operations in generating visually interesting results.