Enter the Column < size of matrix 2

N4:

12 5 13 4

N8:

12 5 13 4 19 10 23 7

ND:

19 10 23 7

RESULT:

REVIEW QUESTIONS:

1. What does it mean to implement relationships between pixels in image processing?
2. Give an example of an image processing task that relies on defining and using relationships between pixels.
3. How would you implement a pixel-wise contrast stretching operation in MATLAB, and why might this be useful in enhancing images?
4. What is image filtering, and why is it an essential technique in image processing?
5. Describe the process of applying a convolution filter to an image and provide an example of a filter that can be used for a specific purpose.

Ex. No.	ANALYSIS OF IMAGES WITH DIFFERENT COLOR MODELS	Date

AIM: The aim is to analyze images using different color models, such as RGB, HSL, and CMYK, to gain insights into color-based features and properties within the images.

SOFTWARE REQUIRED:

MATLAB 2013b

THEORY:

Color models are mathematical representations of colors in digital images. Common color models include RGB (Red, Green, Blue), HSL (Hue, Saturation, Lightness), and CMYK (Cyan, Magenta, Yellow, Key/Black).

In the RGB model, colors are represented by combining various intensities of red, green, and blue. It is the most common color model used in digital displays and cameras.

The HSL model focuses on human perception of colors. It represents colors based on their hue, saturation, and lightness. This model is valuable for color manipulation and image analysis.

CMYK is used primarily in color printing and subtractive color mixing. It represents colors using cyan, magenta, yellow, and key (black) components. Analyzing images in CMYK can help with color correction and print-related tasks.

PROCEDURE:

1. Load the image in MATLAB.
2. Convert the loaded image to different color models, such as RGB, HSL, and CMYK, using built-in functions or libraries. In MATLAB, you can use functions like `rgb2hsl` or `rgb2cmyk`.

3. Calculate statistical metrics for each color channel in the chosen color model. These metrics may include mean, standard deviation, or histograms of pixel values.
4. Display the image in each color model to visually compare and understand how the image's appearance changes.
5. Apply color manipulations or corrections within the chosen color model to enhance or modify specific aspects of the image, such as brightness, contrast, or color balance.
6. Extract color-based features from the different color channels, which can be used for further analysis or object recognition.
7. Perform image segmentation using color information from the selected color model to separate objects or regions of interest in the image based on color.
8. Display the results of your analysis, such as histograms of color channels, segmented regions, or color-corrected images.

PROGRAM:

```
% Load an image
originalImage = imread('image.jpg'); % Replace 'image.jpg' with the image filename

% Display the original image
figure;
subplot(2, 2, 1);
imshow(originalImage);
title('Original Image');

% Convert the image to RGB
rgbImage = originalImage;

% Display the RGB image
subplot(2, 2, 2);
imshow(rgbImage);
title('RGB Image');

% Conversion to HSL (Custom function required)
% hslImage = rgb2hsl_custom(rgbImage);

% Display the HSL image
% subplot(2, 2, 3);
% imshow(hsl2rgb_custom(hslImage));
% title('HSL Image');

% Conversion to CMYK (Custom function required)
% cmykImage = rgb2cmyk_custom(rgbImage);

% Display the CMYK image
% subplot(2, 2, 4);
% imshow(cmyk2rgb_custom(cmykImage));
% title('CMYK Image');

% Analyze color-based features
```

```
% Dominant colors in RGB
rgbDominantColor = calculateDominantColor(rgbImage);
fprintf('Dominant color in RGB: R=%d, G=%d, B=%d\n', rgbDominantColor(1),
rgbDominantColor(2), rgbDominantColor(3));

% % Dominant colors in HSL (Custom function required)
% hslDominantColor = calculateDominantColor(hsl2rgb_custom(hslImage));
% fprintf('Dominant color in HSL: R=%d, G=%d, B=%d\n', hslDominantColor(1),
hslDominantColor(2), hslDominantColor(3));

% % Dominant colors in CMYK (Custom function required)
% cmykDominantColor = calculateDominantColor(cmyk2rgb_custom(cmykImage));
% fprintf('Dominant color in CMYK: R=%d, G=%d, B=%d\n', cmykDominantColor(1),
cmykDominantColor(2), cmykDominantColor(3));

% Calculate and plot color histograms
rgbHistogram = calculateColorHistogram(rgbImage);
% hslHistogram = calculateColorHistogram(hsl2rgb_custom(hslImage));
% cmykHistogram = calculateColorHistogram(cmyk2rgb_custom(cmykImage));

% Plot RGB histogram
figure;
subplot(3, 1, 1);
bar(rgbHistogram, 'r');
title('RGB Histogram');

% % Plot HSL histogram
% subplot(3, 1, 2);
% bar(hslHistogram, 'g');
% title('HSL Histogram');

% % Plot CMYK histogram
% subplot(3, 1, 3);
% bar(cmykHistogram, 'b');
% title('CMYK Histogram');

% Function to calculate dominant color (implementation required)

% % Custom RGB to HSL conversion function
% function hslImage = rgb2hsl_custom(rgbImage)
%     % Implement RGB to HSL conversion here
% end
```

% % Function to calculate dominant color

```
function dominantColor = calculateDominantColor(image)
    % Convert the image to double precision
    doubleImage = im2double(image);

    % Reshape the image into a 2D array of RGB values
```

```
    [height, width, ~] = size(doubleImage);
    reshapedImage = reshape(doubleImage, height * width, 3);

    % Perform k-means clustering (change k to the desired number of clusters)
    k = 5; % You can adjust this value
    [clusterIndices, ~] = kmeans(reshapedImage, k);

    % Find the largest cluster (i.e., the dominant color)
    clusterCounts = hist(clusterIndices, 1:k);
    [~, dominantCluster] = max(clusterCounts);

    % Calculate the RGB values of the dominant color
    dominantColor = mean(reshapedImage(clusterIndices == dominantCluster, :), 1);
end
```

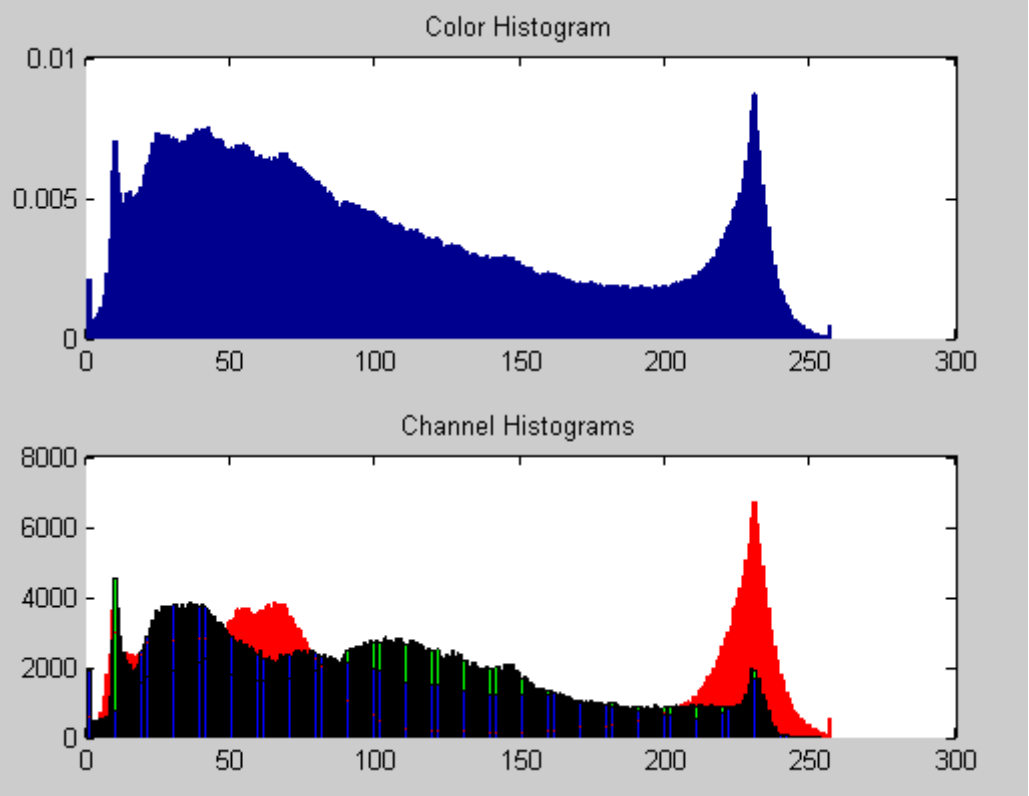
% % Function to calculate color histograms

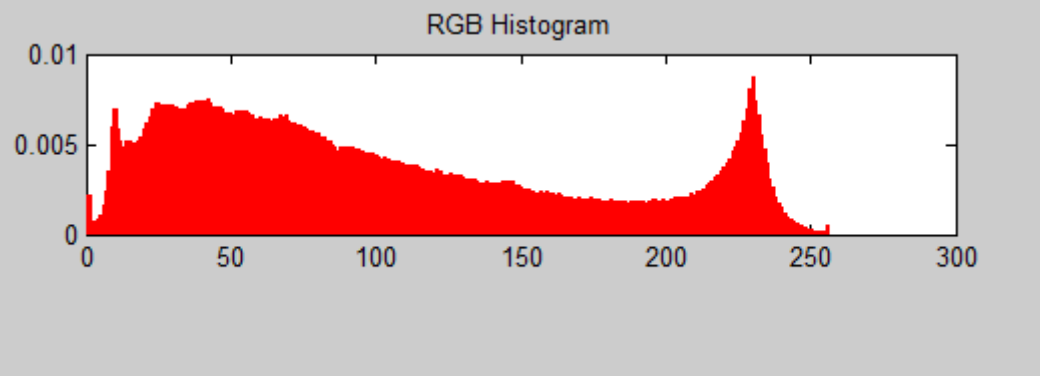
```
function colorHistogram = calculateColorHistogram(image)
    % Convert the image to double precision
    doubleImage = im2double(image);

    % Separate the RGB channels
    redChannel = doubleImage(:, :, 1);
    greenChannel = doubleImage(:, :, 2);
    blueChannel = doubleImage(:, :, 3);

    % Define the number of bins for the histogram
    numBins = 256; % You can adjust this value as needed

    % Calculate histograms for each channel
    redHist = imhist(redChannel, numBins);
    greenHist = imhist(greenChannel, numBins);
    blueHist = imhist(blueChannel, numBins);

    % Combine the individual histograms into a single histogram
    colorHistogram = [redHist, greenHist, blueHist];

    % Normalize the histogram to have values between 0 and 1
    colorHistogram = colorHistogram / sum(colorHistogram);

    % Plot the color histogram (optional)
    figure;
    subplot(2, 1, 1);
    bar(colorHistogram);
    title('Color Histogram');

    % Plot individual channel histograms (optional)
    subplot(2, 1, 2);
    bar(redHist, 'r');
    hold on;
```

```
    bar(greenHist, 'g');
    bar(blueHist, 'b');
    title('Channel Histograms');
end
```

OUTPUT:

Dominant color in RGB: R=1.290451e-01, G=1.700964e-01, B=1.570514e-01





RESULT:

REVIEW QUESTIONS:

1. What are the primary components of the RGB color model, and how are colors represented in this model?
2. Explain the significance of the HSL color model, including its components: Hue, Saturation, and Lightness.
3. How can you extract the hue channel from an image in the HSL color model using MATLAB, and what kind of image analysis tasks might benefit from this?
4. Describe the purpose and usage of the CMYK color model, particularly in the context of color printing.
5. What is the process in MATLAB for converting an image from RGB to CMYK, and how does it contribute to color correction and print-related tasks?

Ex. No.	IMPLEMENTATION OF TRANSFORMATIONS OF AN IMAGE	Date

AIM: The aim of experiment is to implement transformations on digital images, such as resizing, rotation, and flipping, to alter the appearance and structure of the image while maintaining image quality.

SOFTWARE REQUIRED:

MATLAB 2013b

THEORY:

Image transformations are operations that modify the spatial and visual properties of digital images. Common transformations include resizing, rotation, cropping, and flipping. Image resizing involves changing the dimensions of the image, either by scaling it up (enlarging) or down (shrinking). This transformation is useful for adjusting image dimensions or preparing images for different display or printing sizes. Image rotation is the process of changing the image's orientation by a specific angle, often 90, 180, or 270 degrees. It is used to correct image alignment or for artistic effects. Image flipping includes horizontal and vertical mirroring, which reverses the image along the horizontal or vertical axis. Flipping is valuable for creating mirror images or altering the image's perspective.

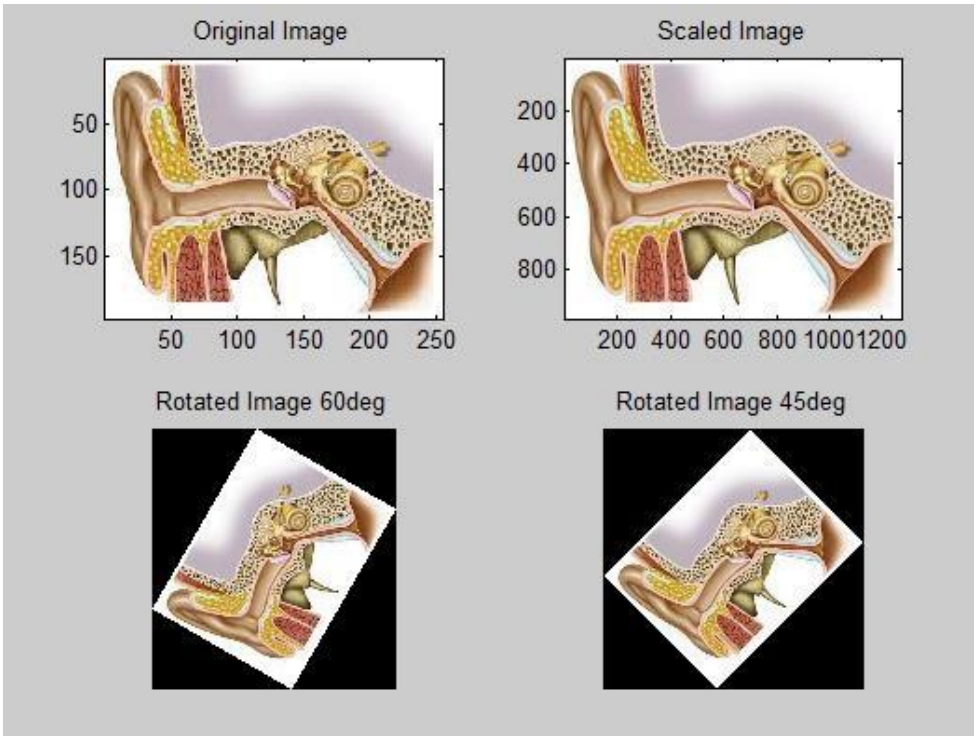
PROCEDURE:

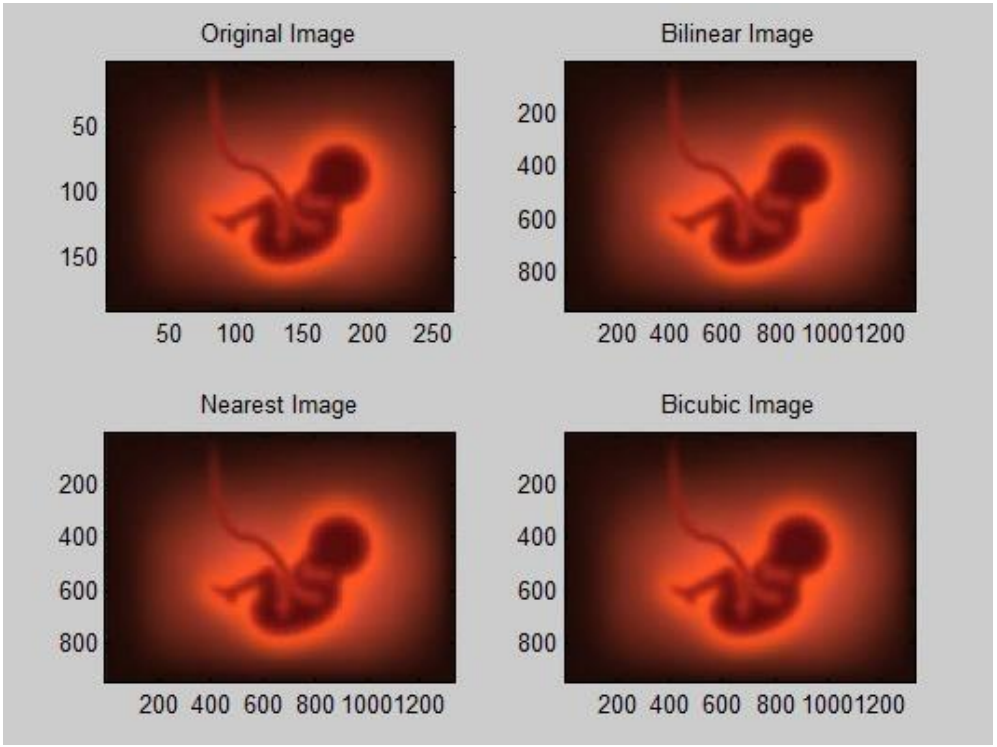
1. Load the digital image you want to transform using MATLAB.
2. Implement resizing by specifying the new dimensions or scaling factor. Interpolation methods, such as nearest-neighbor or bilinear, can be used to fill in pixel values during resizing.
3. Rotate the image by a specified angle, typically using transformation matrices. Be mindful of the interpolation method to maintain image quality during rotation.
4. Perform horizontal or vertical flipping to mirror the image. This involves rearranging pixel values along the specified axis.
5. Display the original image and the transformed image to visually compare the effects of the applied transformations.

PROGRAM:

```
%Scaling & Rotation
 % Scaling (Resize)
I=imread('earcell.jpg');
subplot(2,2,1); subimage(I); title('Original Image');
s=input('Enter Scaling Factor');
j=imresize(I,s);
subplot(2,2,2); subimage(j); title('Scaled Image');
 % Rotation
K=imrotate(j,60);
subplot(2,2,3); imshow(K); title('Rotated Image 60deg');
R=imrotate(j,45);
subplot(2,2,4); imshow(R); title('Rotated Image 45deg');
 %Display the color image and its Resized images by different methods
 %Display the color image
I=imread('embryo.jpg'); figure,
subplot(2,2,1);
subimage(I);
title('Original Image');
 %Display Resized image by Bilinear method
B=imresize(I,5);
subplot(2,2,2); subimage(B);
title('Bilinear Image');
 %Display Resized image by Nearest method
C=imresize(I,5,'nearest');
subplot(2,2,3); subimage(C);
title('Nearest Image');
 %Display Resized image by Bicubic method
D=imresize(I,5,'Bicubic');
subplot(2,2,4); subimage(D);
title('Bicubic Image');
```

OUTPUT:





RESULT: