| Ex. No. | ACQUIRE AND DISPLAY AN IMAGE, NEGATIVE OF AN IMAGE (BINARY & GRAY SCALE) | Date |
|---|---|---|
| | | |

**AIM:**

The aim is to acquire an image, convert it into its negative, and display both the original and negative images in both binary and grayscale formats using MATLAB.

**SOFTWARE REQUIRED:**

MATLAB 2013b

**THEORY:**

**Image Acquisition:** To acquire an image, you can use the `imread` function in MATLAB. This function reads an image file and stores it as a matrix. The image can be in various formats such as JPEG, PNG, or BMP.

**Negative of an Image:**

For Grayscale Images: The negative of a grayscale image can be obtained by subtracting each pixel value from the maximum intensity value. For an 8-bit image, this maximum value is 255.

For Binary Images: In binary images, where pixel values are typically 0 or 1, the negative is obtained by swapping 0s with 1s and vice versa.

**Displaying Images:** You can use the `imshow` function in MATLAB to display images. For binary images, you may want to adjust the colormap to have a clear representation of 0s and 1s.

**PROCEDURE:**

1. Image Acquisition - Read the original image with the image filename
2. Negative of an Image:
   a) Convert the original image to grayscale if needed
   b) Calculate the negative of the grayscale image
   c) For binary images, you should threshold the image first
   d) Assuming your binary image is already threshold
   e) If not, use an appropriate thresholding technique
   f) Adjust the threshold value as needed
   g) Calculate the negative of the binary image
3. Displaying Images:
   a) Display the original grayscale image
   b) Display the negative grayscale image
   c) Display the original binary image
   d) Display the negative binary image
4. Save the displayed images to files using `imwrite` if needed.

**PROGRAM:**

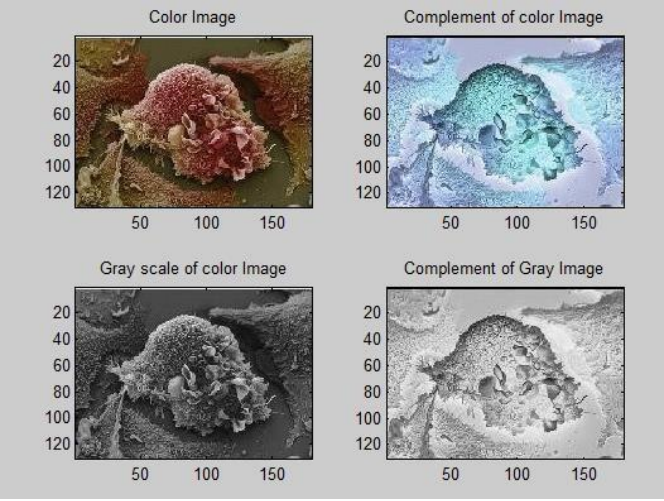
```
% Red Blue and Green and Gray Components
i=imread('cancercell.jpg');
subplot(3,2,1); imshow(i); title('Original Image');
%Red Component
r=i(:,:,1);
subplot(3,2,2); imshow(r);title('Red Component');
%Green Component
g=i(:,:,2);
subplot(3,2,3); imshow(g); title('Green Component');
```

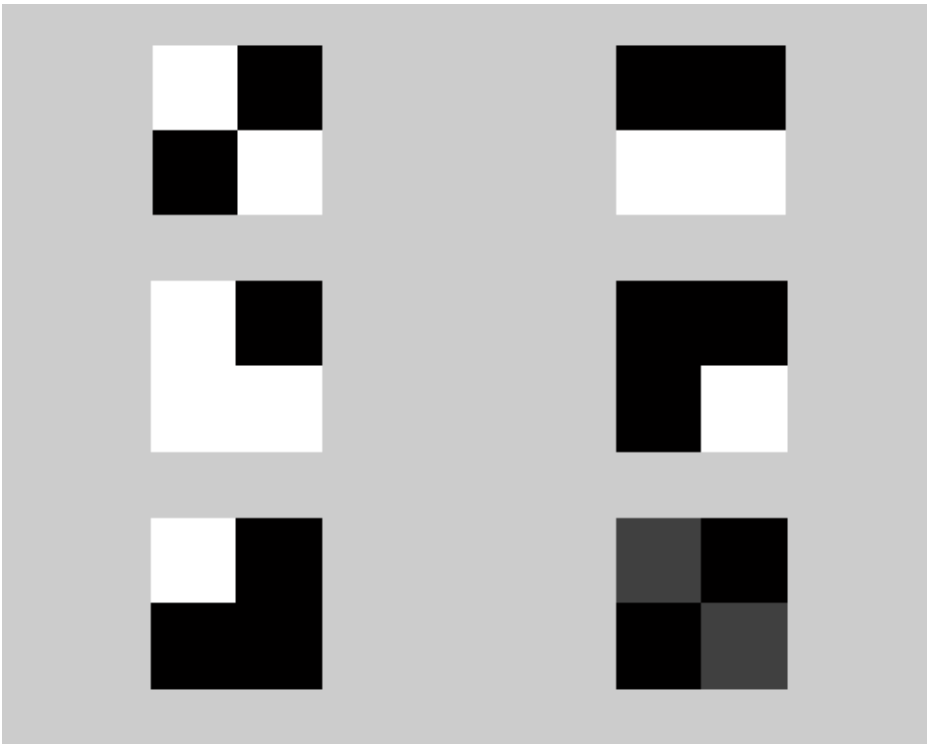
```
%Blue Component
b=i(:,:,3);
subplot(3,2,4); imshow(b); title('Blue Component');
%Color to Gray Image
rg=rgb2gray(i);
subplot(3,2,5); imshow(rg); title('Gray Image');
```

**Complement, Converting and Simulation of an Image**

```
% Display color Image, find its complement and convert to gray scale
I=imread('cancercell.jpg');
subplot(2,2,1); imshow(I); subimage(I);
title('Color Image');
c=imcomplement(I);
subplot(2,2,2); imshow(c); subimage(c);
title('Complement of color Image');
r=rgb2gray(I);
subplot(2,2,3); imshow(r); subimage(r);
title('Gray scale of color Image');
%Complement of Gray Image
b=imcomplement(r);
subplot(2,2,4); imshow(b); subimage(b);
title('Complement of Gray Image');
%Simulation of an Image (Arithmetic & Logic Operation)
a=ones(40); b=zeros(40); c=[a b;b
a]; d=[b b;a a]; A=10*(c+d);
M=c.*d;
S=c-d; D=c/4; figure;
subplot(3,2,1);  imshow(c);
subplot(3,2,2);  imshow(d);
subplot(3,2,3);  imshow(A);
subplot(3,2,4);  imshow(M);
subplot(3,2,5);  imshow(S);
subplot(3,2,6); imshow(D);
```

**OUTPUT:**





**RESULT:**

**REVIEW QUESTIONS:**

1. What MATLAB function is used to acquire an image from a file?
2. Why might it be necessary to convert a colour image to grayscale before processing it further?
3. How is the negative of a grayscale image calculated, and what is the significance of the value 255 in this context?
4. In binary image processing, how is the negative of a binary image typically obtained?
5. Can you describe a scenario where calculating the negative of an image is useful in image processing?

| Ex. No. | IMPLEMENTATION OF RELATIONSHIPS BETWEEN PIXELS | Date |
|---|---|---|
| | | |

**AIM:**

The aim is to implement relationships between pixels in a digital image.

**SOFTWARE REQUIRED:**

MATLAB 2013b

**THEORY:**

Relationships between pixels in an image refer to the interactions and dependencies among neighboring or non-neighboring pixels. These relationships are crucial for tasks such as noise reduction, feature extraction, object detection, and more. MATLAB represents an image as a matrix, with each element corresponding to a pixel. The central pixel's location is typically

denoted by its row and column coordinates in the matrix. MATLAB represents an image as a matrix, with each element corresponding to a pixel. To find a pixel's neighbors, we can specify a neighborhood size, such as a square or circular region, centered on the pixel of interest. MATLAB's array indexing allows for easy extraction of the neighboring pixels.

**PROCEDURE:**

1. Load the image into MATLAB.
2. Use the `imread` function in MATLAB to read the image.
3. Determine the specific relationship or operation you want to implement.
4. Implement the chosen relationship between pixels. Calculate the new pixel values based on a mapping function that stretches the intensity values.
5. Create a kernel or mask for the specific filtering operation (e.g., Gaussian, Sobel, or custom filters). Use convolution to apply the filter to the image.
6. For segmentation tasks, choose an appropriate method such as thresholding, region-growing, or edge-based segmentation.
7. Implement the chosen method to divide the image into distinct regions or objects.
8. Display the original image alongside the processed image to visualize the impact of the implemented relationship or operation.
9. Testing and Evaluation: Test the implemented relationship or operation on different images and evaluate its effectiveness for the intended image processing task.

**PROGRAM:**

```
% To find Neighbour of a given Pixel
a=magic(5);
disp('a='); disp(a);
b=input('Enter the row < size of the Matrix');
c=input(' Enter the Column < size of matrix');
disp('Element'); disp(a(b,c));
% 4 Point Neighbour
N4=[a(b+1,c), a(b-1,c), a(b,c+1), a(b,c-1)];
disp('N4='); disp(N4);
%8 Point Neighbour
N8=[a(b+1,c), a(b-1,c), a(b,c+1), a(b,c-1), a(b+1,c+1), a(b+1,c-1), a(b-1,c-1), a(b-1,c+1)];
disp('N8='); disp(N8);
%Diagonal Neighbour
ND=[ a(b+1,c+1), a(b+1,c-1), a(b-1,c-1), a(b-1,c+1)];
disp('ND='); disp(ND);
```

**OUTPUT:**

```
>> pixel
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

Enter the row < size of the Matrix 3
```

**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`.