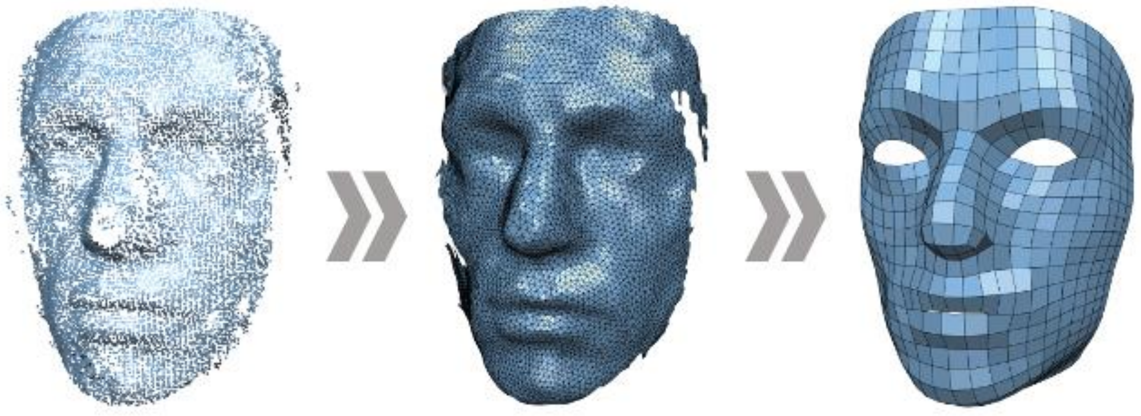
POKHARA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

APEX COLLEGE

MID-BANESHWOR, KATHMANDU



COMPUTER GRAPHICS AND IMAGE PROCESSING

REPORT: 2

SUBMITTED TO:

Abhinash Khanal

Bijaya Khadka

SUBMITTER: Piyush Manandhar

Signature

BCIS MAXTHON

INTRODUCTION

In this lab, we get to know about how images are changed through geometric and arithmetic operations to mirror an image as well as how we can add noise to the image on MATLAB. Similarly, we will also learn about various image types such as RGB images and binary images, along with various color spaces. We also use geometric functions to perform operations on images like flipping images.

OBJECTIVES

- To study about Image Class and Image Types
- To study about image operations.
- To generate Mirror Image.
- To generate Flipped Image.

THEORY

1. Image Classes:

Data types to represent pixel values supported by MATLAB/Octave. Uint8 and logical are mostly used for image processing. Various types of image classes are:

- **i.** double
- **ii.** single
- **iii.** uint8
- **iv.** uint16
- **v.** uint332
- **vi.** int8
- **vii.** int16
- **viii.** int32
- **ix.** char
- **x.** logical

2. Image Types:

There are four types of images in MATLAB.

- **i.** **Grayscale Images**: A grayscale image M pixels tall and N pixels wide is represented as a matrix of double datatype of size [M X N]. Element values for e.g., demo_img(m,n) denote the pixel grayscale intensities in [0,1] with 0=black and 1=white.

ii. **RGB (TrueColor) Images:** An RGB image is represented as a three-dimensional double matrix. Each pixel has red, green, blue components along the third dimension with values in [0,1], for e.g., the color components of pixel (m,n) are demo_img (m,n,1) = red, demo_img (m,n,2) = green, demo_img (m,n,3) = blue. If each of these components has a range 0–255, this gives a total of 256*3 different possible colors.

iii. **Indexed Images:** Indexed images are represented with an index matrix of size M×N and a colormap matrix of size K×3. The image has in total K different colors. The colormap holds all colors used in the image and the index matrix represents the pixels by referring to colors in the colormap. For example, if the 18th color is violet in demo_img(18,:) = [1,0,1], then demo_img(m,n) = 18 is a violet-colored pixel.

iv. **Binary Images:** A binary image is represented by an M×N logical matrix where pixel values are 1 (true) or 0 (false).

3. <u>Color Spaces:</u>

Color space means the use of a specific color model or system that turns colors into numbers. Some of the common color spaces are:

i. NTSC
ii. YCbCr
iii. HSY
iv. CMY
v. CMYK
vi. HIS

COMMANDS/ SYNTAX

1. B=image_class(A);
2. B=im2uint8(A);

im2uint8 takes an image as input and returns an image of class uint8. If the input image is of class uint8, the output image is identical to it. If the input image is not of class uint8, im2uint8 returns the equivalent image of class uint8, rescaling or offsetting the data as necessary.

3. B=im2uint16(A);

im2uint16 takes an image as input and returns an image of class uint16. If the input image is of class uint16, the output image is identical to it. If the input image is not of class uint15, im2uint16 returns the equivalent image of class uint16, rescaling or offsetting the data as necessary.

4. B=im2double(A);

im2double takes an image as input, and returns an image of class double. If the input image is of class double, the output image is identical to it. If the input image is not double, im2double returns the equivalent image of class double, rescaling or offsetting the data as necessary.

5. B=im2single(A);

im2single takes an image as input and returns an image of class single. If the input image is of class single, the output image is identical to it. If the input image is not of class single, im2single returns the equivalent image of class single, rescaling or offsetting the data as necessary.

6. B=mat2gray(A);

B = mat2gray (A, [MIN, MAX]) converts the matrix A to the intensity image B. The returned matrix B contains values in the range 0.0 (black) to 1.0 (full intensity or white). MIN and MAX are the values in A that correspond to 0.0 and 1.0 in B.

7. B=im2bw(A);

im2bw produces binary images from indexed, intensity, or RGB images. To do this, it converts the input image to grayscale format (if it is not already an intensity image), and then uses thresholding to convert this grayscale image to binary. The output binary image BW has values of 1 (white) for all pixels in the input image with luminance greater than level and 0 (black) for all other pixels. (Note that you specify level in the range [0,1], regardless of the class of the input image.)

RUNNING THE LAB

1. Perform pixel wise flipping of the given image

```
##using pixel calculation
img=imread('demo.png');
imshow(img);
r=img(:,:,1);
g=img(:,:,2);
b=img(:,:,3);
g_img=.299*r+.587*g+.114*b;
flip=g_img(end:-1:1,end:-1:1);
subplot(1,2,1), imshow(img);
subplot(1,2,2), imshow(flip);
imwrite (flip,'image1_roll25.jpeg' );
```

```
##flipped image using inbuilt function
img=imread('demo_resized.png');
noise=100*randn(1000);
result=img+noise;
subplot(1,2,1), imshow(img);
subplot(1,2,2), imshow(result);
imwrite(result,'image3_roll25.jpeg');
```

2. List the syntax and example of Arithmetic and Geometric image operations with outputs.

Arithmetic Operations Syntax:

- **i.** imabsdiff: absolute difference of two images

  Syntax: img= imabsdiff(x,y)
- **ii.** imadd: add two images or add constant to image

  Syntax: img= imadd(x,y)
- **iii.** imcomplement: complement image

  Syntax: img= imcomplement(x)
- **iv.** imlincomb: linear combination of images

  Syntax = imlincomb(K1, A1,K2,A2,...,Kn,An)
- **v.** imsubtract: subtract one image from another or subtract constant from image

  Syntax: img= imsubtract(x,y)
- **vi.** imdivide: divide one image into another or divide image by constant.

  Syntax: img= imdivide(x,y)

Geometric Operations Syntax:

- **i.** imcrop: crop image

  Syntax: img= imcrop(x)
- **ii.** imrotate: rotate image

  Syntax: img= imrotate(x, angle)
- **iii.** imtranslate: translate image

  Syntax: img= imtranslate(x, translation)

```
##adding noise to image
img=imread('demo_resized.png');
noise=100*randn(1000);
result=img+noise;
subplot(1,2,1), imshow(img);
subplot(1,2,2), imshow(result);
imwrite(result,'image3_roll25.jpeg');
```

```
##mirrored image
img=imread('demo.png');
[row,col,l]=size(img);
for i=1:1:row
 k=1;
 for j=col:-1:1
 temp=img(i,k,:);
 result(i,k,:)=img(i,j,:);
 result(i,j,:)=temp;
 k=k+1;
 endfor
endfor
subplot(1,2,1), imshow(img),
subplot(1,2,2), imshow(uint8(result))
imwrite(result,'image4_roll25.jpeg')
```

DISCUSSION

From this lab, we come to learn that using various geometric as well as arithmetic functions we can change the look of an image. We also use the image classes in this lab.