LAB - 5

Name	Keval D Gandevia
Roll Number	CE046
ID	19CEUEG017
Subject	Image Processing

Aim: Implement the following algorithms.

Q. 1: Take 'pout.tif' image and perform Gray-level Slicing on it. Diplay images with 2, 16, 64, 128 and 256 graylevels.

- **Code:**

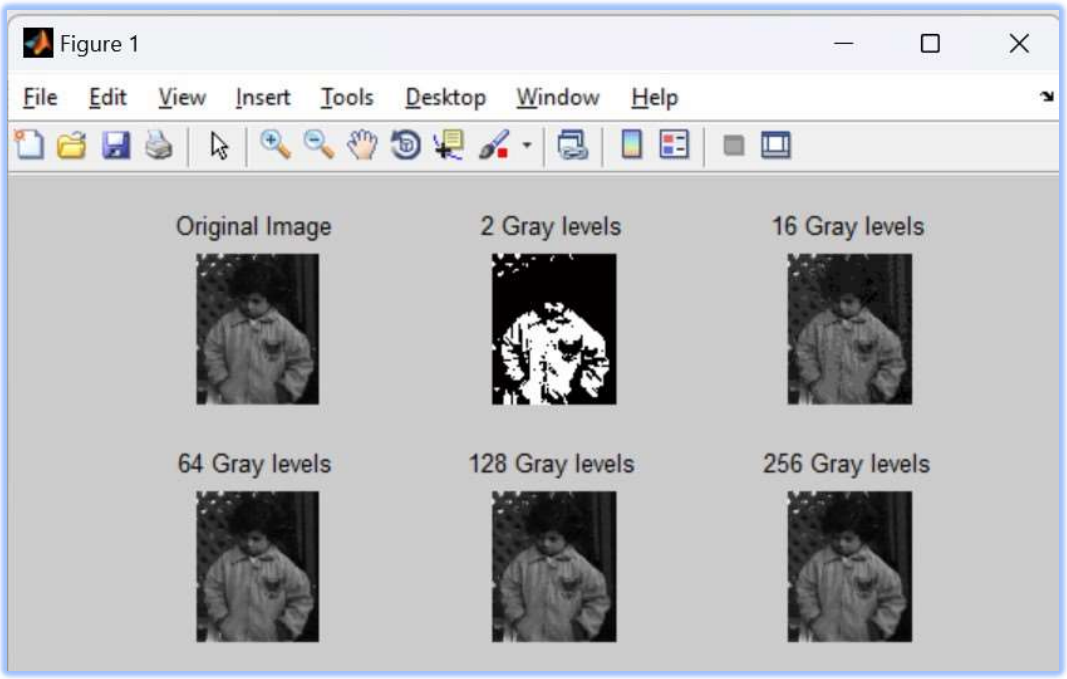
a_1.m

```
img = imread('pout.tif');
img = double(img);

subplot(3, 3, 1);
imshow(img, []);
title('Original Image');

% 2 Gray Levels
gray2img = floor(img / 128);
subplot(3, 3, 2);
imshow(gray2img, []);
title('2 Gray levels');

% 16 Gray Levels
gray16img = floor(img / 16);
subplot(3, 3, 3);
imshow(gray16img, []);
title('16 Gray levels');
```

a_1.m

```

% 64 Gray Levels
gray64img = floor(img / 4);
subplot(3, 3, 4);
imshow(gray64img, []);
title('64 Gray levels');

% 128 Gray Levels
gray128img = floor(img / 2);
subplot(3, 3, 5);
imshow(gray128img, []);
title('128 Gray levels');

% 256 Gray Levels
gray256img = floor(img / 1);
subplot(3, 3, 6);
imshow(gray256img, []);
title('256 Gray levels');
```

❖ **Output:**



Q. 2: Consider an image of 128x128 (Hint: You can resize 'cameraman.tif' to 0.5) and Implement Nearest-Neighbour Interpolation Algorithm and covert into 256x256. Don't use in-buit functions like linspace, meshgrid and interp2. Compare your result with the result obtained using the function interp2.

❖ **Code:**

a_1.m | lab_5_2.m

```
imshow(img);
title("Original Image");

c = 4;
[m, n] = size(img);
m = m * c;
d = (127 / m);

for i = 1 : m
    p(i) = 1 + (d * i) - d;
end

for i = 2 : m
    for j = 1 : m
        p(i, j) = p(1, j);
    end
end

q = p';

for i = 1 : m
    for j = 1 : m
        new_img(i, j) = img(round(q(i, j)), round(p(i, j)));
    end
end

```

```
    figure;
    imshow(new_img);
    title('Transformed Image');


    interp2Img = interp2(double(img));
    figure;
    imshow(interp2Img, []);
    title('Through interp2');

```

❖ Output:

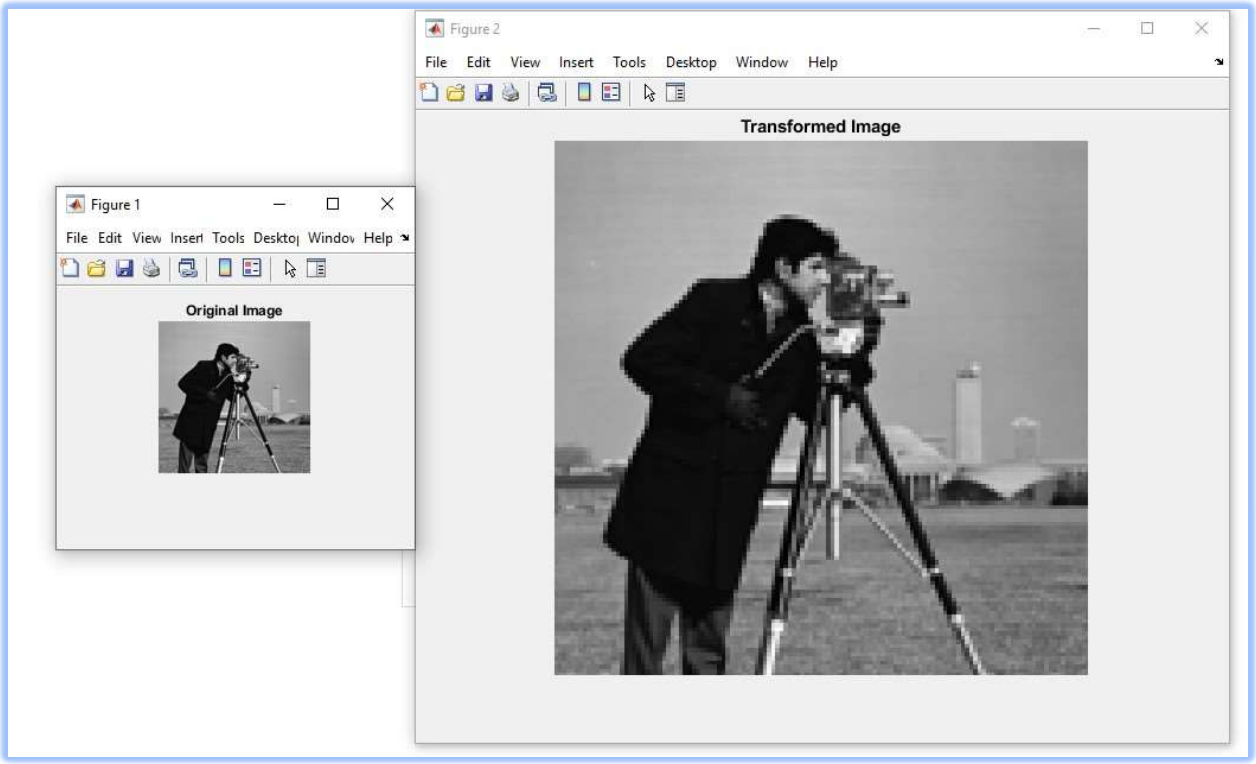


Figure 2
File Edit View Insert Tools Desktop Window Help
Transformed Image
Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image

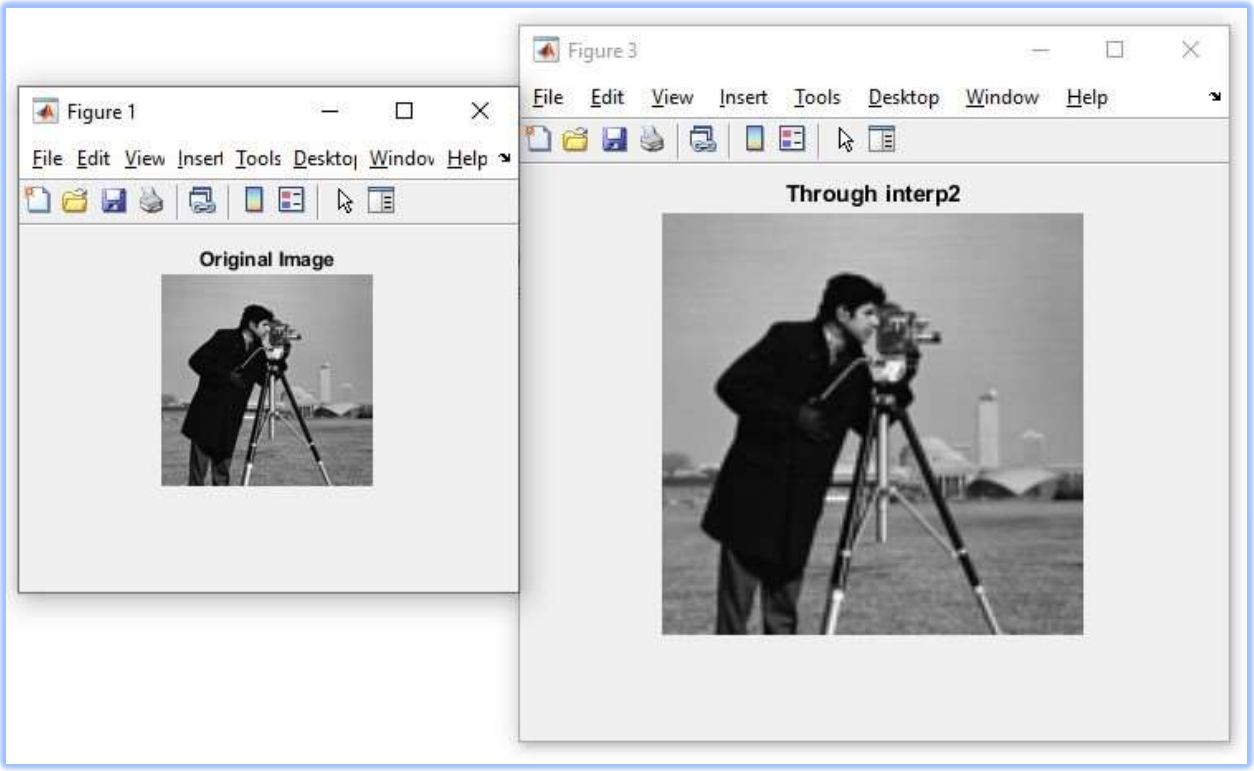


Figure 3
File Edit View Insert Tools Desktop Window Help
Through interp2
Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image

Q. 3: Take 'Cameraman.tif' image and implement Shear Transformation. A). Apply shear transformation in X-direction with value 2. B). Apply shear transformation in Y-Direction with value 3. C). Compare your result with the output generated by in-built function imtransform.

❖ **Code:**

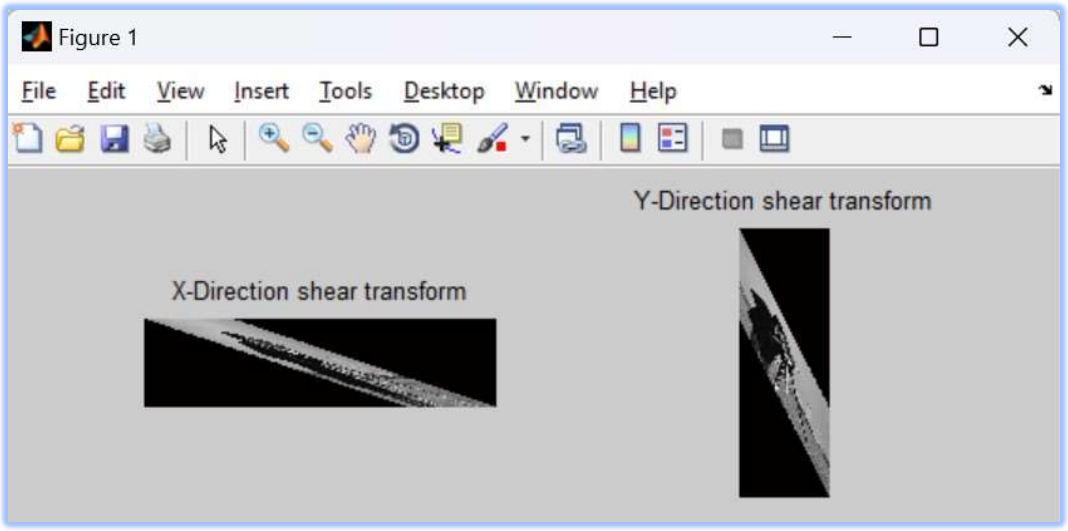
a_1.m | a_3.m

```
clear all;
img = imread('cameraman.tif');
[m, n] = size(img);

for i = 1 : m
    for j = 1 : n
        x = i;
        y = j + 3 * i;
        shear_x(x, y) = img(i, j);
    end
end

subplot(2, 2, 1);
imshow(shear_x);
title('X-Direction shear transform');
```

```
for i = 1 : m
    for j = 1 : n
        x = i + 2 * j;
        y = j;
        shear_y(x, y) = img(i, j);
    end
end

subplot(2, 2, 2);
imshow(shear_y);
title('Y-Direction shear transform');
```

```
30 -    tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);
31 -    xImg = imtransform(img, tform);
32 -    subplot(2, 2, 3);
33 -    imshow(xImg);
34 -    title('Through imtransform X-Direction');
35
36 -    tform = maketform('affine',[1 3 0; 0 1 0; 0 0 1]);
37 -    yImg = imtransform(img, tform);
38 -    subplot(2, 2, 4);
39 -    imshow(yImg);
40 -    title('Through imtransform Y-Direction');
41
```

❖ Output:



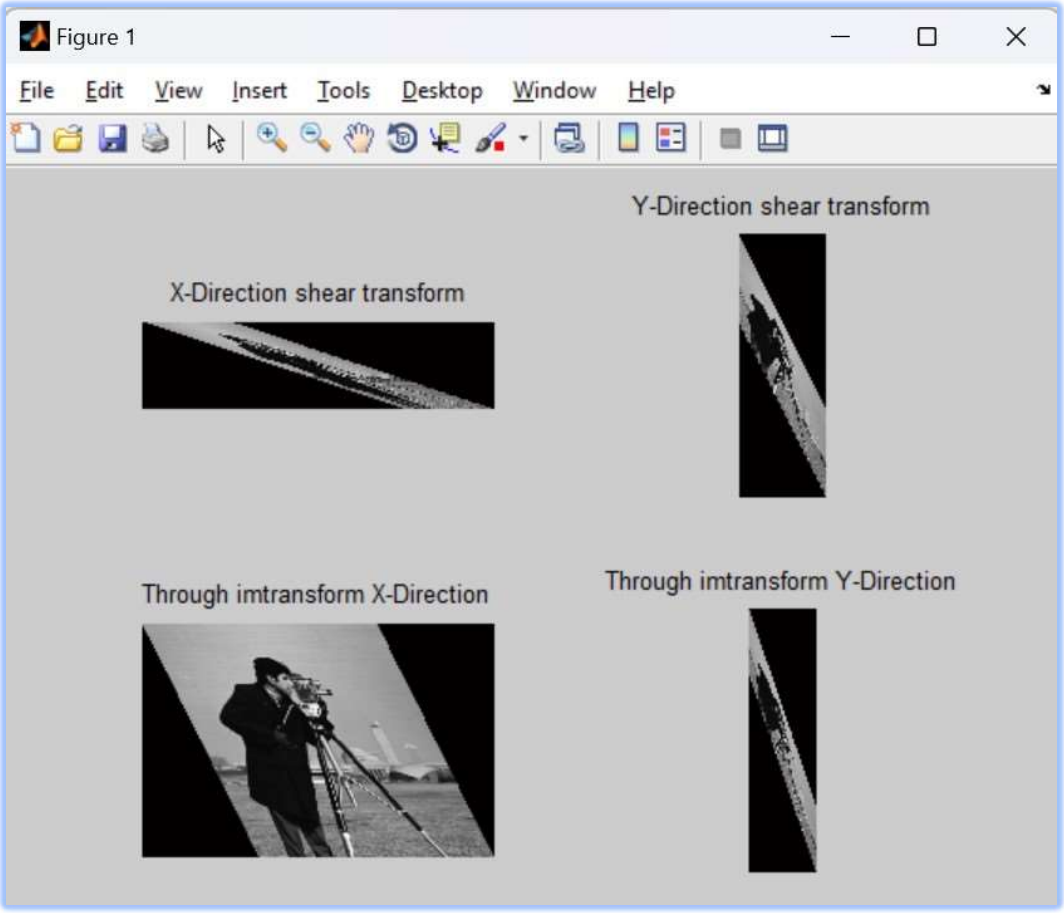


Figure 1
File Edit View Insert Tools Desktop Window Help
Y-Direction shear transform
X-Direction shear transform
Through imtransform X-Direction
Through imtransform Y-Direction

Lab – 4

Q. 4: Histogram Equalization. A). Create a function that would be able to perform histogram equalization on

a grayscale image. B). Use this function to equalize a low contrast image ex_contrast.tif (from

Lab 2). C). Use the function histeq(image) on the same image ex_contrat.tif. D). Compare the results of b) and c).

❖ **Code:**

a_4.m | histogram_equalization.m

```
img = imread('ex_contrast.tif');

subplot(2, 3, 1);
imshow(img);
title('Original Image');

equalized_image = histogram_equalization(img);
subplot(2, 3, 2);
imshow(equalized_image, []);
title('Equalized Image');

% histeq function
hist_img = histeq(img);
subplot(2, 3, 3);
imshow(hist_img, []);
title('Using histeq function');
```

a_4.m | histogram_equalization.m

```
function equalized_image = histogram_equalization(img)
    [m, n] = size(img);
    final_img = zeros(m, n);
    freq = zeros(256, 1);
    pdf = zeros(256, 1);
    cdf = zeros(256, 1);
    new_img = zeros(256, 1);

    for i = 1 : m
        for j = 1 : n
            freq(img(i, j) + 1) = freq(img(i, j) + 1) + 1;
            pdf(img(i, j) + 1) = freq(img(i, j) + 1) / (m * n);
        end
    end

    sum = 0;
```

```

    for i = 1 : size(pdf)
        sum = sum + pdf(i);
        cdf(i) = sum;
        new_img(i) = round(cdf(i) * 255);
    end

    for i = 1 : m
        for j = 1 : n
            final_img(i, j) = new_img(img(i, j) + 1);
        end
    end

    equalized_image = final_img;
end

```

❖ **Output:**

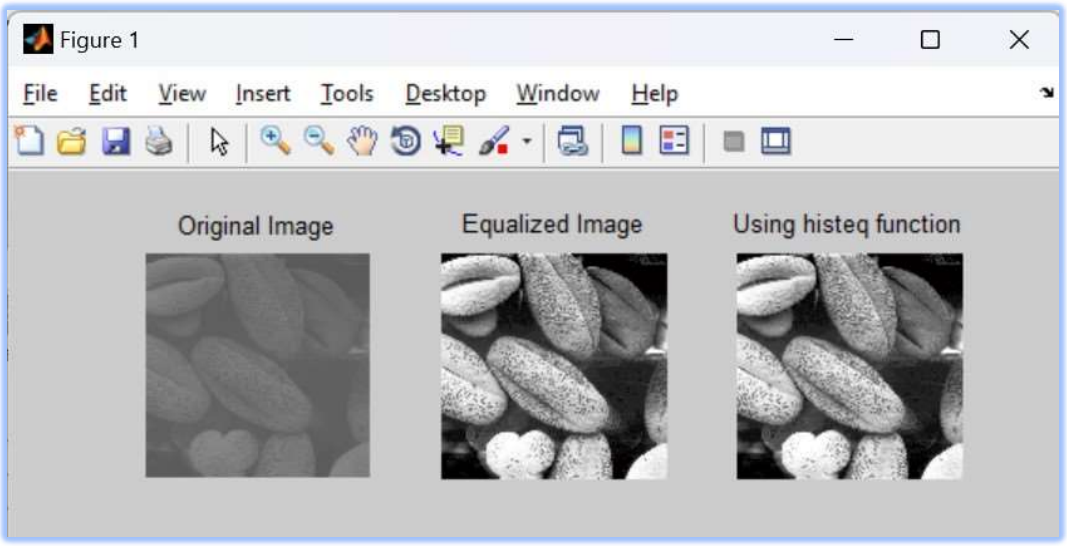


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image
Equalized Image
Using histeq function