DIGITAL IMAGE PROCESSING:PCA-2

Name: ADITYA KRISHNA Roll: 30001221016

1. Display an image to illustrate change in image quality with decreasing gray levels-128, 64, 32, 16, and 8.

Sol.

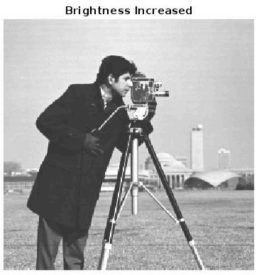
Command Window

```
>> img = imread('cameraman.tif');
figure, imshow(img), title('Original Image');
gray_img = im2gray(img);
levels = [128, 64, 32, 16, 8];
for i = 1:length(levels)
    reduced_img = uint8(floor(double(gray_img) / 256 * levels(i)));
    figure, imshow(reduced_img, [0 levels(i)-1]), title(['Gray levels: ', num2str(levels(i))]);
end
>>
>>
```

2. Write a code in Matlab to perform the following operations on an image:

• a. Increase and decrease brightness of an image.

Sol:

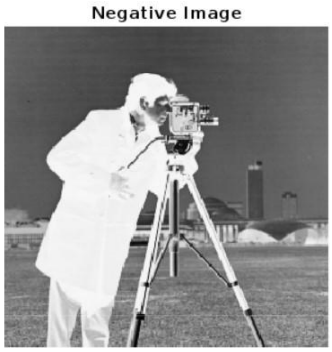



```
>> brightness_increase = 50;
brightness_decrease = -50;

bright_img = gray_img + brightness_increase;
dark_img = gray_img + brightness_decrease;

figure, imshow(bright_img), title('Brightness Increased');
figure, imshow(dark_img), title('Brightness Decreased');
>> increase = imread("/MATLAB Drive/increase.jpg");
>>
```

• b. Manipulate contrast of an image.

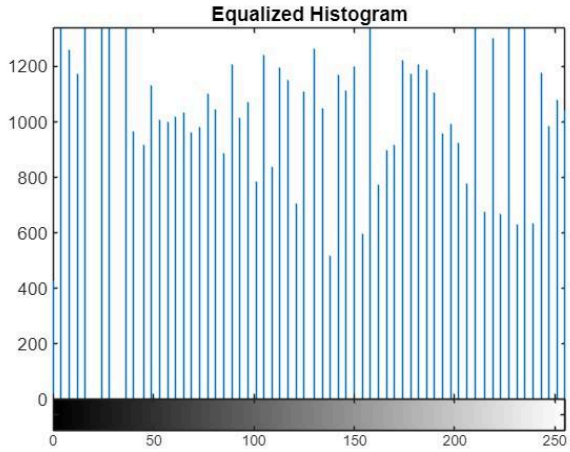

```
>> contrast_factor = 1.5; % Contrast factor
contrasted_img = imadjust(gray_img, [], [], contrast_factor);
figure, imshow(contrasted_img), title('Contrast Manipulated');
>>
```

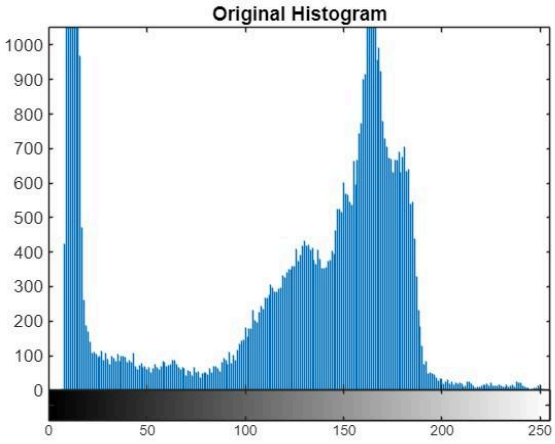
• c. Determine negative of an image




```
>> negative_img = imcomplement(gray_img);
figure, imshow(negative_img), title('Negative Image');
>>
```

3. Read an image and perform histogram equalization of the input image and analyse the result.

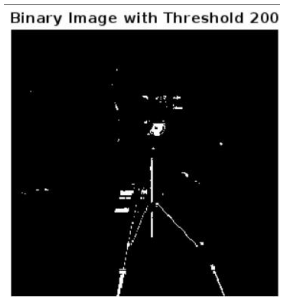




```
>> equalized_img = histeq(gray_img);
figure, imshow(equalized_img), title('Histogram Equalized Image');
figure, imhist(gray_img), title('Original Histogram');
figure, imhist(equalized_img), title('Equalized Histogram');
>>
```

4. Read a grayscale image and convert it to a binary image using hard thresholding. Make the threshold
value a user defined parameter. Vary the threshold and observe the result.

Sol:

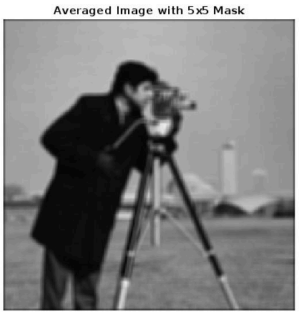


5. Read an image convolve the image with the mask $\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$

And show that it performs averaging operation which results in blurring of the image. Also analyse the impact of increasing the size of the mask to 5x5, that is, mask is

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

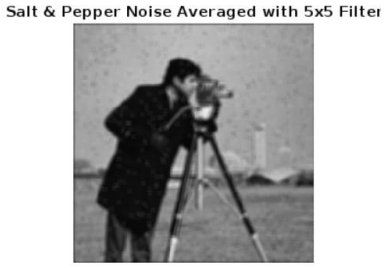
Sol:

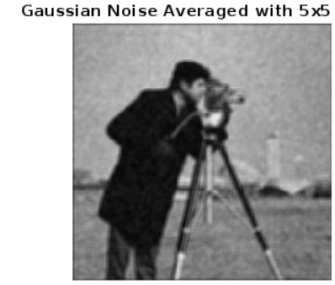



```
>> mask_3x3 = ones(3,3) / 9;
avg_img_3x3 = conv2(double(gray_img), mask_3x3, 'same');
figure, imshow(uint8(avg_img_3x3)), title('Averaged Image with 3x3 Mask');
mask_5x5 = ones(5,5) / 25;
avg_img_5x5 = conv2(double(gray_img), mask_5x5, 'same');
figure, imshow(uint8(avg_img_5x5)), title('Averaged Image with 5x5 Mask');
>>
```

6. Read an image and then corrupt the image by salt-and-pepper noise and Gaussian noise. Then apply an averaging filter of size 3 X 3 and 5 x 5 to this corrupted image. Comment on the result obtained.

Sol:

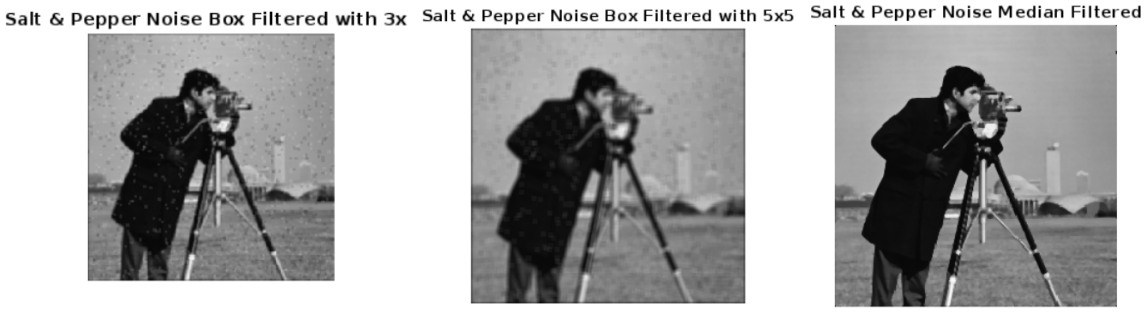




```
>> sp_noise_img = imnoise(gray_img, 'salt & pepper', 0.02);
gaussian_noise_img = imnoise(gray_img, 'gaussian', 0, 0.01);

avg_filter_3x3 = fspecial('average', [3 3]);
avg_filter_5x5 = fspecial('average', [5 5]);

avg_sp_3x3 = imfilter(sp_noise_img, avg_filter_3x3);
avg_sp_5x5 = imfilter(sp_noise_img, avg_filter_5x5);
avg_gaussian_3x3 = imfilter(gaussian_noise_img, avg_filter_3x3);
avg_gaussian_5x5 = imfilter(gaussian_noise_img, avg_filter_5x5);

figure, imshow(avg_sp_3x3), title('Salt & Pepper Noise Averaged with 3x3 Filter');
figure, imshow(avg_sp_5x5), title('Salt & Pepper Noise Averaged with 5x5 Filter');
figure, imshow(avg_gaussian_3x3), title('Gaussian Noise Averaged with 3x3 Filter');
figure, imshow(avg_gaussian_5x5), title('Gaussian Noise Averaged with 5x5 Filter');
>>
```

7. Read an image and then corrupt the image by salt-and-pepper noise. Now apply a 3 x 3 box filter, a 5 x 5 box filter and a median filter to the corrupted image and comment on the result obtained.

Sol:




```
>> box_filter_3x3 = fspecial('average', [3 3]);
box_filter_5x5 = fspecial('average', [5 5]);
box_sp_3x3 = imfilter(sp_noise_img, box_filter_3x3);
box_sp_5x5 = imfilter(sp_noise_img, box_filter_5x5);
median_sp = medfilt2(sp_noise_img, [3 3]);
% Display results
figure, imshow(box_sp_3x3), title('Salt & Pepper Noise Box Filtered with 3x3');
figure, imshow(box_sp_5x5), title('Salt & Pepper Noise Box Filtered with 5x5');
figure, imshow(median_sp), title('Salt & Pepper Noise Median Filtered');
>>
```

8. Write a matlab program that performs a two-dimensional Butterworth low-pass and high-pass filter of the given image for two different cut-off frequencies.

Sol:

Butterworth Low-pass Filtered Image

Butterworth High-pass Filtered Image

```
>> D0 = 30;
n = 2;
[rows, cols] = size(gray_img);
[u, v] = meshgrid(-floor(cols/2):floor((cols-1)/2), -floor(rows/2):floor((rows-1)/2));
D = sqrt(u.^2 + v.^2);
H_low = 1 ./ (1 + (D./D0).^(2*n));
H_high = 1 - H_low;
img_fft = fftshift(fft2(double(gray_img)));
low_filtered_img = real(ifft2(ifftshift(img_fft .* H_low)));
high_filtered_img = real(ifft2(ifftshift(img_fft .* H_high)));
figure, imshow(low_filtered_img, []), title('Butterworth Low-pass Filtered Image');
figure, imshow(high_filtered_img, []), title('Butterworth High-pass Filtered Image');
>>
```

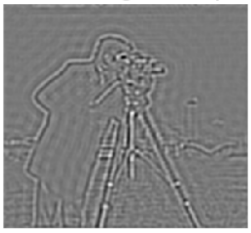
9. Read an input image to perform the following operations:

- a. High-pass filtering in the frequency domain
- b. Low-pass filtering in the frequency domain
- c. Band-pass filter in the frequency domain
- d. Band-stop filter in the frequency domain

gh-pass Filtered Image in Frequency Dom

›w-pass Filtered Image in Frequency Dom

nd-pass Filtered Image in Frequency Don

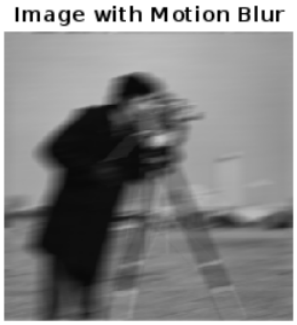


nd-stop Filtered Image in Frequency Dom

```
>> %9.a
D0 = 30;
H_high = 1 - 1 ./ (1 + (D./D0).^(2*n));

high_filtered_img = real(ifft2(ifftshift(img_fft .* H_high)));
figure, imshow(high_filtered_img, []), title('High-pass Filtered Image in Frequency Domain');

%9.b
H_low = 1 ./ (1 + (D./D0).^(2*n));

low_filtered_img = real(ifft2(ifftshift(img_fft .* H_low)));
figure, imshow(low_filtered_img, []), title('Low-pass Filtered Image in Frequency Domain');

%9.c
D1 = 20; D2 = 50;
H_bandpass = double(D >= D1 & D <= D2);

bandpass_filtered_img = real(ifft2(ifftshift(img_fft .* H_bandpass)));
figure, imshow(bandpass_filtered_img, []), title('Band-pass Filtered Image in Frequency Domain');

%9.d

H_bandstop = 1 - H_bandpass;

bandstop_filtered_img = real(ifft2(ifftshift(img_fft .* H_bandstop)));
figure, imshow(bandstop_filtered_img, []), title('Band-stop Filtered Image in Frequency Domain');
>>
```

10. Read an image and degrade the image using motion blur.

Sol:




```
>> LEN = 21;
THETA = 11;
PSF = fspecial('motion', LEN, THETA);
blurred_img = imfilter(gray_img, PSF, 'conv', 'circular');
figure, imshow(blurred_img), title('Image with Motion Blur');
>>
```