Digital Image Processing: PCA-2

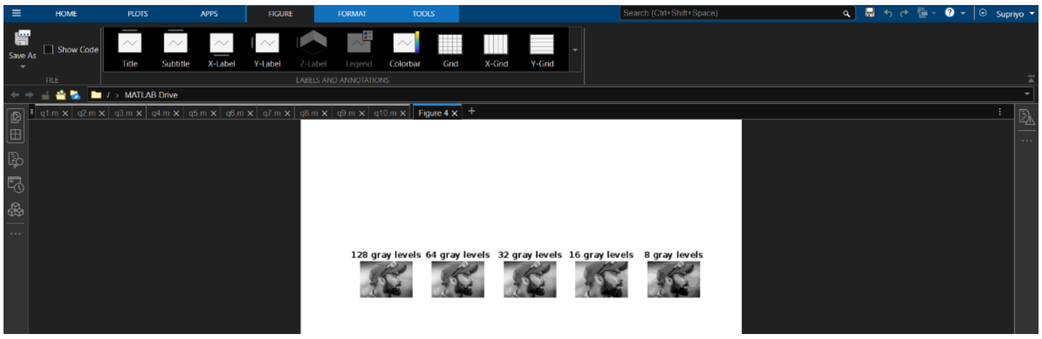
Full Marks:40 Time:90 min

Name: Supriyo Maity Roll:30001221043

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

```
img = imread('vk.jpg');
gray_img = im2gray(img);
levels = [128, 64, 32, 16, 8];
figure;
for i = 1:length(levels)
        subplot(1,5,i);
        imshow(uint8(floor(double(gray_img) / 256 * levels(i)) * (256 / levels(i))));
        title([num2str(levels(i)) ' gray levels']);
end
```

   Output:

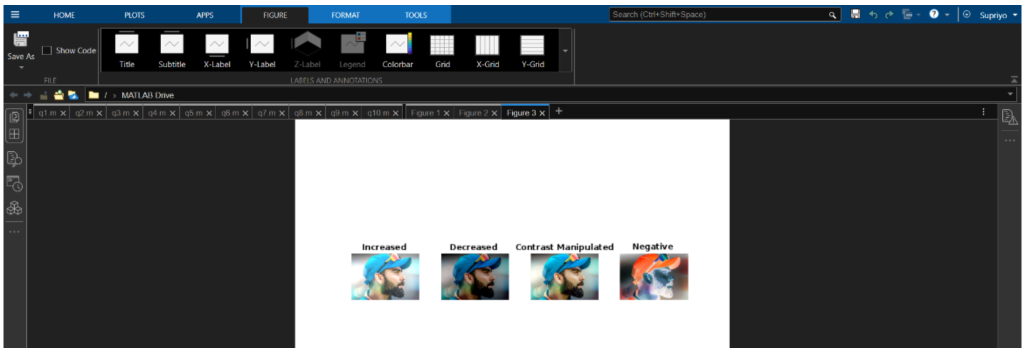


2. Write a code in Matlab to perform the following operations on an image:
   - a. Increase and decrease brightness of an image.
   - b. Manipulate contrast of an image.
   - c. Determine negative of an image.

   Code:

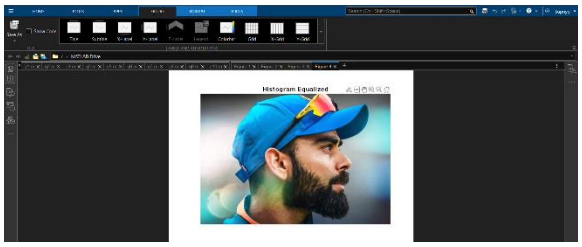
```
img = imread('vk.jpg');
light = img + 50;
dark = img - 50;
contrast_manupulate = imadjust(img, stretchlim(img), []);
negative = 255 - img;
figure;
subplot(1, 4, 1), imshow(light), title('Increased);
subplot(1, 4, 2), imshow(dark), title('Decreased);
subplot(1, 4, 3), imshow(contrast_manupulate), title('Contrast Manipulated');
subplot(1, 4, 4), imshow(negative), title('Negative);
```

   Output:



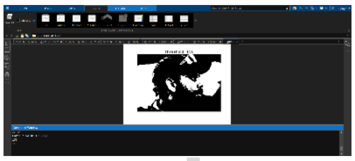
3. Read an image and perform histogram equalization of the input image and analyse the result.

Code:
```
 img = imread('vk.jpg');
 hist_equ = histeq(img);
    figure;
    subplot(1,1,1), imshow(hist_equ), title('Histogram Equalized');
```
Output:



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.

Code:
```
    img = imread('vk.jpg');
    img = im2gray(img);
    prompt = 'Enter a value (0-255): ';
    threshold = input(prompt);
    binary_img = img > threshold;
    figure;
             subplot(1, 1, 1), imshow(binary_img), title(['Threshold: ', num2str(threshold)]);
```
Output:



5. Read an image, convolve the image with the mass
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

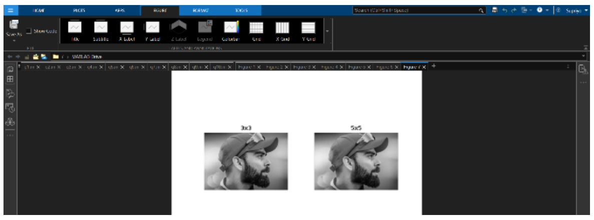
1	1	1
1	1	1
1	1	1
1	1	1
1	1	1

Code:

```
img = imread('vk.jpg');
gray_img = im2gray(img);
conv_img_3x3 = imfilter(gray_img, ones(3,3) / 9);
conv_img_5x5 = imfilter(gray_img, ones(5,5) / 25);


figure;
        subplot(1,2,1), imshow(conv_img_3x3), title('3x3');
        subplot(1,2,2), imshow(conv_img_5x5), title('5x5');
```

Output:



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.

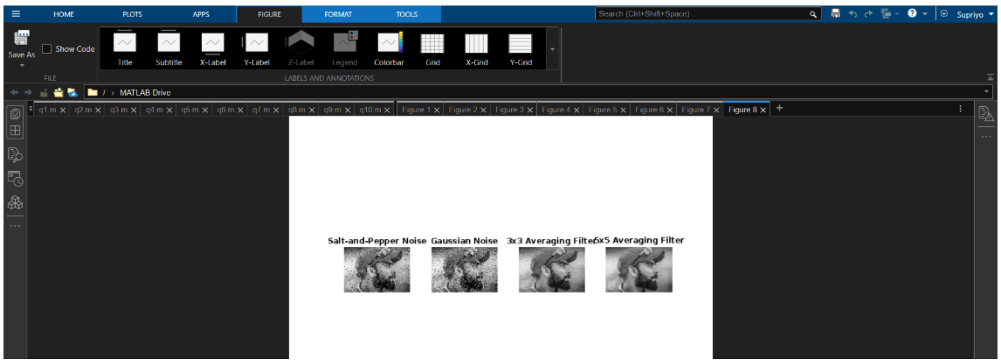
   Code:

```
  img = imread('vk.jpg');
  img = rgb2gray(img);


  noisy_img_salt_pepper = imnoise(img, 'salt & pepper', 0.05);
  noisy_img_gaussian = imnoise(noisy_img_salt_pepper, 'gaussian', 0, 0.01);
  filtered_img_gaussian_3x3 = imfilter(noisy_img_gaussian, ones(3) / 9, 'symmetric');
  filtered_img_gaussian_5x5 = imfilter(noisy_img_gaussian, ones(5) / 25, 'symmetric');


figure;
  subplot(1, 4, 1), imshow(noisy_img_salt_pepper), title('Salt-and-Pepper Noise');
  subplot(1, 4, 2), imshow(noisy_img_gaussian), title('Gaussian Noise');
  subplot(1, 4, 3), imshow(filtered_img_gaussian_3x3), title('3x3 Averaging Filter');
  subplot(1, 4, 4), imshow(filtered_img_gaussian_5x5), title('5x5 Averaging Filter');
```

Output:

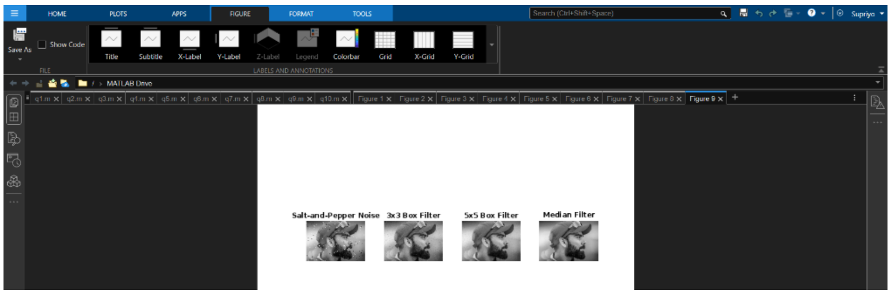


The results obtained from applying averaging filters (3x3 and 5x5) to image corrupted with salt-and-pepper noise and Gaussian noise demonstrate that with these filters help in reducing noise while also smoothing the image.

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.

   Code:
```
img = imread('vk.jpg');
gray_img = rgb2gray(img);
noisy_img = imnoise(gray_img, 'salt & pepper', 0.02);
box_filter_3x3 = fspecial('average', [3, 3]);
box_filter_5x5 = fspecial('average', [5, 5]);
box_3x3_img = imfilter(noisy_img, box_filter_3x3);
box_5x5_img = imfilter(noisy_img, box_filter_5x5);
median_img = medfilt2(noisy_img, [3, 3]);


figure;
subplot(1,4,1), imshow(noisy_img), title('Salt-and-Pepper Noise');
subplot(1,4,2), imshow(box_3x3_img), title('3x3 Box Filter');
subplot(1,4,3), imshow(box_5x5_img), title('5x5 Box Filter');
subplot(1,4,4), imshow(median_img), title('Median Filter');
```
   Output:



The results obtained from applying the median filter effectively removes the salt-and-pepper noise while preserving image details and edges better than the box filters.

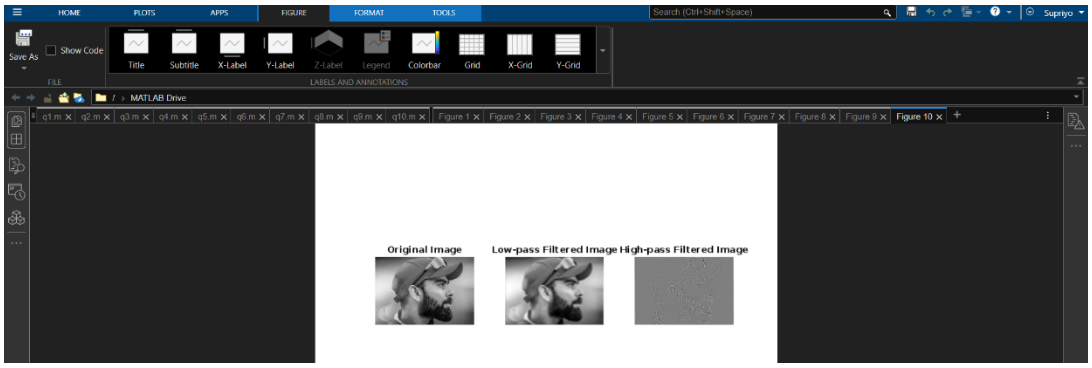
8. Write a matlab program that performs a two-dimensional Butterworth low-pass and high-pass filter of the given image for two different cut-off frequencies.

   code:
```
img = imread('vk.jpg');
gray_img = rgb2gray(img);
img_fft = fftshift(fft2(gray_img));
[M, N] = size(gray_img);
[X, Y] = meshgrid(1:N, 1:M);
centerX = ceil(N/2);
centerY = ceil(M/2);
distance = sqrt((X - centerX).^2 + (Y - centerY).^2);
cutoff_frequency = 30;
```

```
butter_lp = 1 ./ (1 + (distance / cutoff_frequency).^(2 * 2));
img_lp_filtered = real(ifft2(ifftshift(img_fft .* butter_lp)));
butter_hp = 1 - butter_lp;
img_hp_filtered = real(ifft2(ifftshift(img_fft .* butter_hp)));

figure;
        subplot(1,3,1), imshow(gray_img), title('Original Image');
        subplot(1,3,2), imshow(img_lp_filtered, []), title('Low-pass Filtered Image');
        subplot(1,3,3), imshow(img_hp_filtered, []), title('High-pass Filtered Image');
```

output:



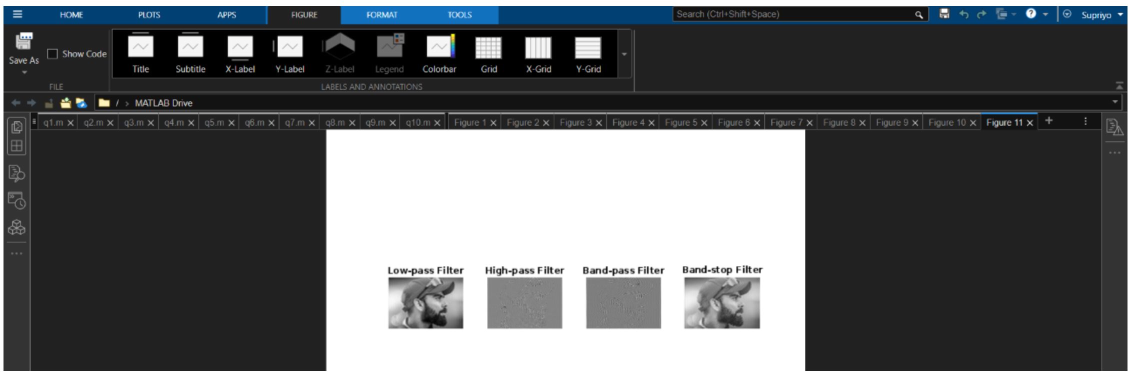
9. Read an input image to perform the following operations:
   - a. High-pass filtering in the frequency domain
   - b. Low-pass filtering in the frequency domain
   - c. Band-pass filter in the frequency domain
   - d. Band-stop filter in the frequency domain

Code:

```
img = imread('vk.jpg');
  gray_img = rgb2gray(img);
  img_fft = fftshift(fft2(gray_img));
  [M, N] = size(gray_img);
  [X, Y] = meshgrid(1:N, 1:M);
  centerX = ceil(N/2);
  centerY = ceil(M/2);
  distance = sqrt((X - centerX).^2 + (Y - centerY).^2);
  cutoff_frequency1 = 30;
  cutoff_frequency2 = 80;
  high_pass = distance > cutoff_frequency1;
  img_hp_filtered = real(ifft2(ifftshift(img_fft .* high_pass)));
  low_pass = distance < cutoff_frequency1;
  img_lp_filtered = real(ifft2(ifftshift(img_fft .* low_pass)));
  band_pass = (distance > cutoff_frequency1) & (distance < cutoff_frequency2);
  img_bp_filtered = real(ifft2(ifftshift(img_fft .* band_pass)));
  band_stop = 1 - band_pass;
  img_bs_filtered = real(ifft2(ifftshift(img_fft .* band_stop)));
```

```
figure;
        subplot(1,4,1), imshow(img_lp_filtered, []), title('Low-pass Filter');
        subplot(1,4,2), imshow(img_hp_filtered, []), title('High-pass Filter');
        subplot(1,4,3), imshow(img_bp_filtered, []), title('Band-pass Filter');
        subplot(1,4,4), imshow(img_bs_filtered, []), title('Band-stop Filter');
```

Output:



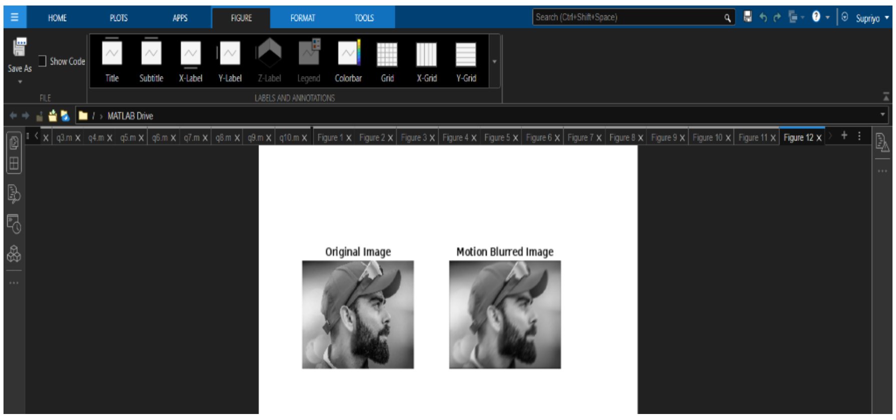
10. Read an image and degrade the image using motion blur.

Code:

```
img = imread('vk.jpg');
gray_img = rgb2gray(img);
motion_blur = fspecial('motion', 20, 45);
blurred_img = imfilter(gray_img, motion_blur);

figure;
        subplot(1,2,1), imshow(gray_img), title('Original Image');
        subplot(1,2,2), imshow(blurred_img), title('Motion Blurred Image');
```

Output:



GITHUB: GitHub-PCA-2-Link