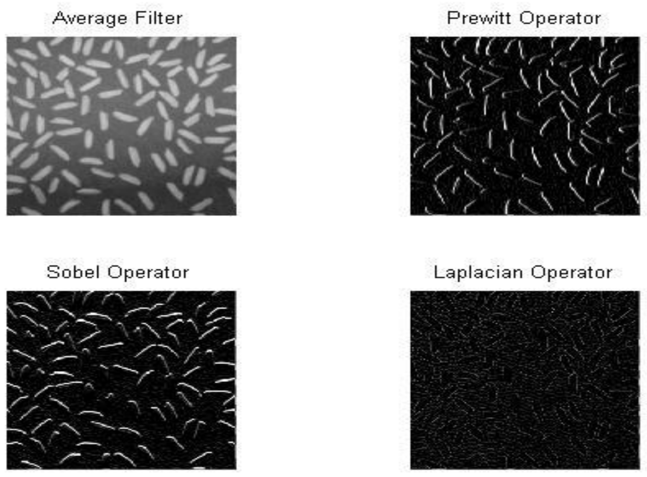
FILTERING AND EDGE DETECTION OF AN IMAGE

Q1: Neighborhood Averaging and Edge Detection

Main program:
```
clear all;close all;clc;
I = imread('rice.png');
In = imnoise(I,'salt & pepper');
subplot(121);imshow(I);
title('Original Image');
subplot(122);imshow(In)
title('Noisy Image');

kernal_averaging = (1/9)*ones(3);
kernal_prewitt = [-1 0 1; -1 0 1; -1 0 1];
kernal_sobel = [-1 -2 -1 ; 0 0 0 ;1 2 1];
kernal_laplacian = [0 1 0; 1 -4 1; 0 1 0];

figure;
subplot(2,2,1);
I_avg = convolution(double(I),kernal_averaging);
imshow(uint8(I_avg));title('Average Filter');

subplot(2,2,2);
I_prewitt = convolution(double(I),kernal_prewitt);
imshow(uint8(I_prewitt));title('Prewitt Operator');

subplot(2,2,3);
I_sobel = convolution(double(I),kernal_sobel);
imshow(uint8(I_sobel));title('Sobel Operator');

subplot(2,2,4);
I_laplacian = convolution(double(I),kernal_laplacian);
imshow(uint8(I_laplacian));title('Laplacian Operator');
```

Convolution function:
```
function out = convolution(image,kernel)
    [m,n]= size(image);
    a = size(kernel);
    I_double = double(image);
    for i=1:(n-(a-1))
        for j=1:(m-(a-1))
            I_double(i,j) = sum(sum(kernel.*I_double(i:(i+a-1),j:(j+a-1))));
        end
    end
    out = uint8(I_double);
end
```

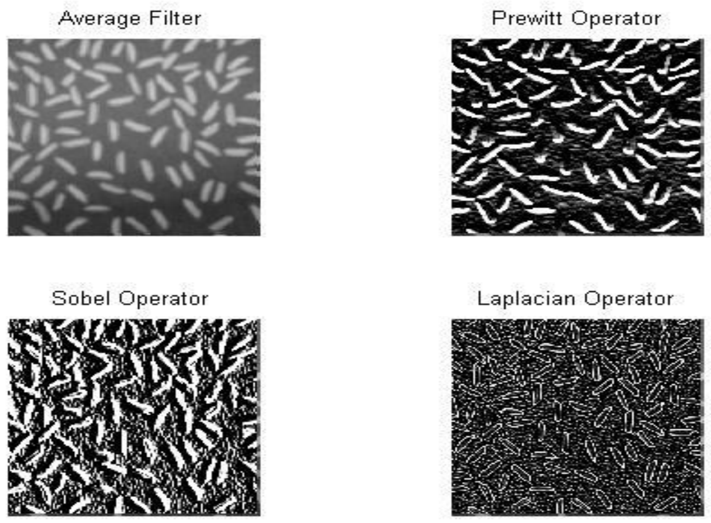
OUTPUT: **For kernel size of 3X3**



For kernel size 5X5:

```
kernal_averaging = (1/25)*ones(5);
kernal_prewitt = [2 2 2 2 2; 1 1 1 1 1; 0 0 0 0 0; -1 -1 -1 -1 -1; -2 -2 -2 -2 -2];
kernal_sobel = [1 2 0 -2 -1 ; 4 8 0 -8 -4 ; 6 12 0 -12 -6; 4 8 0 -8 -4; 1 2 0 -2 -1];
kernal_laplacian = [0 0 -1 0 0; 0 -1 -2 -1 0;-1 -2 16 -2 -1;0 -1 -2 -1 0; 0 0 -1 0 0];
```

OUTPUT:



For kernel size of 5X5

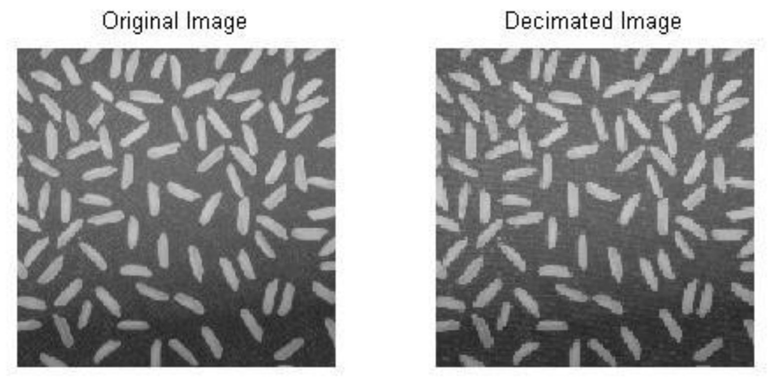
Conclusion:

As the size of the kernel is increased, the sharpness of the operation is also getting increased significantly which quite visible in the two pictures attached above.

Q2: Decimation of an Image

```
clear all; close all; clc;
image = double(imread('rice.png'));
[m,n] = size(image);
a=0; b=0;
Image_decimated = zeros();
for i=1:2:m
    a=a+1;
    b=0;
    for j=1:2:n
        b = b+1;
        Image_decimated(a,b)=image(i,j);
    end
end
subplot(122);
imshow(uint8(Image_decimated));title('Decimated Image');
subplot(121);
imshow(uint8(image));title('Original Image');
```

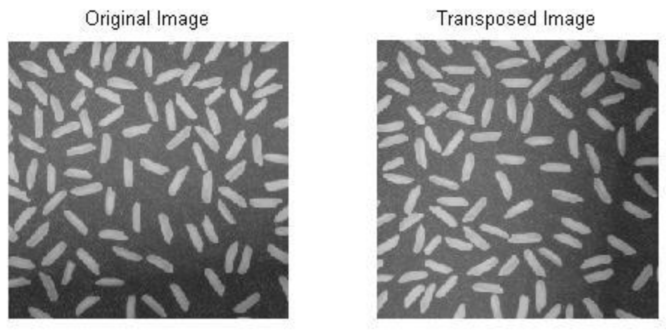
OUTPUT:



Q3: Transposing an Image

```
clear all; close all; clc;
image = double(imread('rice.png'));
[m,n] = size(image);
for i=1:m
    for j=1:n
        Image_transposed(i,j) = image(j,i);
    end
end
subplot(122);
imshow(uint8(Image_transposed));title('Transposed Image');
subplot(121);
imshow(uint8(image));title('Original Image');
```

OUTPUT:



Q4: Flipping an Image about its Vertical and Horizontal Axis

```
clear all; close all; clc;
image = double(imread('rice.png'));
[m,n] = size(image);
if(rem(n,2)==0)
    for i=1:n
        Flipped_imageV(:,i) = image(:,abs(n-i+1));
        Flipped_imageH(i,:) = image(abs(m-i+1),:);
    end
else
    for i=1:n
        if(i==(n+1)/2)
            Flipped_imageV(:,i) = image(:,i);
            Flipped_imageH(i,:) = image(i,:);
        else
            Flipped_imageV(:,i) = image(:,abs(n-i+1));
            Flipped_imageH(i,:) = image(abs(m-i+1),:);
        end
    end
end
subplot(131);imshow(uint8(image));title('Original Image');
subplot(132);imshow(uint8(Flipped_imageV));title('Vertically Flipped Image');
subplot(133);imshow(uint8(Flipped_imageH));title('Horizontally Flipped
Image');
```

OUTPUT: