ECE: 415
Image Analysis & Computer Vision I
Mid-Term Project

Name: Shreyas Gaonkar

UIN: 657613409

MASTER OF SCIENCE

IN

ELECTRICAL AND COMPUTER ENGINEERING

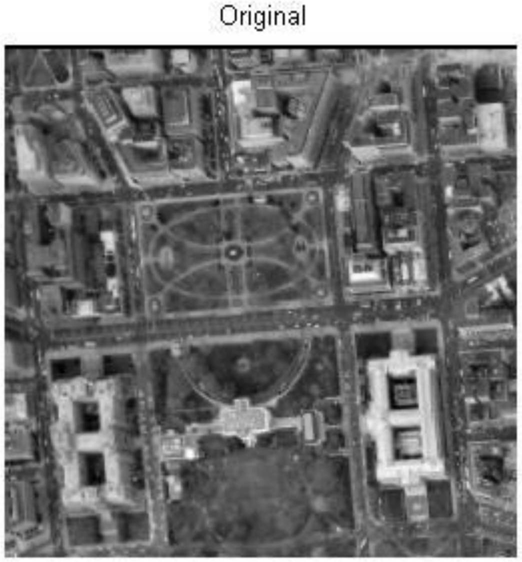
UNIVERSITY OF ILLINOIS AT CHICAGO

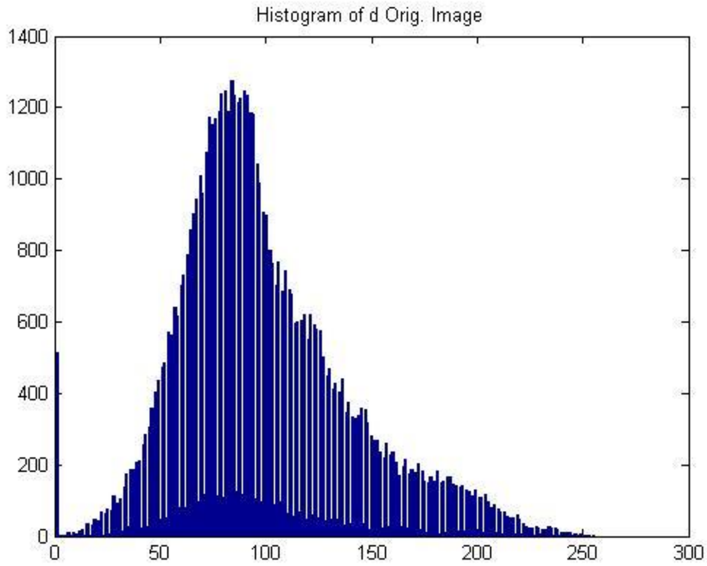
Q.1 Write code (to be submitted) to perform global histogram equalization. Use it to equalize the image “white_house256.jpg”. Plot the histograms of the image before and after equalization and also print the images before and after equalization.

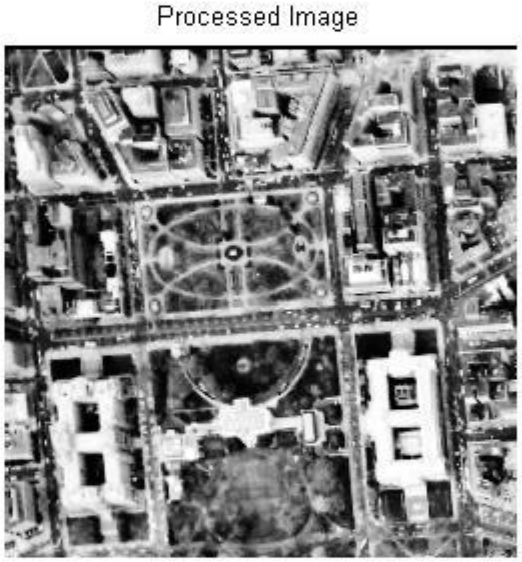
Code:

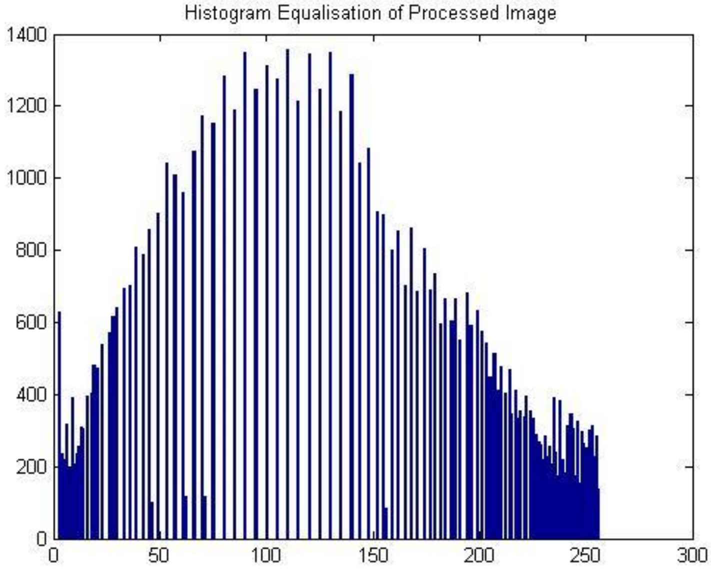
```
clear all
clc
I=imread('C:\Users\ShreyasG\Desktop\Image Analysis Project\white_house256.jpg');
I_edit=histeq(I);
I=double(I);      %To increase precision
maximum_value=max((max(I))); %To find the largest element of the Image array
[row col]=size(I);  %Return the value of Image size and store it in the variable
c=row*col;          %Calculate the total pixel as multiplication of rows and cols
h=zeros(1,256);
z=zeros(1,256);
for n=1:1:row
for m=1:1:col
if I(n,m) == 0
I(n,m)=1;
end
end
end
for n=1:1:row
for m=1:1:col
t = I(n,m);
h(t) = h(t) + 1;
end
end
pdf = h/c;        %Calculate the PDF of each pixel values and store
cdf(1) = pdf(1);  %Calculate the CDF of each pixel values and store
for x=2:1:maximum_value
cdf(x) = pdf(x) + cdf(x-1);
end
new = round(cdf * maximum_value);
new= new + 1;
for p=1:1:row
for q=1:1:col
temp=I(p,q);
b(p,q)=new(temp);
t=b(p,q);
z(t)=z(t)+1;
end
end
figure; imshow(uint8(I)) , title(' Original');
figure; bar(h) , title('Histogram of d Orig. Image');axis([0 255 0 1280]);
figure; imshow(uint8(b)) , title('Processed Image');
figure; bar(z) , title('Histogram Equalization of Processed Image');axis([0 255 0 1360])
```

Output Images:







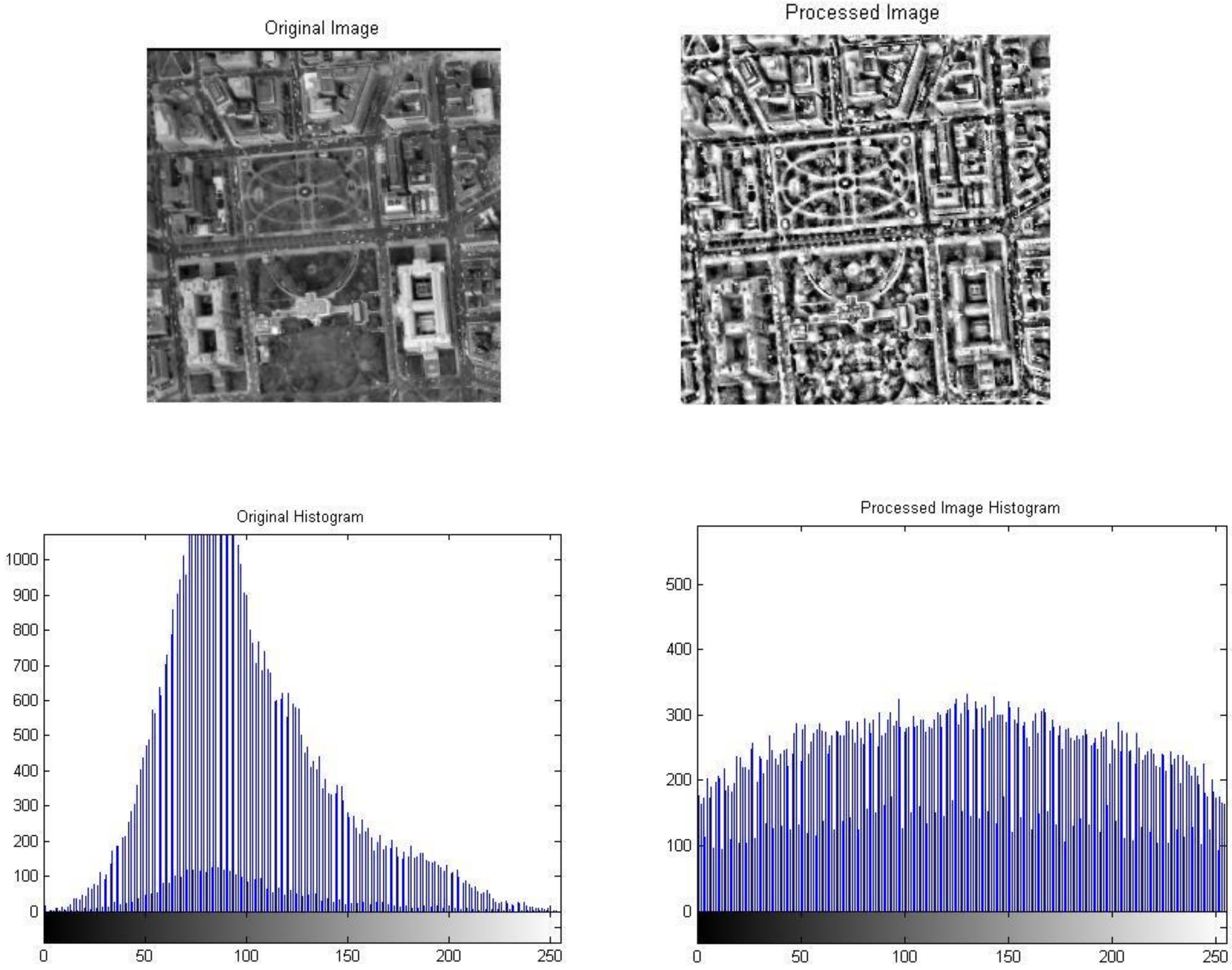


Q.2 Write a code that performs local histogram equalization with a window of 15x15 pixels on the image “white_house256.jpg”. Plot the histograms of the images before and after equalization and also print the images before and after equalization. To avoid tiling effects perform the equalizations with overlapped windows

Code:

```
clear all
clc
a = input('Enter size of padding');
b=a; %Enter Value of the padding, here it's 15
im=imread('C:\Users\ShreyasG\Desktop\Image Analysis Project\white_house256.jpg');
[M,N]=size(im);
impad = padarray(im,[((a-1)/2) ((b-1)/2)],0,'both'); %Padding = (N-1)/2. Here zero
padding of 7 will be used for 15x15filter

for u=0:1:M-a-1
%Sets limit on the rows from min to max with increment of one. Max value        would be
the last pixel of u-a-1 since the mask shouldn't go out of bounds.
for v=0:1:N-b-1
 %Sets limit on the columns from min to max with increment of one. Max value would be the
last pixel of v-a-1 since the mask shouldn't go out of bounds.
for i=1:1:a
%Sets limit on mask which will take the max value of what the user has entered
for j=1:1:b
%Sets limit on mask which will take the max value of what the user has entered
abc(i,j)=im(u+i,v+j); %Copy contents of the image to new variable
end
end
h=histeq(abc,256);
%Equalize the image with pixel values upto 256
xyz(u+1,v+1)=h(((a+1)/2),((b+1)/2));
%Copy the contents of the center pixel value into variable
end
end
figure;  imshow(im); title('Original Image');
figure;  imhist(im); title('Original Histogram');
figure;  imshow(xyz); title('Processed Image');
figure;  imhist(xyz); title('Processed Image Histogram');
```

Output Image:



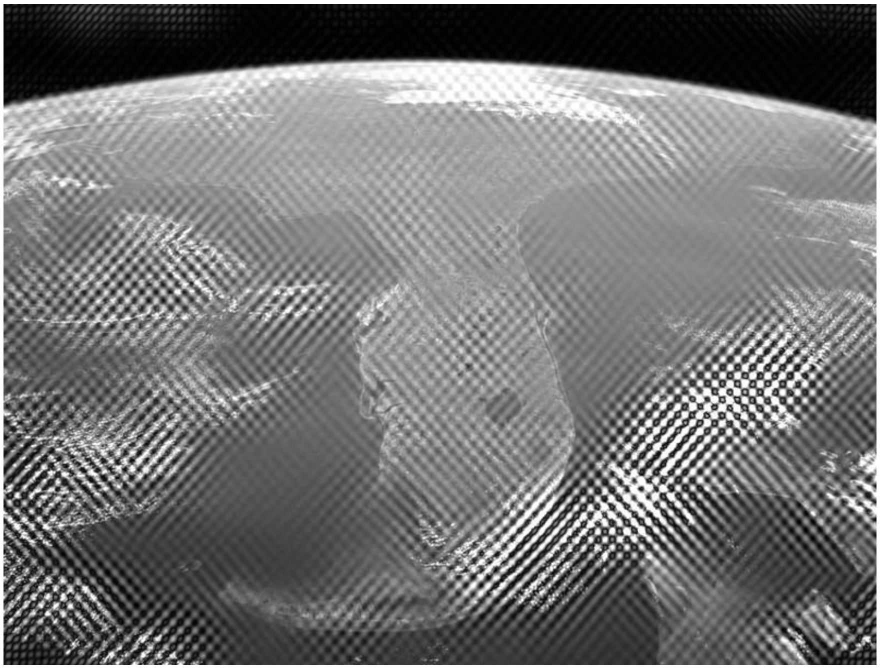
Q.3 Analyze the noisy image “nasaNoise.jpg” and find the properties of the interference noise. Design a filter to remove the interference. Plot the Fourier transform of the filter in 3D and print the image before and after filtering. Explain in detail your analysis and design.

Code:

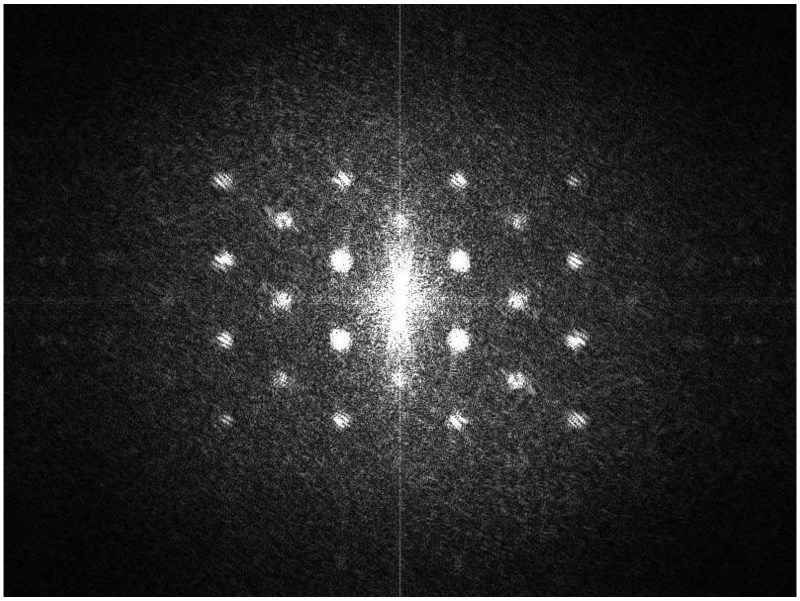
```
clc
clear all
image=(imread('C:\Users\ShreyasG\Desktop\Image Analysis Project\nasaNoise_1_.jpg'));
im2fft=fft2(image);
imshift=fftshift(im2fft);
figure
imshow(abs(imshift),[24 50000]);
[x,y]=ginput(24); %Manually select 24 points that needs to be filtered out and each x & y
pixel value is stored in variable x & y
[a,b]=size(imshift);
M=ones(601,801);
M1=M;
M2=M;
m=15;
%Radius of the mask. Here I have designed a filter to be working as a circle of fixed
radius which will mask out the pixel surrounding it’s center value.
for c=1:1:24 %For 36 point mask
for u=1:1:601 %Rows
for v=1:1:801 %Columns
if (m > ((((round(y(c))-u)^2)+((round(x(c))-v)^2))^0.5))) %Condition to find noise, if
it's greater than threshold, then set it to black with fixed radius
M(u,v)=0;
end
end
M1=M1.*M; %multiply to get the resultant image
end
end
figure;
imshow(M1); %Show the figure with the filtered points
%H3=ones(601,801);
%To further improve the filtering, we add mask around the image
for s=1:1:601
for r=1:1:801
if (s>=0 && s<=250)
if(r>=390 && r<=410)
M2(s,r)=0;
end
end
if (s>=351 && s<=601)
if(r>=390 && r<=410)
M2(s,r)=0;
end
end
if (s>=300 && s<=315)
if(r>=0 && r<=361)
M2(s,r)=0;
end
end
if (s>=300 && s<=315)
if(r>=437 && r<=801)
```

```
M2(s,r)=0;
end
end
end
end
mul=M1.*imshift.*M2;
imgfft=ifft2(mul); %Calculate FFT
figure; imshow(abs(mul),[25 50000]); %Show the Multiplied Image
figure; imshow(abs(imgfft),[0 255]); %Show Figure
```

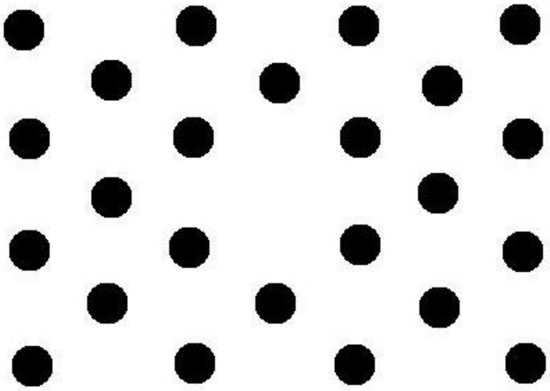
Output Images:



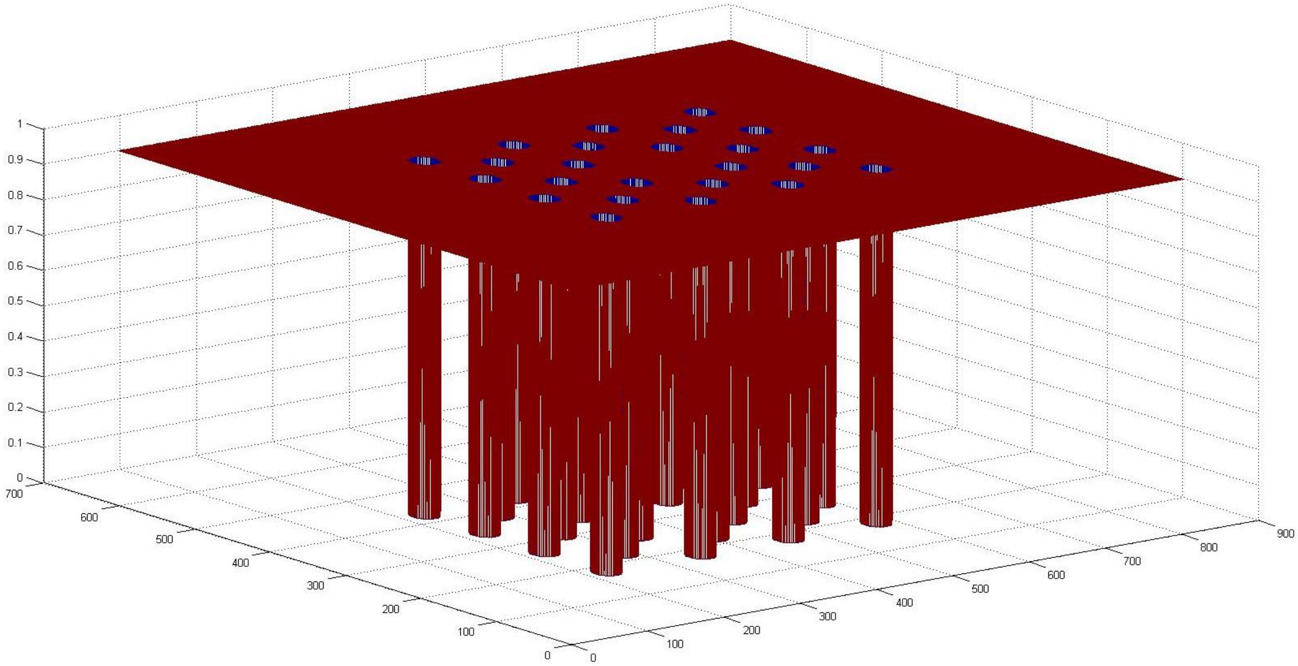
Given distorted image



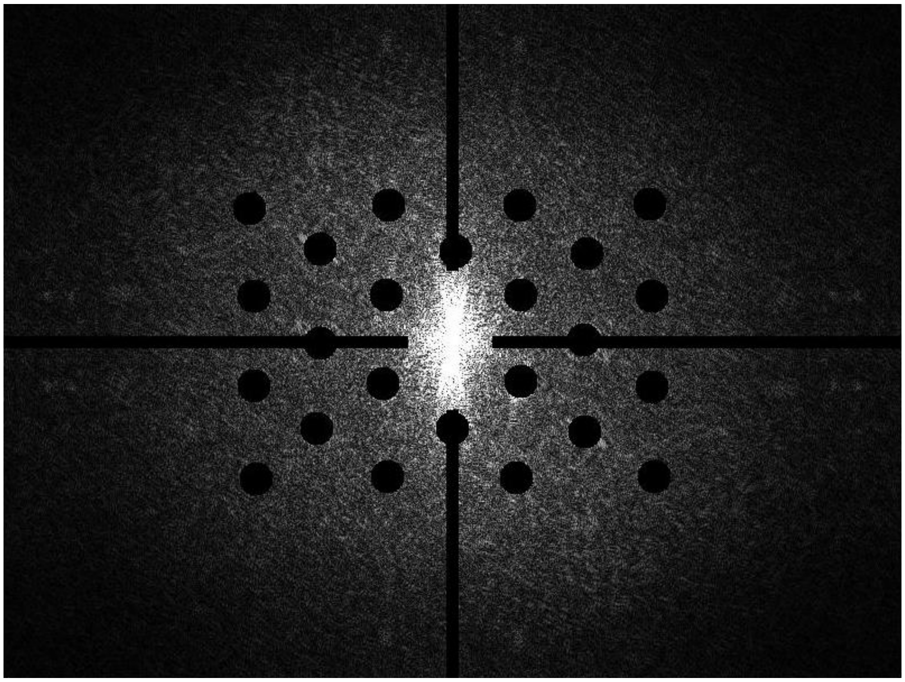
Fourier Transfer of the given image



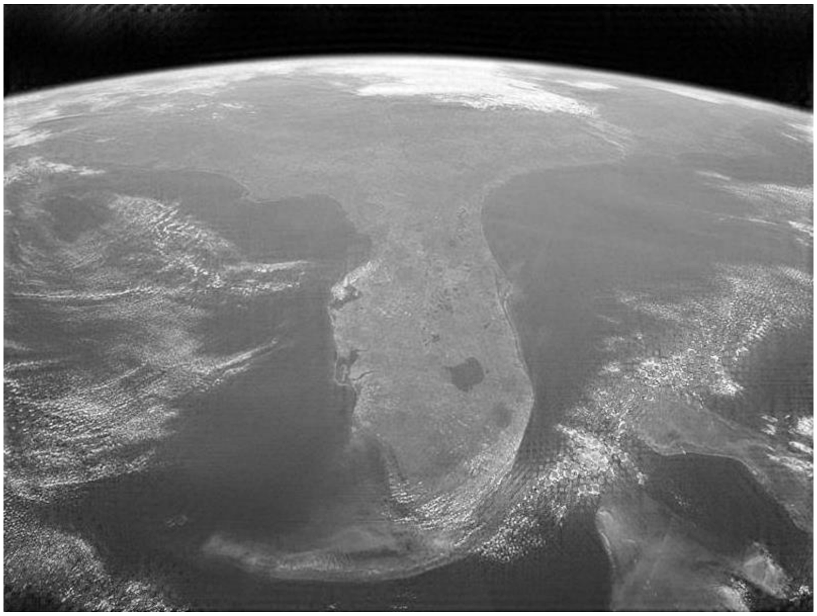
Mask for the above image



3D ‘Mesh’ of the mask



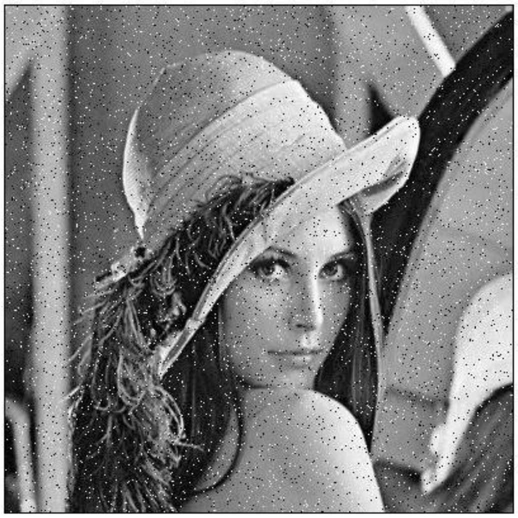
Multiplication of Fourier Transform and the Filter



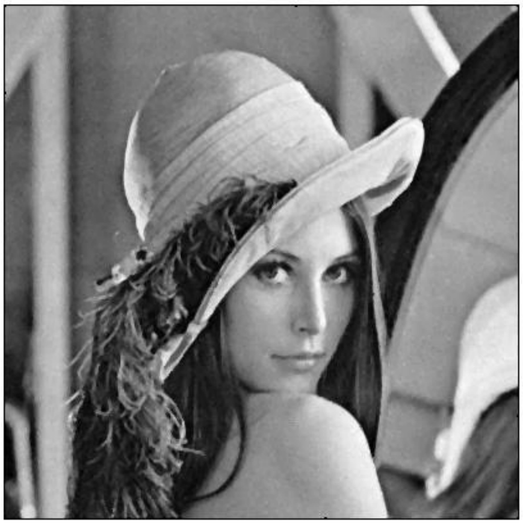
Final Processed Image

Q. 4 Design the best filter to remove the shot noise from the image “LenaDeltaNoise.jpg”. Select various sizes of the filter and try to find the optimal size. Explain your design and print the results with several such filters.

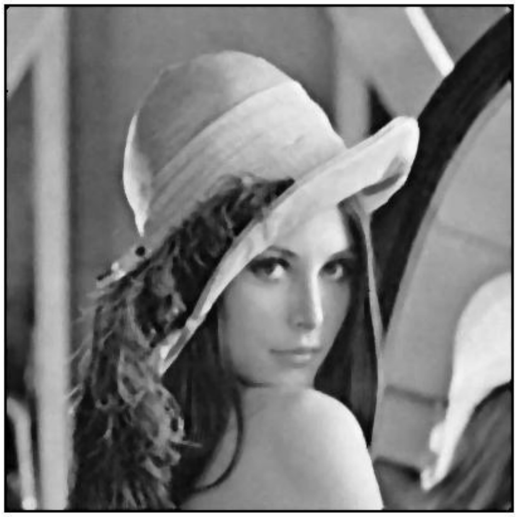
```
clear all;
clc;
I1 =imread('C:\Users\ShreyasG\Desktop\Image Analysis Project\LennaDeltaNoise.jpg');
I = padarray(I1,[1 1],0,'both'); %Padding for the first Image
I2 = padarray(I1,[2 2],0,'both'); %Padding for the second Image
a=double(I);  %Increase precision
s=double(I2); %Increase precision
b=a;
b1=s;
[row col]=size(a);

%3x3 Mask
for x=2:1:row-1;
for y=2:1:col-1;
a1=[a(x-1,y-1) a(x-1,y) a(x-1,y+1) a(x,y-1) a(x,y) a(x,y+1) a(x+1,y-1) a(x+1,y)
a(x+1,y+1)];
a2=sort(a1);
med=a2(5);
b(x,y)=med;
end
end

%5x5 Mask
for  d=3:1: row-2;
for e=3:1:col-2;
a11=[s(d-2,e-2) s(d-2,e-1) s(d-2,e) s(d-2,e+1) s(d-2,e+2) s(d-1,e-2) s(d-1,e-1) s(d-1,e)
s(d-1,e+1) s(d-1,e+2) s(d,e-2) s(d,e-1) s(d,e) s(d,e+1) s(d,e+2) s(d+1,e-2) s(d+1,e-1)
s(d+1,e) s(d+1,e+1) s(d+1,e+2) s(d+2,e-2) s(d+2,e-1) s(d+2,e) s(d+2,e+1) s(d+2,e+2)];
a21=sort(a11);
med1=a21(13);
b1(d,e)=med1;
end
end

%here above, I have tried to arrange the pixels present in the underlying layer of the
image to be sorted in the ascending order of their pixel values. The value at the center
of the mask is stored in a variable which is later used to filter out the noise. Function
‘sort’ is used to sort the pixel values in ascending order and to easily calculate the
center valued pixel.

subplot(1,3,1)
imshow(uint8(I)); title('Original Image');
subplot(1,3,2)
imshow(uint8(b)); title('3x3 Mask');
subplot(1,3,3)
imshow(uint8(b1)); title('5x5 Mask');
```



Original Image



3x3 Mask



5x5 Mask

In order to remove the shot noise in the image, I have implemented the use of median filtering for best optimized results in this case. I tried using couple of NxN masks which is applied to the original image and observed the result. The noise was easily filtered with lower order filter as well has higher order filters. Also, we come to know that increasing size of the filter causes the processed image to appear blur and less sharp. Thus 3x3 filtering provides the optimum filtering with noise removal and less blur, compared to 5x5 or any other higher masks.

Q.5 The image “PeanutRotated.jpg” is distorted by rotation, translation and scaling. Write a program to restore the given image using bilinear interpolation. Display the restored image and the given image. Detail the design parameters you used for the restoration.

Code:

```
clc;
clear all;
img = imread('C:\Users\ShreyasG\Desktop\Image Analysis Project\PeanutRotated-B.jpg');
figure; imshow(img);
[x,y,z]=size(img);
%[m,n,p]=size(img);
angle = 100;
xy = x*sqrt(2);
yx = y*sqrt(2);
for t=1:xy
for s=1:yx
i = uint16((t-xy/2)*(0.866)+(s-yx/2)*(-0.5)+x/2);
j = uint16(-(t-xy/2)*(-0.5)+(s-yx/2)*(0.866)+y/2);
if i>0 && j>0 && i<=x && j<=y
img2(t,s,:)=img(i,j,:);
end
end
end
figure;
imshow(img2); %Show Rotated Image
x,y,z]=size(img2);
for t=1:1:x
for s=1:1:y
i = uint16(t+122);
%value of 122 was found to be the perfect x-position shift
j = uint16(s+102);
%value of 102 was found to be the perfect y-position shift
if i>0 && j>0 && i<=x && j<=y
img3(t,s,:)=img2(i,j,:);
%Copy the shifted value into new variable img3 for displaying.
end
end
end
figure;
imshow(img3);
%Show Rotated plus translated Image
[x,y,z]=size(img3);
for t=1:1:x
for s=1:1:y
i = uint16((t)*((127/240)));
 %This is the ratio of the Image to the scaled image for x
j = uint16((s)*(250/310));
%This is the ratio of the Image to the scaled image for y
if i>0 && j>0 && i<=x && j<=y
img4(t,s,:)=img3(i,j,:);
%Copy the scaled value into new variable img4 for displaying.
end
end
end
figure;
imshow(img4);
```

Output Images:

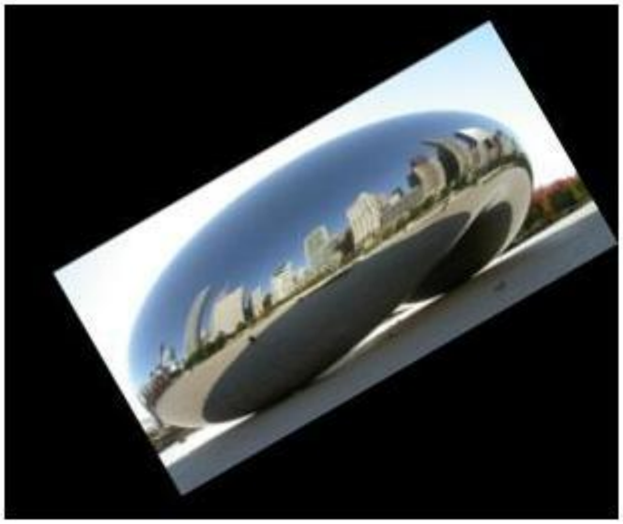


Figure 1- Original Image

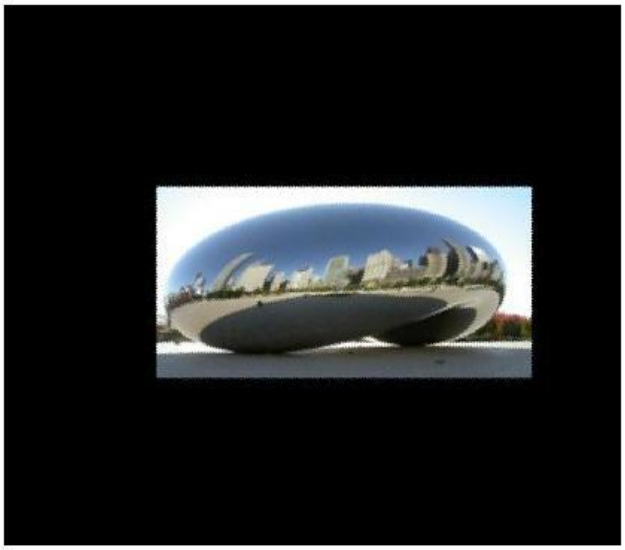


Figure 2- Image after Rotation

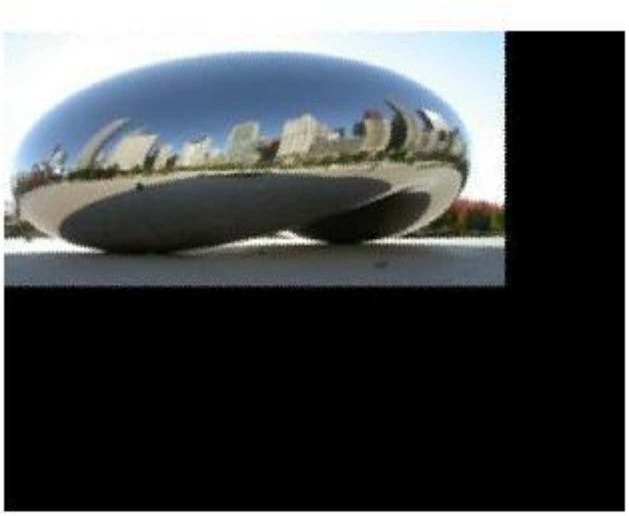


Figure 3- Image after Rotation and Translation

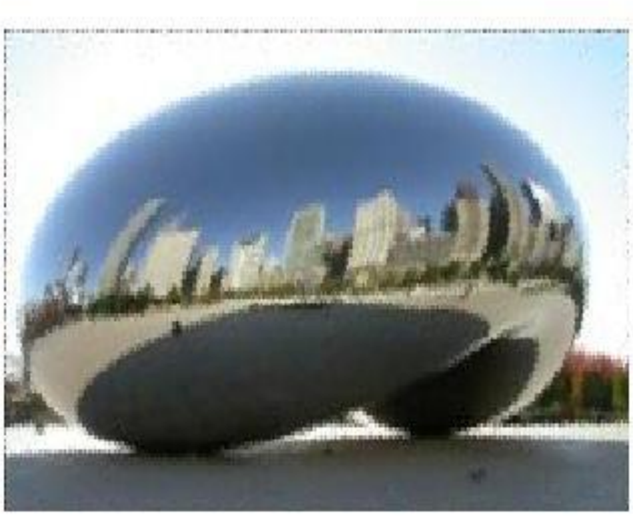


Figure 4- Image after Rotation, Translation and Scaling