# Math 155 Fall 2023 Homework 8 Solutions

UCLA ID: 605900342
Name: Caleb Traxler
Collaborators: N/A

By turning in this assignment, I agree by the UCLA honor code and declare that all of this is my own work.

## Problem 1 Periodic Noise Reduction Using a Notch Filter

(a) Write a program that implements sinusoidal noise of the form:

$n(x, y) = Asin(2\pi u_0 x + 2\pi v_0 y)$.

The program must be the amplitude, A, and the two frequency components $u_0$ and $v_0$.

Recall the algorithm from the lecture notes:

Given $f(x, y)$, multiply a degradation function $h(x, y)$ to $f(x, y)$, i.e. $H(f(x, y)) = h(x, y) * f(x, y)$. Then add noise n to $H(f(x, y))$ to obtain $g(x, y)$. Then apply a restoration filter, in this case a notch filter to then obtain $\hat{f}(x, y)$, which is the restored image.

Note that in this case I am not asked to apply any degradation function, thus I will assume that $H=$ the identity operator (no blur), only noise... Thus, $g(x, y) = f(x, y) + n(x, y)$.

Code:

```
%Math 155 Hw 8 Problem 1
f = imread('Fig5.26a.jpg');

f = double(f);

figure(), imshow(uint8(f)), title('Original Image');

[M, N] = size(f);

A = input('Enter Value for A: ');

u0 = input('Enter Value for u0: ');

v0 = input('Enter Value for v0: ');

n = zeros(M,N);

for x = 1:M
    for y = 1:N
        n(x,y) = A*sin(2 * pi* u0 * x + 2 * v0 * y);
        f(x,y) = f(x,y) + n(x,y);
    end
end

f = uint8(f);

figure();
imshow(f);
```

(b) Download image 5.26(a) of size M x N and add sinusoidal noise to it, with $v_0 = 0$.

The value of A must be high enough for the noise to be quite visible in the image (for example, you can take A = 100, $u_0 = 134.4, v_0 = 0$).

Code:

```
%Math 155 Hw 8 Problem 1
f = imread('Fig5.26a.jpg');

f = double(f);

figure(), imshow(uint8(f)), title('Original Image');

[M, N] = size(f);

A = 100;

u0 = 134.4;

v0 = 0;

n = zeros(M,N);

for x = 1:M
    for y = 1:N
        n(x,y) = A*sin(2 * pi* u0 * x + 2 * v0 * y);
        f(x,y) = f(x,y) + n(x,y);
    end
end

f = uint8(f);

figure();
imshow(f);
```

Output:

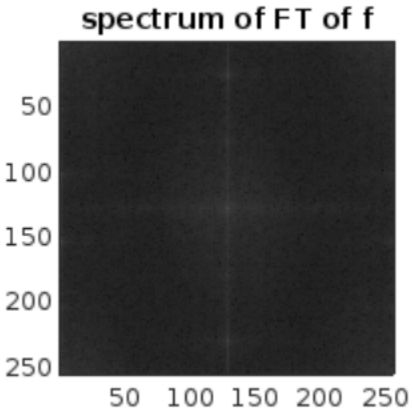

(c) Compute and display the degraded image and its spectrum (you may need to apply a log transform to visualize the spectrum).

Recall from homework 6 the process of compute the Fourier spectrum. I will re-use the code from homework 6:

Code:

```
%Math 155 Hw 8 Problem 1
f = imread('Fig5.26a.jpg');
f = double(f);
figure(), imshow(uint8(f)), title('Original Image');
[M, N] = size(f);
A = 100;
u0 = 134.4;
v0 = 0;
n = zeros(M,N);
for x = 1:M
    for y = 1:N
        n(x,y) = A*sin(2 * pi* u0 * (x-1) + 2 * pi * v0 * (y-1));
        f(x,y) = f(x,y) + n(x,y);
    end
end
f = uint8(f);
figure();
imshow(f);

%Specturm

% multiply f by (-1)^(x+y) to shift the center
for i = 1:M
   for j = 1:N
         %d = ((x-1),(y-1))
         d=(i-1)+(j-1);
         H(i,j)=f(i,j)*(-1)^d;
   end
end

% compute the DFT of f*(-1)^{x+y}
F=fft2(H);

% compute the spectrum
R=abs(F);

% Plot the centered spectrum, after rescaling using the log transform
% I know that s = clog(r+1) is the log transform -> S = clog(1+R)
c=5;
for i = 1:M
    for j = 1:N

        S(i,j)=c*log(1+R(i,j));
    end
end

% plot the spectrum (transformed by log for visualisation purposes)
figure
image(S); colormap(gray); axis image; title('spectrum of FT of f');
```

Output:

(d) Notch-filter the image using a notch filter of the form shown in Fig. 5.19(c) to remove the periodic noise.

Recall from lecture notes the definition and information about the Notch Filters: rejects or pass frequencies in a predefined neighborhood, symmetric about the center of the frequency rectangle.

$H_{NR} = 1 - H_{NP}$,

where $H_{NR}$ removes uniform textured patterns (periodic noise).

$H_{NR}(u,v) = \prod_{k=1}^{Q} H_{NR}^{k}(u,v) H_{NR}^{-k}(u,v)$, where $H_{NR}^{k}(u,v)$ and $H_{NR}^{-k}(u,v)$ are highpass filter transfer functions whose centers are at $(u_k, v_k)$ and $(-u_k, -v_k)$, resp. These centers are specified with respect to the center of the frequency rectangle $(M/2, N/2)$.

From the Lecture notes I see that I can define $H(u,v)$ as the following:

$$H(u,v) = \begin{cases} 0 & ,(u,v) = (\frac{M}{2}, \frac{N}{2}) \\ 1 & ,otherwise \end{cases}$$

I found that the bright spots are at four positions, which are located at N/2 and are

notch centers $= [128.5000, 230.9615; 129.1154, 181.1154; 130.7391, 79.9869; 129.8617, 26.0319]$

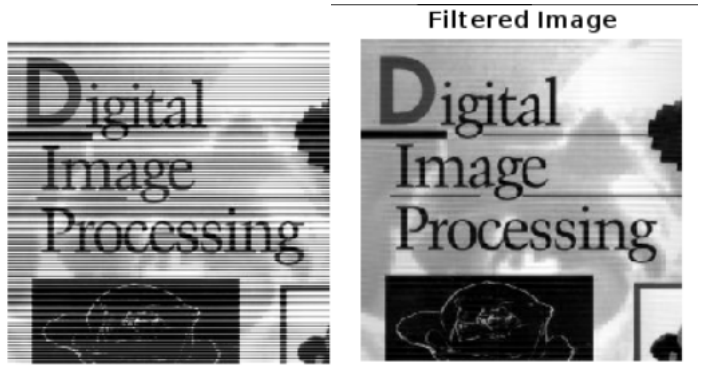
Now at each position, I will incorporate a 10x10 black rectangle to get rid of these bright spots. In return, I find that this ultimately reduced the sinusoidal noise, hence restoring the image.

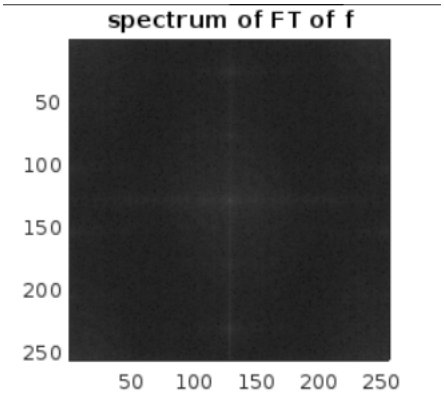
Code:

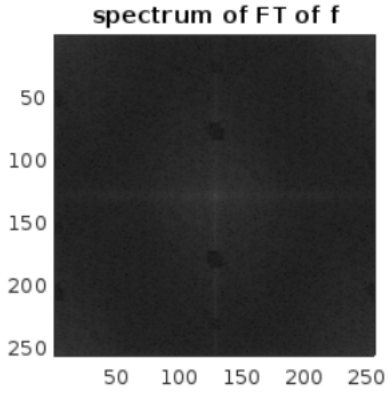
```
%Notch Filter
H_NR = ones(M, N);
notch_centers = [128.5000, 230.9615; 129.1154, 181.1154; 130.7391, 79.9869; 129.8617, 26.0319];
rect_size = [10, 10];  % 5x5 rectangle

% Loop over each notch center and apply a 5x5 rectangle
for i = 1:size(notch_centers, 1)
    u_k = notch_centers(i, 1);
    v_k = notch_centers(i, 2);

    % Calculate the indices of the rectangle
    rows = round(v_k-(rect_size(1)-1)/2):round(v_k+(rect_size(1)-1)/2);
    cols = round(u_k-(rect_size(2)-1)/2):round(u_k+(rect_size(2)-1)/2);

    % Ensure the indices are within the image boundaries
    rows(rows < 1 | rows > M) = [];
    cols(cols < 1 | cols > N) = [];

    % Set the values of H_NR to zero at the rectangle's location
    H_NR(rows, cols) = 0;
    H_NR(M-rows+1, N-cols+1) = 0; % Mirror the rectangle to the opposite quadrant
end

% Apply the notch filter to the shifted Fourier Transform
F_filtered = F_shifted .* H_NR;

% Inverse Fourier Transform to get the filtered image
f_filtered = ifft2(ifftshift(F_filtered));
f_filtered = real(f_filtered);  % Take the real part

% Display the filtered image
figure, imshow(uint8(f_filtered)), title('Filtered Image');
```

```
%%%%%%%%%%%%%%%%
%Updated Specturm

% multiply f by (-1)^(x+y) to shift the center
for i = 1:M
   for j = 1:N
         %d = ((x-1),(y-1))
         d=(i-1)+(j-1);
         H(i,j)=f_filtered(i,j)*(-1)^d;
   end
end

% compute the DFT of f*(-1)^{x+y}
F=fft2(H);

% compute the spectrum
R=abs(F);

% Plot the centered spectrum, after rescaling using the log transform
% I know that s = clog(r+1) is the log transform -> S = clog(1+R)
c=5;
for i = 1:M
    for j = 1:N

        S(i,j)=c*log(1+R(i,j));
    end
end

% plot the spectrum (transformed by log for visualisation purposes)
figure
image(S); colormap(gray); axis image; title('spectrum of FT of f');
```

Result:



# Problem 2

(a) Recall the definition of the convolution $f * g(x, y)$ in continuous variables in two dimensions.

Definition:

$f, g : R^2 \to R$, $f * g : R^2 \to R$

$f * g(x, y) = \int_\infty^\infty \int_\infty^\infty f(m, n)g(x - m, y - n)dmdn$

(b) Show that $\nabla^2(f * g) = f * (\nabla^2 g)$, where $\nabla^2$ denotes the Laplace operator in $(x, y)$.

Recall that $\nabla^2$ in $(x, y)$ is: $\nabla^2 = \frac{\partial^2}{\partial x^2}(x, y) + \frac{\partial^2}{\partial y^2}(x, y)$, thus I can write the following

$\nabla^2(f * g) = (\frac{\partial^2}{\partial x^2}(x, y) + \frac{\partial^2}{\partial y^2}(x, y)) \int_\infty^\infty \int_\infty^\infty f(m, n)g(x - m, y - n)dmdn$

$= \frac{\partial^2}{\partial x^2}(x, y) \int_\infty^\infty \int_\infty^\infty f(m, n)g(x - m, y - n)dmdn + \frac{\partial^2}{\partial y^2}(x, y) \int_\infty^\infty \int_\infty^\infty f(m, n)g(x - m, y - n)dmdn$

$= \int_\infty^\infty \int_\infty^\infty f(m, n)(\frac{\partial^2}{\partial x^2} g(x - m, y - n))dmdn + \int_\infty^\infty \int_\infty^\infty f(m, n)(\frac{\partial^2}{\partial y^2} g(x - m, y - n))dmdn$

$= \int_\infty^\infty \int_\infty^\infty f(m, n)((\frac{\partial^2}{\partial x^2} g(x - m, y - n)) + (\frac{\partial^2}{\partial y^2} g(x - m, y - n)))dmdn$

$= \int_\infty^\infty \int_\infty^\infty f(m, n)((\frac{\partial^2}{\partial x^2}) + (\frac{\partial^2}{\partial y^2})g(x - m, y - n))dmdn$

Thus, by the definition of the convolution in 2D, I can rewrite this in the following form

$\int_\infty^\infty \int_\infty^\infty f(m, n)((\frac{\partial^2 g}{\partial x^2}) + (\frac{\partial^2 g}{\partial y^2}))(x - m, y - n))dmdn = f * (\nabla^2 g)$

Therefore, I was able to show that $\nabla^2(f * g) = f * (\nabla^2 g)$, where $\nabla^2$ denotes the Laplace operator in $(x, y)$.

# Problem 3

Download form the class web page the image Fig5.07(b).jpg (X-Ray image corrupted by Gaussian noise).

(a) Write a computer program to implement the arithmetic mean filter of size 3x3. Apply the program to the image Fig5.07(b).jpg

Recall the arithmetic mean filter definition from lecture:

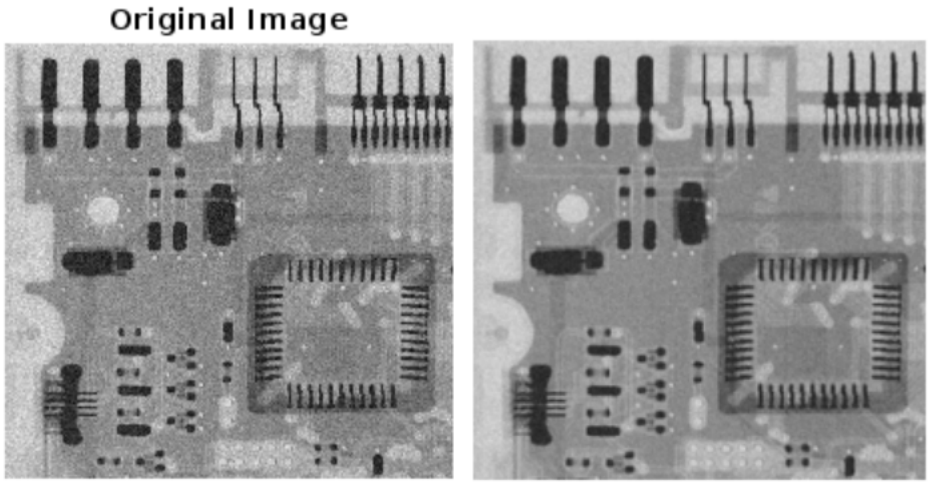
$\hat{f}(x,y) = \frac{1}{MN} \sum_{(s,t)\in S_{(x,y)}} g(s,t)$, such that $\hat{f}(x,y)$ is the restored image.

Note that this method works well for smooth local variations in an image. Noise gets reduced but has a blurring effect. Works well for random noise like Gaussian noise or uniform noise.

Code:

```
%Math 155 Hw 8 Problem 3

f = imread('Fig5.07(b).jpg');
f = double(f);
figure(), imshow(uint8(f)), title('Original Image');
[M, N] = size(f);
f_hat = zeros(M, N);

%For Loops for entire Image
for x = 1:M
    for y = 1:N
        %Reset sum to zero
        sum = 0;
        %Resent count to zero
        count = 0;
        %Subimage Foor Loops for 3x3
        for i = max(1, x-1):min(M, x+1)%3x3 neighborhood for (x,y)
            for j = max(1, y-1):min(N, y+1)%3x3 neighborhood for (x,y)
                sum = sum + f(i,j); %compound sum with subimage
                count = count + 1; %Increase counter, includes boarder
            end
        end
        f_hat(x,y) = sum / count; %Re-define the unnoisy image
    end
end
f_hat = uint8(f_hat);
figure();
imshow(f_hat);
```

Output:



(b) Write a computer program to implement the geometric mean filter of size 3x3. Apply the program to the image Fig5.07(b).jpg

Recall the geometric mean filter definition from lecture:

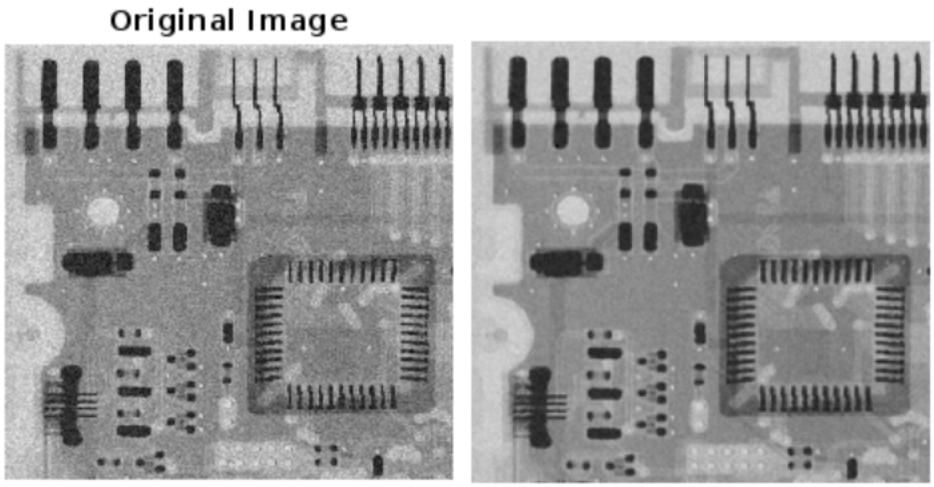
$\hat{f}(x,y) = [\prod_{(s,t)\in S_{(x,y)}}]^{\frac{1}{MN}}$

Note that the geometric mean filter is comparable to the arithmetic mean filter Ṫends to lose less image detail in the process. It works well for salt noise, fails for pepper noise, because $g(s,t) = 0$ is undefined. Works well for random noise like Gaussian noise or uniform noise.

Code:

```
%Math 155 Hw 8 Problem 3

f = imread('Fig5.07(b).jpg');
f = double(f);
figure(), imshow(uint8(f)), title('Original Image');
[M, N] = size(f);
f_hat = zeros(M, N);

%For Loops for entire Image
for x = 1:M
    for y = 1:N
        %Reset sum to zero
        prod = 1;
        %Resent count to zero
        count = 0;
        %Subimage Foor Loops for 3x3
        for i = max(1, x-1):min(M, x+1)%3x3 neighborhood for (x,y)
            for j = max(1, y-1):min(N, y+1)%3x3 neighborhood for (x,y)
                prod = prod * f(i,j); %compound product with subimage
                count = count + 1; %Increase counter, includes boarder
            end
        end
        f_hat(x,y) = prod^(1 / count); %Re-define the unnoisy image
    end
end
f_hat = uint8(f_hat);
figure();
imshow(f_hat);
```

Output:




(c) Explain your results. Evaluate the SNR (signal-to-noise-ratio) for both results in (a) and (b) (before denoising and after denoising). note, higher SNR, better denoised image. let $\hat{f}$ be the denoised image,

and f the clean true image. Then $SNR = 10log_{10}\frac{\sum_{x,y}(\hat{f})^2}{\sum_{x,y}(f-\hat{f})^2)}$. To evaluate the SNR before denoising, substitute $\hat{f}$ by g in the above formula.

Original : $SNR = 10log_{10}\frac{\sum_{x,y}(\hat{f})^2}{\sum_{x,y}(f-\hat{f})^2)}$

Denoised : $SNR = 10log_{10}\frac{\sum_{x,y}(g)^2}{\sum_{x,y}(f-g)^2)}$

(a) Arithmetic Mean:

Code:

```
%Math 155 Hw 8 Problem 3
f = imread('Fig5.07(b).jpg');
fclean = imread('Fig5.07(a).jpg');
f = double(f);
fclean = double(fclean);
figure(), imshow(uint8(f)), title('Original Image');
[M, N] = size(f);
f_hat = zeros(M, N);

%For Loops for entire Image
for x = 1:M
    for y = 1:N
        %Reset sum to zero
        LocalSum = 0;
        %Resent count to zero
        count = 0;
        %Subimage Foor Loops for 3x3
        for i = max(1, x-1):min(M, x+1)%3x3 neighborhood for (x,y)
            for j = max(1, y-1):min(N, y+1)%3x3 neighborhood for (x,y)
                LocalSum= LocalSum + f(i,j); %compound sum with subimage
                count = count + 1; %Increase counter, includes boarder
            end
        end
        f_hat(x,y) = LocalSum / count; %Re-define the unnoisy image
    end
end

figure();
imshow(uint8(f_hat));

% Corrected SNR calculation
SNRnoisy = 10 * log10(sum(f(:).^2) / sum((fclean(:) - f(:)).^2));
SNRdenoised = 10 * log10(sum(f_hat(:).^2) / sum((fclean(:) - f_hat(:)).^2));

% Corrected display of SNR values
disp(['SNR for noisy image: ', num2str(SNRnoisy)]);
disp(['SNR for denoised image: ', num2str(SNRdenoised)]);

% MSE calculation for noisy image
MSE_noisy = sum((fclean(:) - f(:)).^2) / numel(fclean);

% MSE calculation for denoised image
MSE_denoised = sum((fclean(:) - f_hat(:)).^2) / numel(fclean);

% Display the MSE values
disp(['MSE for noisy image: ', num2str(MSE_noisy)]);
disp(['MSE for denoised image: ', num2str(MSE_denoised)]);
```

Output:

```
>> ArithmeticMeanFIlter
SNR for noisy image: 14.1841
SNR for denoised image: 14.0501
MSE for noisy image: 841.4651
MSE for denoised image: 856.0437
>>
```

Explanation: The SNR after using the arithmetic mean filter is 14.0501, which is slightly lower than the SNR for the noisy image, which seems incorrect.

(b) Geometric Mean:

Code:

```
%Math 155 Hw 8 Problem 3

f = imread('Fig5.07(b).jpg');
fclean = imread('Fig5.07(a).jpg');
f = double(f);
fclean = double(fclean);
figure(), imshow(uint8(f)), title('Original Image');
[M, N] = size(f);
f_hat = zeros(M, N);

%For Loops for entire Image
for x = 1:M
    for y = 1:N
        %Reset sum to zero
        LocalProd = 1;
        %Resent count to zero
        count = 0;
        %Subimage Foor Loops for 3x3
        for i = max(1, x-1):min(M, x+1)%3x3 neighborhood for (x,y)
            for j = max(1, y-1):min(N, y+1)%3x3 neighborhood for (x,y)
                LocalProd = LocalProd * f(i,j); %compound product with subimage
                count = count + 1; %Increase counter, includes boarder
            end
        end
        f_hat(x,y) = LocalProd^(1 / count); %Re-define the unnoisy image
    end
end
figure();
imshow(uint8(f_hat));

% Corrected SNR calculation
SNRnoisy = 10 * log10(sum(f(:).^2) / sum((fclean(:) - f(:)).^2));
SNRdenoised = 10 * log10(sum(f_hat(:).^2) / sum((fclean(:) - f_hat(:)).^2));

% Corrected display of SNR values
disp(['SNR for noisy image: ', num2str(SNRnoisy)]);
disp(['SNR for denoised image: ', num2str(SNRdenoised)]);

% MSE calculation for noisy image
MSE_noisy = sum((fclean(:) - f(:)).^2) / numel(fclean);

% MSE calculation for denoised image
MSE_denoised = sum((fclean(:) - f_hat(:)).^2) / numel(fclean);

% Display the MSE values
disp(['MSE for noisy image: ', num2str(MSE_noisy)]);
disp(['MSE for denoised image: ', num2str(MSE_denoised)]);
```

Output:

```
>> GeometricMeanFilter
SNR for noisy image: 14.1841
SNR for denoised image: 14.0528
MSE for noisy image: 841.4651
MSE for denoised image: 843.1962
>>
```

Explanation: The SNR after using the geometric mean filter is 14.0528, which is also slightly lower than the noise (seems incorrect) but very marginally higher than the arithmetic mean filter, which seems correct.

Overall Explanation: The SNR for the geometric mena filter is slightly larger than the arithmetic mean filter, which implies that the geometric mean filter produces a slightly better and more clear result compared to the arithmetic mean filter. Also note that the SNR of the original image is close to the SNR for the arithmetic and geometric SNR's, which implies that the denoised images are close to the original image.