Image Processing Lab 03

Mohammad Sorkhian

This lab is about frequency-domain filtering and has been consisted of three main parts. In the first two sections, we get familiar with the implementation of frequency domain filtering. And we generate different outputs for a single image with various filter types and parameters like cutoff and order. In the third part, there are some discussions that we will address in the following.

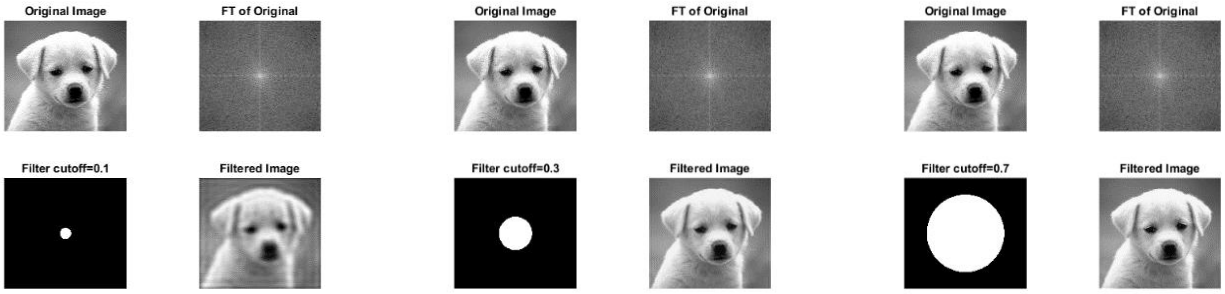
Part 1 : Implemented code

```
img_rgb=imread('puppy.jpg');      % Read the image
img=rgb2gray(img_rgb);            % Convert to Grayscale
im_size=size(img);                % Obtain the size of the image
P=2*im_size(1); Q=2*im_size(2);   % Optaining padding parameters as 2*image size
FTIm=fft2(double(img),P,Q);       % FT with padded size
filter_type = 'ideal';
lp_hp = 'lp';
cutoff = 0.3;
n=1;                              % For use only in Butterworth filters(n)>0
D0 = cutoff*im_size(1);           % Cutoff freqency radius is 0.1 times the hight of
the image
% Filter_type = ('ideal' or 'btw' or 'gaussian')
% lp_or_hp = ('lp' or 'hp' for low pass or high pass),
Filter = lp_hp_filters(filter_type , lp_hp, P, Q, D0, n);
Filtered_image=real(ifft2(Filter.*FTIm));                  % multiply the FT of
image by the filter and turn back to spatial domain
Filtered_image=Filtered_image(1:im_size(1), 1:im_size(2)); % Resize the image (
undo padding)
Fim=fftshift(FTIm);                                        % move the origin of the
FT to the center
FTI=log(1+abs(Fim));                                       % compute the magnitude
(log to brighten display)
Ff=fftshift(Filter);                                       % move the origin of the
FT to the center
FTF=log(1+abs(Ff));                                        % compute the magnitude
(log to brighten display)
lable = sprintf('Filter cutoff=%.1f', cutoff);
subplot(2,2,1), imshow(img,[]), title('Original Image');     % show the image
subplot(2,2,2), imshow(FTI,[]), title('FT of Original');   % show the image
subplot(2,2,3), imshow(FTF,[]), title(lable);              % filter
subplot(2,2,4), imshow(Filtered_image,[]), title('Filtered Image');  % show the
image
```

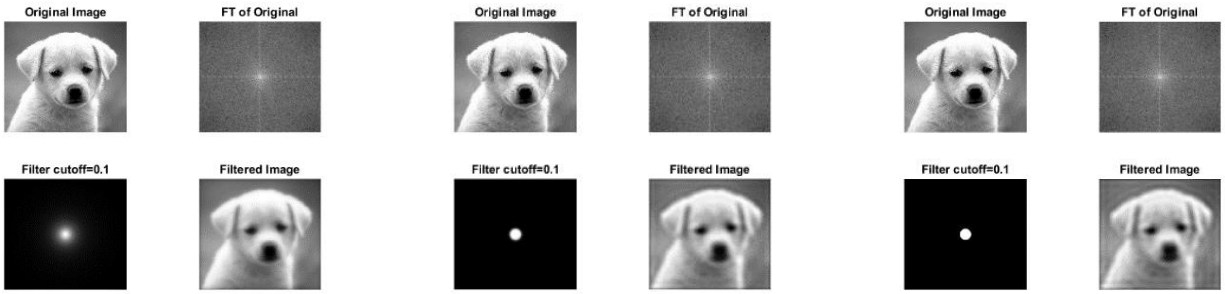
```
function H_out = lp_hp_filters(filter_type, lp_or_hp, P, Q, D0, n)
% The function calculates the LP or HP filters based on the parameters
% given. Filter_type=('ideal' or 'btw' or 'gaussian'), %lp_or_hp=('lp' or 'hp' for
low pass or high pass),
% P, Q are the padded image size, D0 is the cutoff frequency, and n for the
% order of the filter for btw filters
%  Developing frequency domain coordinates
u = 0:(P-1);
v = 0:(Q-1);

idx = find(u > P/2);
u(idx) = u(idx) - P;
idy = find(v > Q/2);
v(idy) = v(idy) - Q;
% Compute the meshgrid coordinates
[V, U] = meshgrid(v, u);
% Compute the istance matrix
D = sqrt(U.^2 + V.^2);

% Begin fiter computations.
switch filter_type
case 'ideal'
   H = double(D <=D0);
     if (strcmp(lp_or_hp,'lp'))
      H_out=H;
   elseif (strcmp(lp_or_hp,'hp'))
     H_out=1-H;
   else
       error('Filter should be Low pass (lp) or High pass (hp).')
    end
case 'btw'
    if (n==0)
        error('Butterworth Filter should have order (n)>0.')
    end
      H = 1./(1 + (D./D0).^(2*n));
    if (strcmp(lp_or_hp,'lp'))
      H_out=H;
   elseif (strcmp(lp_or_hp,'hp'))
     H_out=1-H;
   else
       error('Filter should be Low pass (lp) or High pass (hp).')
    end
case 'gaussian'
   H = exp(-(D.^2)./(2*(D0^2)));
   if (strcmp(lp_or_hp,'lp'))
      H_out=H;
   elseif (strcmp(lp_or_hp,'hp'))
     H_out=1-H;
   else
       error('Filter should be Low pass (lp) or High pass (hp).')
    end
otherwise
   error('Unknown filter type.')
   end
```

Part 2: Output images

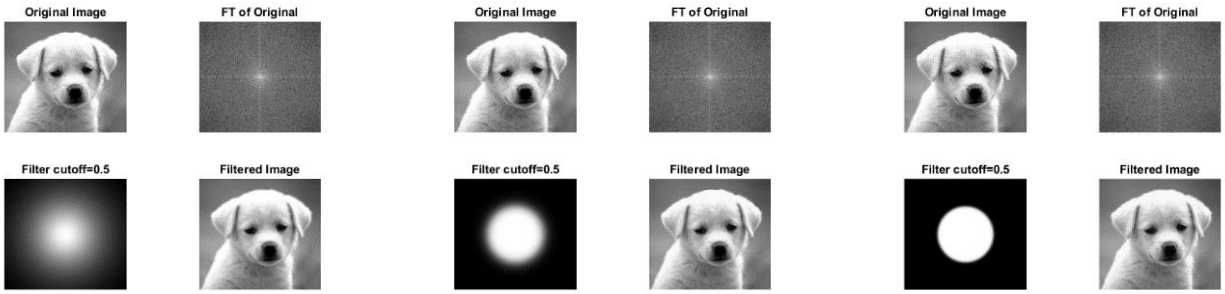
1. Ideal - Low pass – cutoff=0.1, 0.3, 0.7



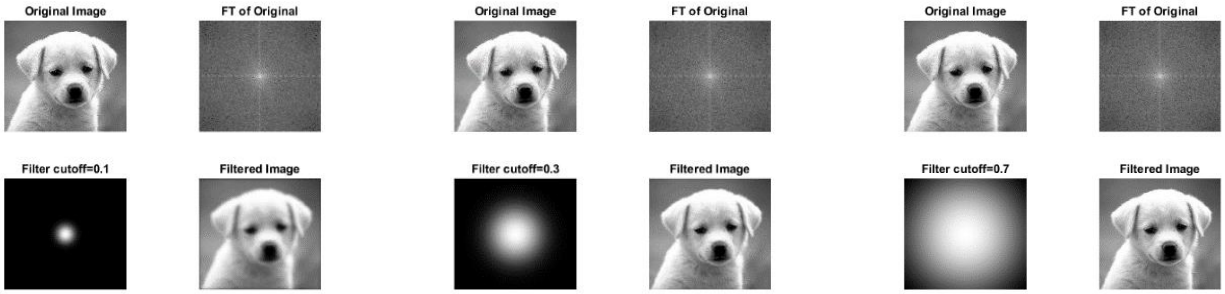
2. Butterworth - Low pass - cutoff=0.1 - n=1, 5, 20



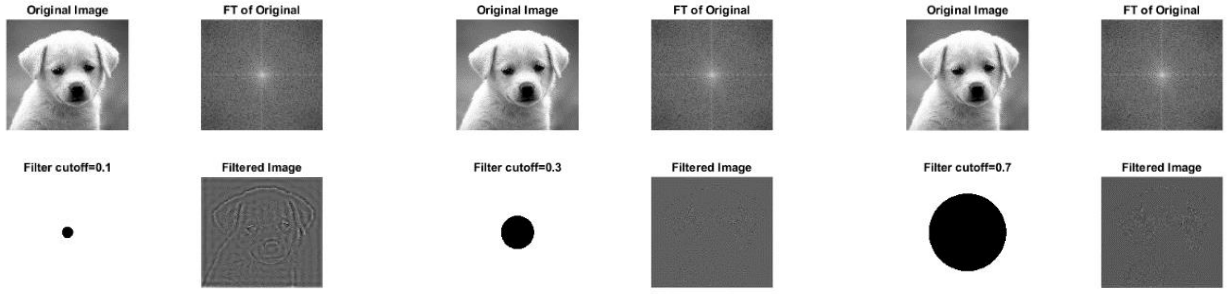
Butterworth - Low pass - cutoff=0.5 - n=1, 5, 20



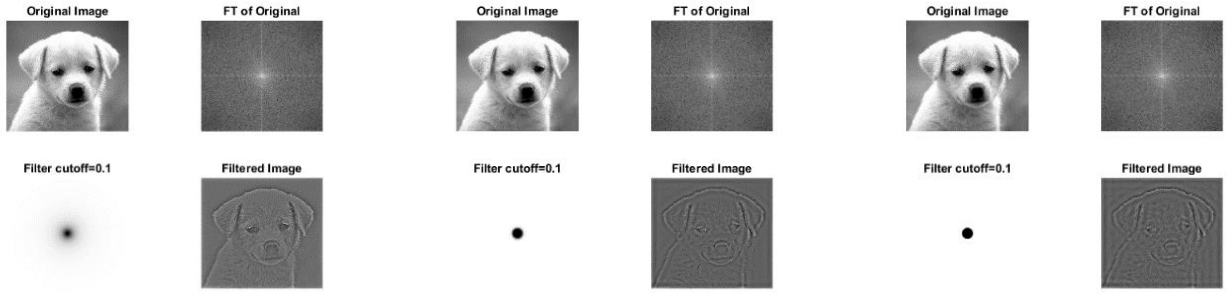
3. Gaussian - Low pass - cutoff=0.1, 0.3, 0.7



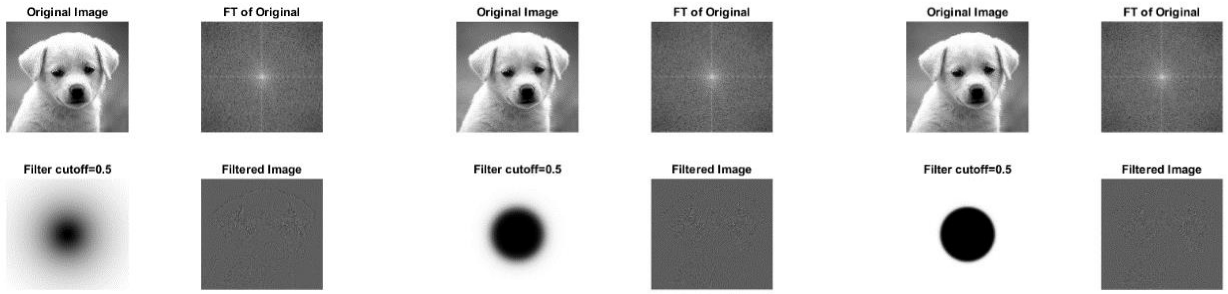
4. Ideal - High pass - cutoff=0.1, 0.3, 0.7



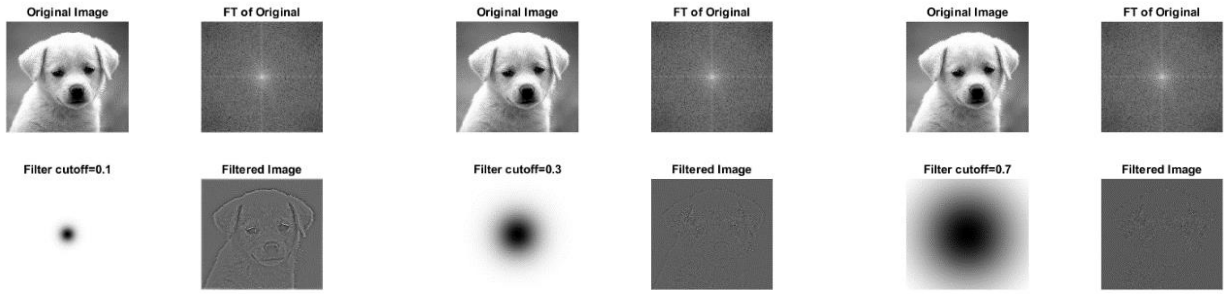
5. Butterworth - High pass - cutoff=0.1 - n=1, 5, 20



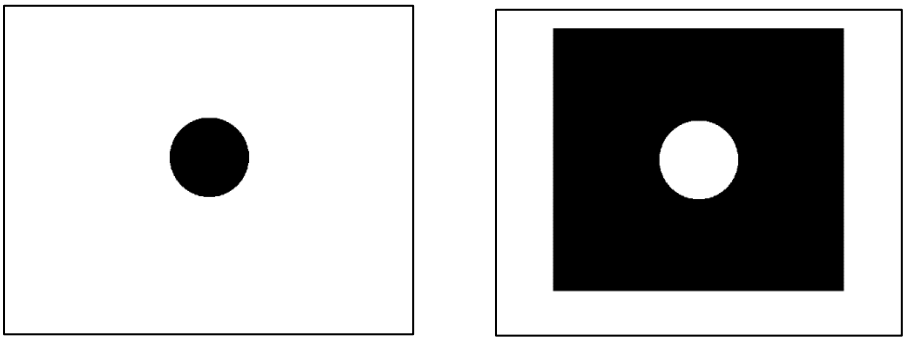
Butterworth - High pass - cutoff=0.5 - n=1, 5, 20



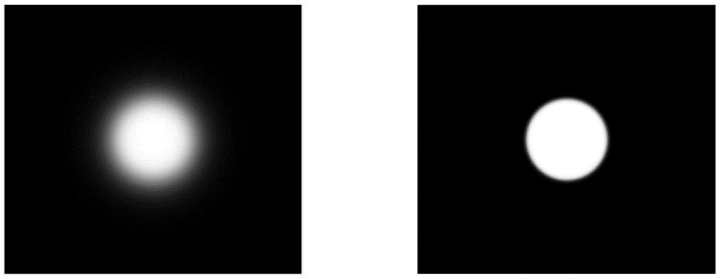
6. Gaussian - High pass - cutoff=0.1, 0.3, 0.7



1. An Ideal Low Pass Filter eliminates frequencies above the cutoff frequency, while lets the other low frequencies pass completely. But in the high pass Ideal filter, this procedure is reversed. In Ideal Filter, we can define a radius as a threshold (D0) and set any values in the radius (under the threshold) to one and others to zero or vice versa to represent low or high pass filters. The following images represent Ideal high and low pass filters. Frequency in the center of these images is zero and increases as the distance increase from the center. Therefore, when we increment the radius in the low-pass filter, we let higher frequency components pass and indeed decrease the smoothing effect. On the other hand, in high-pass filter increasing the radius, augment the sharpening effect since we are filtering lower frequencies more and just let higher ones to pass.



2. Unlike the Ideal filter, in the Butterworth filter, we do not have a sharp transition (clear cutoff) between filtered and passed frequencies, but Butterworth filter has another parameter(n) which controls the intensity of cutoff frequency. As we increase n, cutoff becomes clearer and more similar to the Ideal filter. (left n=3, right n=20)



3. In the Ideal filter, we have a phenomenon called ringing effect, while in the Gaussian, we do not have this problem since the cutoff transition in Gaussian is very smooth. Butterworth filter(n=1) relatively represents the Gaussian filter, but in the Gaussian filter, we have a smoother cutoff than the Butterworth. It means that in the Gaussian filter, the transition between passed and filtered frequencies is more blur, which leads to a smoother output picture. Additionally, in the Butterworth filter, we have the potential of having a ringing effect as we increase n.