신호 및 시스템 프로젝트 과제

2014732040 김병준

- 파일 불러오기

```
% open an image file. Ex. ‘Lena.raw’
fid = fopen('Lena.raw','r');
lena_raw  = fread(fid , 'float64');
fclose(fid);
lena_rgb = reshape(lena_raw,[256,256,3]);
```

- Task 1

```
% Task 1 – 컬러 영상 디스플레이
figure(1);
imshow(lena_rgb), title('컬러 Lena');
```

- Task 2

```
% Task 2 – 컬러 영상을 회색조 영상으로 변환하기
figure(2);
lena_gray = rgb2gray(lena_rgb);
imshow(lena_gray), title('회색조 변환 Lena');
```

회색조 변환 Lena

- Task 3

```
% Task 3 – 회색조 영상의 grey level 반전
figure(3);
inverseGrayImage = imcomplement(lena_gray);
imshow(inverseGrayImage), title('grey level 반전 Lena');
```

grey level 반전 Lena

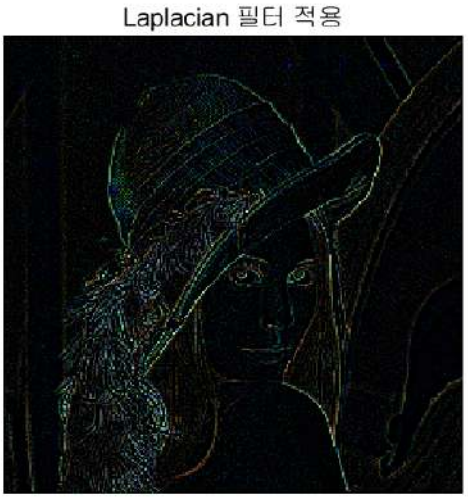
- Task 4

```
% Task 4 – 칼라 영상에 대하여 공간 영역에서 moving average filter (3x3, 5x5, 7x7) 적용,
(R,G.B) 각 component 에 대하여 동일한 filtering
figure(4);
subplot(2, 2, 1), imshow(lena_rgb), title('원본');
windowSize = 3; % 커널 사이즈
kernel = ones(windowSize) / windowSize ^ 2;
R = conv2(lena_rgb(:,:,1), kernel, 'same');
G = conv2(lena_rgb(:,:,2), kernel, 'same');
B = conv2(lena_rgb(:,:,3), kernel, 'same');
```

```
blurredImage = cat(3,R,G,B); % R G B 값 결합
subplot(2, 2, 2), imshow(blurredImage, []), title('3x3 이동평균 필터 적용');
windowSize = 5;
kernel = ones(windowSize) / windowSize ^ 2;
R = conv2(lena_rgb(:,:,1), kernel, 'same');
G = conv2(lena_rgb(:,:,2), kernel, 'same');
B = conv2(lena_rgb(:,:,3), kernel, 'same');
blurredImage = cat(3,R,G,B);
subplot(2, 2, 3), imshow(blurredImage, []), title('5x5 이동평균 필터 적용');
windowSize = 7;
kernel = ones(windowSize) / windowSize ^ 2;
R = conv2(lena_rgb(:,:,1), kernel, 'same');
G = conv2(lena_rgb(:,:,2), kernel, 'same');
B = conv2(lena_rgb(:,:,3), kernel, 'same');
blurredImage = cat(3,R,G,B);
subplot(2, 2, 4), imshow(blurredImage, []), title('7x7 이동평균 필터 적용');
```

- Task 5

```
% Task 5 – 칼라 영상에 대하여 공간 영역에서 Laplacian filter (high pass filter) 적용
figure(5);
laplacian_h = fspecial('laplacian');
B_laplacian = imfilter(lena_rgb, laplacian_h);
imshow(B_laplacian), title('Laplacian 필터 적용');
```



- Task 6

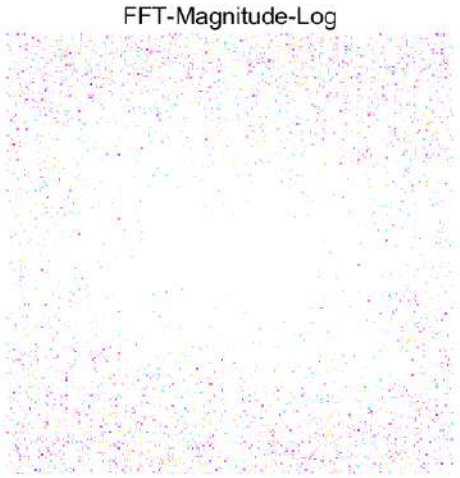
```
% Task 6 – Laplacian filter 적용 영상과 원영상을 적당한 가중치로 합하여 영상 개선
figure(6);
enhancement_lena = zeros([256 256 3]); % 개선된 영상을 담을 빈 행렬 생성
for i=1:256
  for j=1:256
    for k=1:3
      enhancement_lena(i,j,k) = B_laplacian(i,j,k)*0.4 + lena_rgb(i,j,k)*0.6;
    end
  end
end
imshow(enhancement_lena), title('개선된 이미지');
```

- Task 7

```
%[R,G,B] 각 component 별로 아래의 task 수행.
% Task 7 – 이차원 Fourier 변환 (dc 가 가운데 위치)
% 이차원 Fourier 변환으로 fft2 사용
% 이차원 matrix 에서 dc 를 가운데로 옮기는 함수도 fftshift (1차원과 동일)
R = fft2(lena_rgb(:,:,1));
G = fft2(lena_rgb(:,:,2));
B = fft2(lena_rgb(:,:,3));
fftFiltered = cat(3,R,G,B);
shiftedFFT = fftshift(fftFiltered);
```

- Task 8

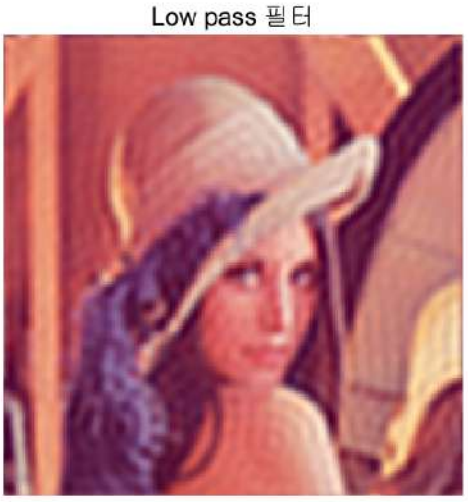
```
% Task 8 – Fourier 변환 결과의 magnitude 를 log scale 로 display
figure(8);
magnituded = abs(shiftedFFT);
logScaled = log(1+abs(magnituded));
imshow(logScaled,[]), title('FFT-Magnitude-Log');
```



- Task 9

```
% Task 9 – 주파수 영역에서 low pass filter (임의로 설정), Fourier 역변환을 통한 filtered
영상 디스플레이
figure(9);
M = 256;
N = 256;
D0 = 30;
% 필터 생성
u = 0:(M-1);
idx = find(u>M/2);
u(idx) = u(idx)-M;
v = 0:(N-1);
idy = find(v>N/2);
v(idy) = v(idy)-N;
[V, U] = meshgrid(v, u);
D = sqrt(U.^2+V.^2);
```

```
% 필터 적용
H = double(D <= D0);
G = H.*fftFiltered;
lowpass_filtered = real(ifft2(double(G))); % 푸리에 역변환
imshow(lowpass_filtered, [ ]), title('Low pass 필터');
```

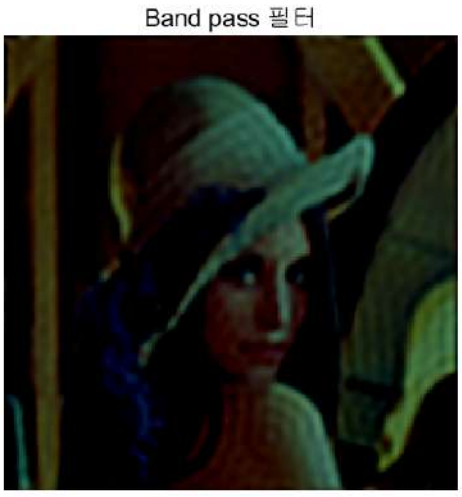


- Task 10

```
% Task 10 - 주파수 영역에서 high pass filter (임의로 설정), Fourier 역변환을 통한 filtered 영상 디스플레이
figure(10);
D1 = 0;
H = double(D > D1);
G = H.*fftFiltered;
highpass_filtered = real(ifft2(double(G)));
imshow(highpass_filtered, [ ]), title("High pass 필터");
```

- Task 11

```
% Task 11 - 주파수 영역에서 band pass filter (임의로 설정), Fourier 역변환을 통한 filtered 영상 디스플레이
figure(11);
% 저주파 고주파 제외하고 pass
H1 = double(D <= D0);
H2 = double(D > D1);
G1 = H1.*fftFiltered;
G2 = H2.*G1;
bandpass_filtered = real(ifft2(double(G2)));
imshow(bandpass_filtered, [ ]), title("Band pass 필터");
```

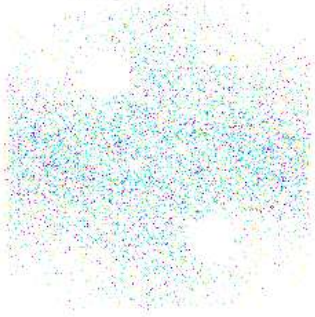


- Task 12

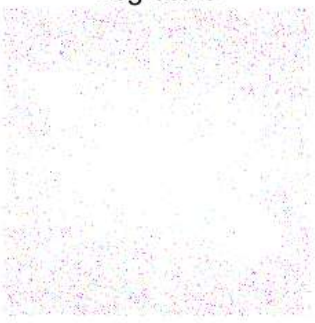
```
% Task 12 –특정 주파수 잡음에 의해 왜곡된 영상을 복원하는 프로그램 작성.
%Hint (Task 12)
% (1) 왜곡된 영상을 Fourier 변환하여 특정 영역들의 노이즈를 파악.
% (2) 특정 영역들의 noise 들을 제거.
%     특정 영역들을 (i) 0 으로 처리하거나, (ii) 인접한 주파수 값들로 처리
% (3) Inverse Fourier 변환으로 영상 복원.
```

```
clc;
clear;
% 주파수 잡음에 왜곡된 영상 불러오기
fid2 = fopen('noised_lena.raw','r');
noised_lena_raw  = fread(fid2, 'float64');
fclose(fid2);
noised_lena = reshape(noised_lena_raw,[256,256,3]);
figure(12);
% 노이즈 파악
noisedlenaF = fft2(noised_lena);
noisedlenaMag = abs(noisedlenaF);
noisedlenaFCenter = fftshift(noisedlenaF);
nosiedlenaLog = log(1+abs(noisedlenaFCenter));
subplot(2,2,1);
imshow(log(noisedlenaMag),[]), title('Magnitude-FT');
subplot(2,2,2);
imshow(nosiedlenaLog,[]), title('Log scale');
% 비교를 위해 왜곡된 영상 Display
subplot(2,2,3);
imshow(noised_lena), title('왜곡된 Lena')
% 인접한 주파수 값으로 처리
subplot(2,2,4)
M = 256;
N = 256;
u = 0:(M-1);
idx = find(u>M/2);
u(idx) = u(idx)-M;
v = 0:(N-1);
idy = find(v>N/2);
v(idy) = v(idy)-N;
[V, U] = meshgrid(v, u);
D = sqrt(U.^2+V.^2);
D0 = 60;
H = double(D <= D0);
G = H.*noisedlenaF;
% Inverse Fourier 변환으로 영상 복원
lowpass_filtered = real(ifft2(double(G)));
imshow(lowpass_filtered, [ ]), title("Low pass 적용");
```

Magnitude-FT

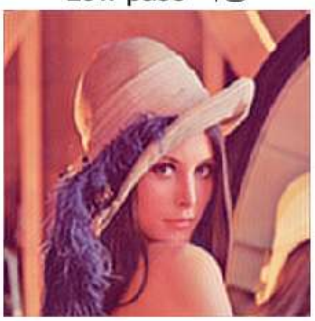


Log scale



왜곡된 Lena

Low pass 적용



- Task 13

```
% Task 13 – 흑/백 마스크 및 마스크를 쓰지 않은 자신의 영상에서의 흑색/백색 마스크를 착용하였는지 여부를 판단하는 프로그램 작성. (자신의 사진은 가급적 256x256 크기 영상으로 변환할 것., color 또는 gray scale 영상은 임의로 선택할 것)
% Hint (Task 13)
% 일정한 크기의 (MxM) 평탄한 kernel을 원 영상과 convolution 하여, 출력 값이 threshold1 이상인 pixel 에는 1, 이하인 pixel 에는 0을 할당.
% (2) Thresholding한 영상에서 1로 할당한 영역이 마스크 영역과 일치하는지를 확인.
%     필요시 kernel 크기와 threshold1 값을 조절.
% (3) Thresholding한 영상에서 pixel 의 값이 1 인 pixel의 개수를 counting 하여  적당한 값(threshold 2) 이상이면 마스크를 착용한 것으로 판정.
% (4) (1)-(3) 과정을 변형하여 마스크의 색깔을 구분할 수 있는 프로그램 작성
% (5) 주어진 영상들 모두에서 잘 동작하는 kernel의 크기, threshold 1 값, threshold 2 값과 이들로 얻은 Threshold 영상, 원영상들을 reporting.
clc;
clear;
% 마스크 이미지 불러오기
whitemask_origin = imread('whitemask.jpeg', 'jpeg'); % 흰색 마스크 이미지
whitemask = imresize(whitemask_origin, [256 256]); % 사이즈 256x256으로 조정
blackmask_origin = imread('blackmask.jpeg', 'jpeg'); % 검정 마스크 이미지
blackmask = imresize(blackmask_origin, [256 256]); % 사이즈 256x256으로 조정
nomask_origin = imread('nomask.jpeg', 'jpeg'); % 마스크 안 쓴 이미지
nomask = imresize(nomask_origin, [256 256]); % 사이즈 256x256으로 조정
% 마스크 사진 Display
```

```
figure(13);
subplot(2,2,1);
% 흰색 마스크 테스트
imshow(whitemask), title('원영상');
mask_gray = rgb2gray(whitemask); % 회색조로 변경 - test할 사진 입력
% 검정 마스크 테스트
%imshow(blackmask), title('원영상');
%mask_gray = rgb2gray(blackmask); % 회색조로 변경 - test할 사진
% 노마스크 테스트
%imshow(nomask), title('원영상');
%mask_gray = rgb2gray(nomask); % 회색조로 변경 - test할 사진 입력
% convolution 적용
windowSize = 7;
kernel = ones(windowSize) / windowSize ^ 2;
conv_img = conv2(mask_gray, kernel, 'same');
subplot(2, 2, 2), imshow(conv_img, []), title('7x7 커널');
% Thresholding 적용
threshold1 = 200; % 200 이상이면 pixel 값 1로 변경
for i=1:256
  for j=1:256
     if(conv_img(i,j) >= threshold1)
        conv_img(i,j) = 1;
     else
        conv_img(i,j) = 0;
     end
  end
end
subplot(2, 2, 3), imshow(conv_img, []), title('Thresholding 적용');
% 1이상인 pixel 개수 counting
threshold2 = 0;
for i=1:256
  for j=1:256
     if(conv_img(i,j) >= 1)
        threshold2 = threshold2 + 1;
     end
  end
end
% Threshold2 12000개 이상이면 흰색 마스크 착용했다고 판단
if(threshold2 >= 12000)
  fprintf('흰색 마스크를 착용했습니다!');
% Threshold2 8000개 이상이면 검정 마스크 착용했다고 판단
elseif(threshold2 >= 8000)
  fprintf('검정 마스크를 착용했습니다!');
else
  fprintf('마스크를 착용하지 않았습니다.')
end
```

원영상

7x7 커널

Thresholding 적용

흰색 마스크를 착용한 경우

원영상

7x7 커널

Thresholding 적용

검정 마스크를 착용한 경우

원영상

7x7 커널

Thresholding 적용

마스크를 착용하지 않은 경우