CSE185
Introduction to Computer Vision
Lab 04: Frequency Domain Operation

Instructor: Daniel Leung
TA: Mohammadkazem Ebrahimpour
Xueqing Deng

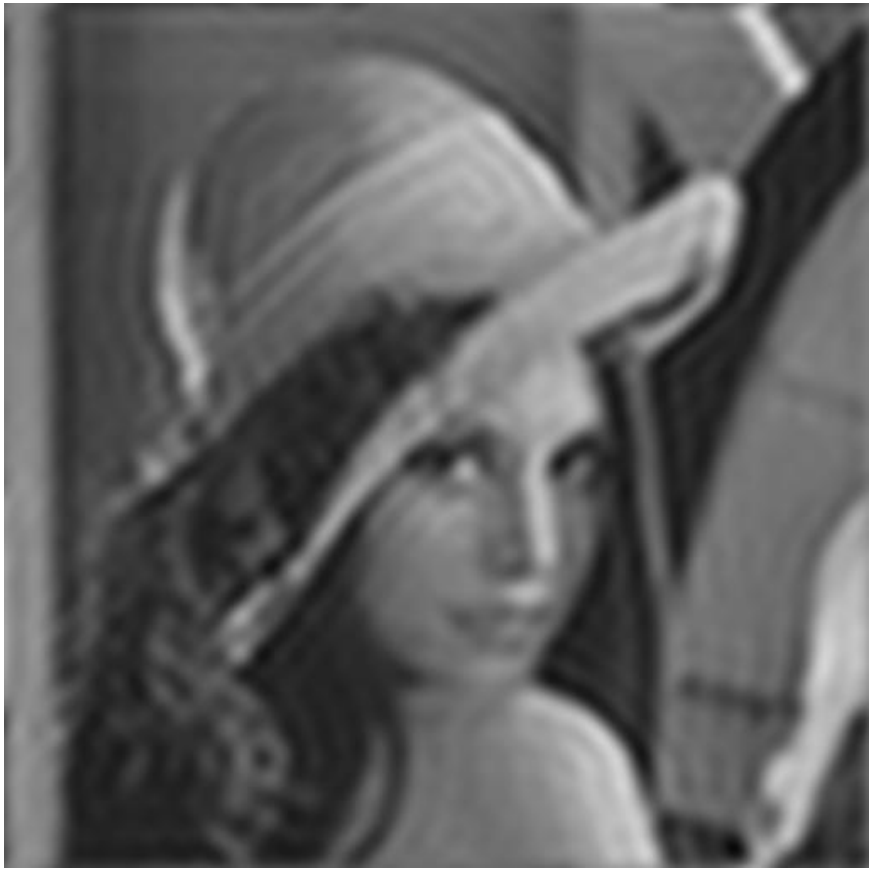
Overview

- Task 1: Splitting low-frequency and high-frequency

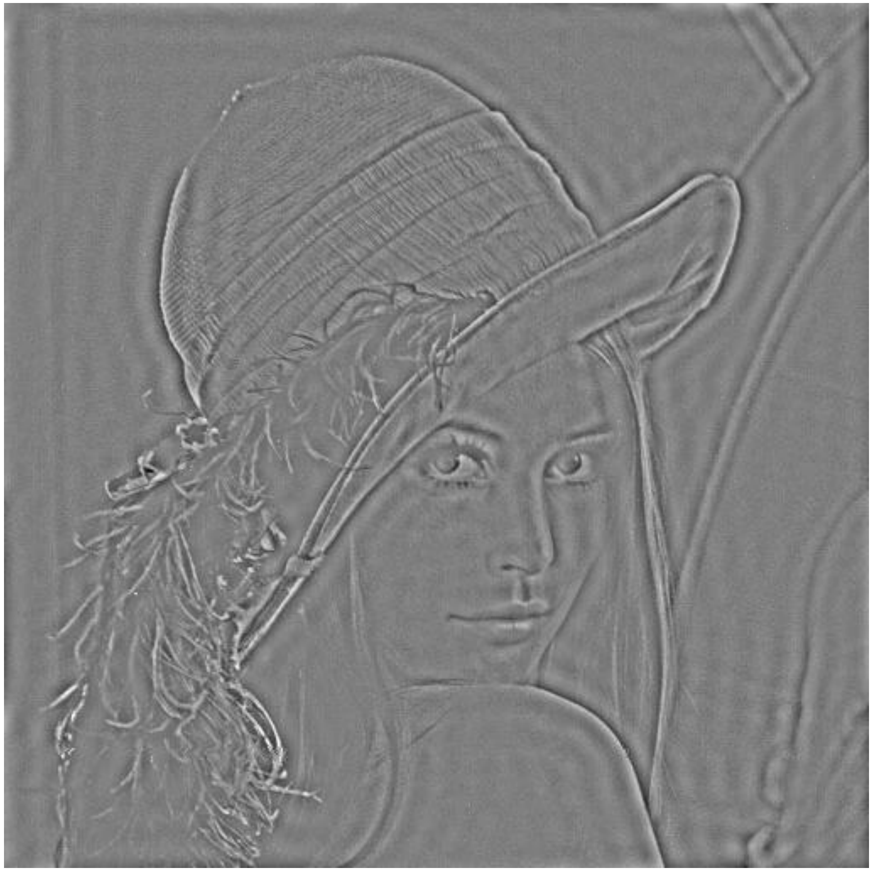
Input Image

Overview

- Task 1: Splitting low-frequency and high-frequency



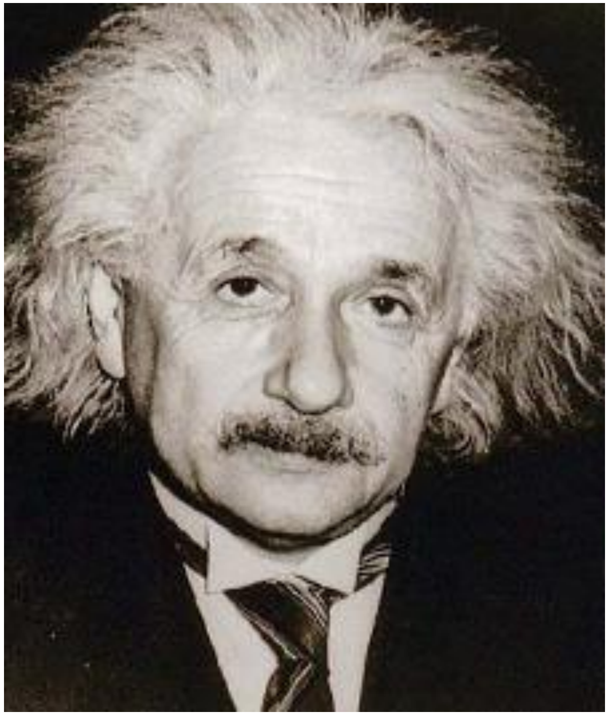
Low-Frequency Part



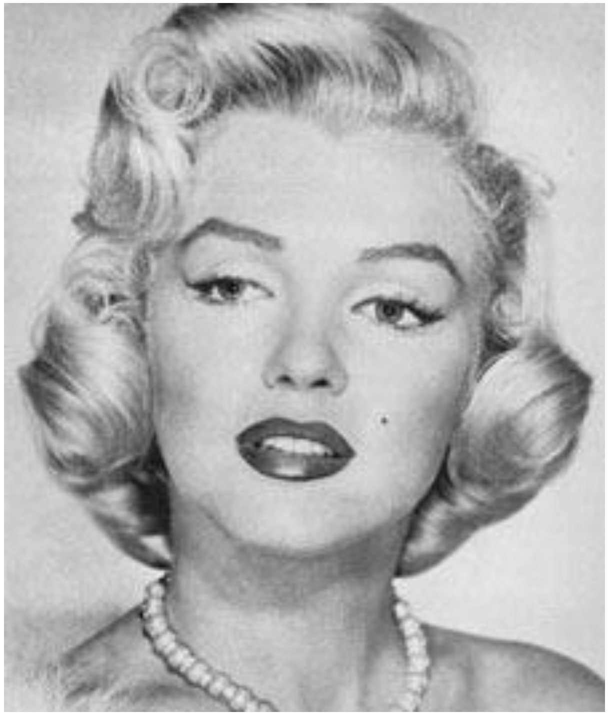
High-Frequency Part

Overview

- Task 2: Hybrid Image



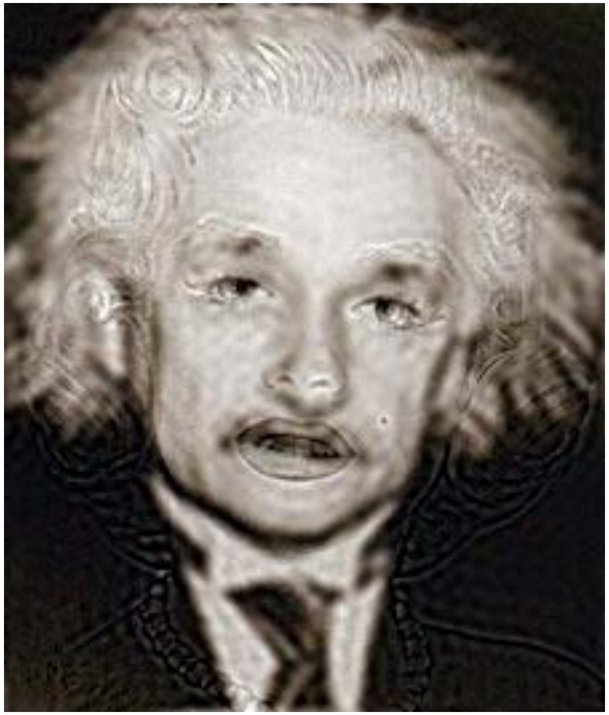
Input 1



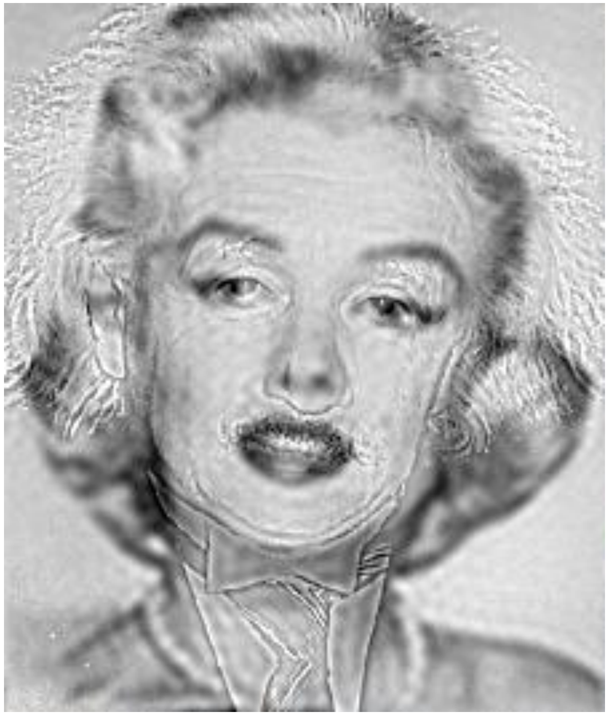
Input 2

Overview

- Task 2: Hybrid Image



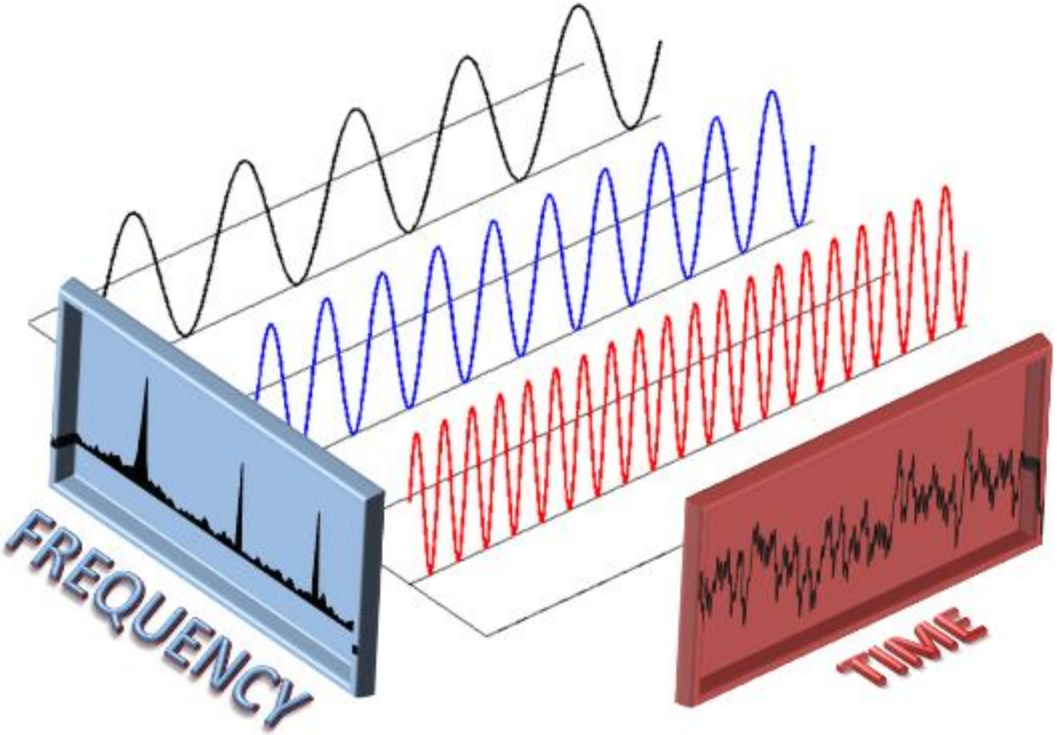
Input 1 low frequency +
Input 2 high frequency



Input 2 low frequency +
Input 1 high frequency

Fourier Transform

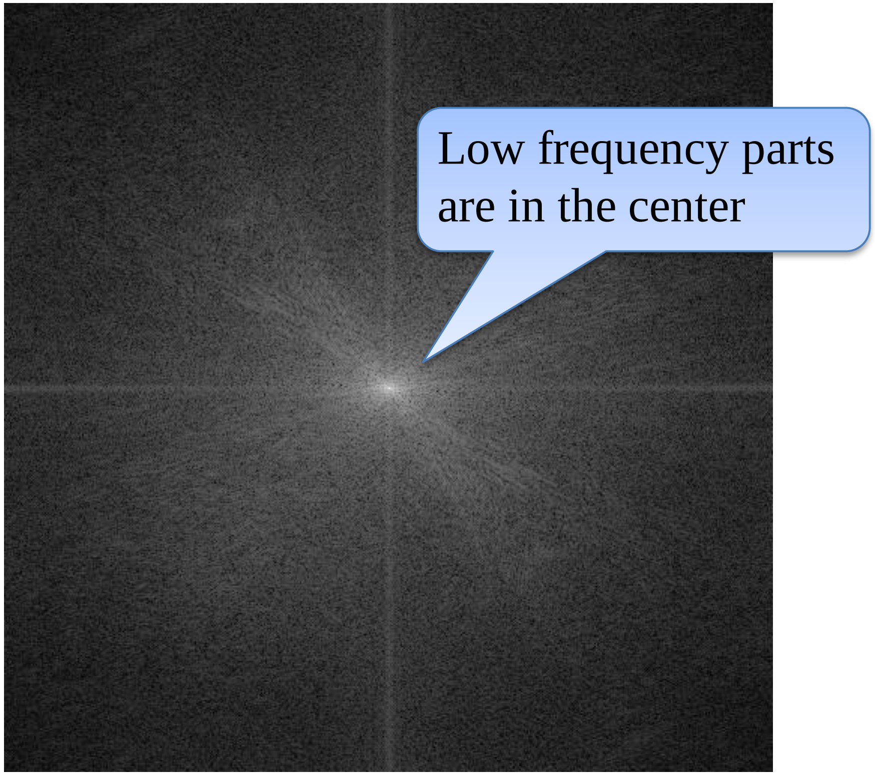
- *Discrete Fourier Transform*: decomposes a signal into its frequency components
- *Fast Fourier Transform* (*FFT*): an algorithm to compute discrete Fourier Transform in $O(N \log N)$ time (Direct method: $O(N^2)$ time)



Fourier Transform

- Apply Fourier Transform to an image:

Input Image



Low frequency parts are in the center

Frequency Map

Fourier Transform in MATLAB

- Step 1: Use `fft2()` to apply fast Fourier Transform:

```
img = im2double(imread(‘images/lena.jpg’));
frequency_map = fft2(img);

figure, imshow( log(abs(frequency_map) + 1), []);
```

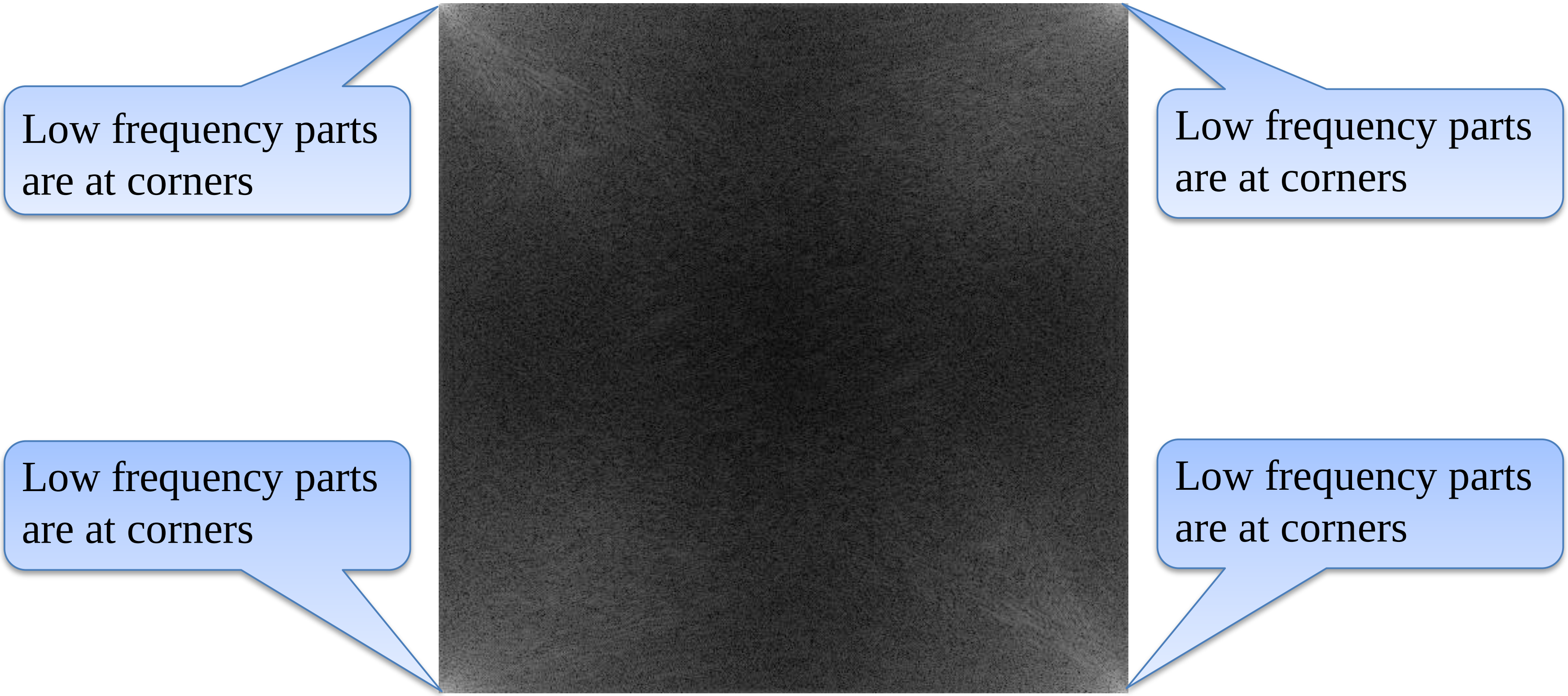
- Display frequency map in MATLAB:
  - `abs()`: take frequency magnitude
  - `log()`: compress range
  - `+1`: avoid log(0)
  - `imshow(x, [])`: auto adjust range by:

$$x = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Fourier Transform in MATLAB

- Step 1: Use `fft2()` to apply fast Fourier Transform:

```
img = im2double(imread(‘images/lena.jpg’));
frequency_map = fft2(img);

figure, imshow( log(abs(frequency_map) + 1), []);
```



Low frequency parts are at corners

Low frequency parts are at corners

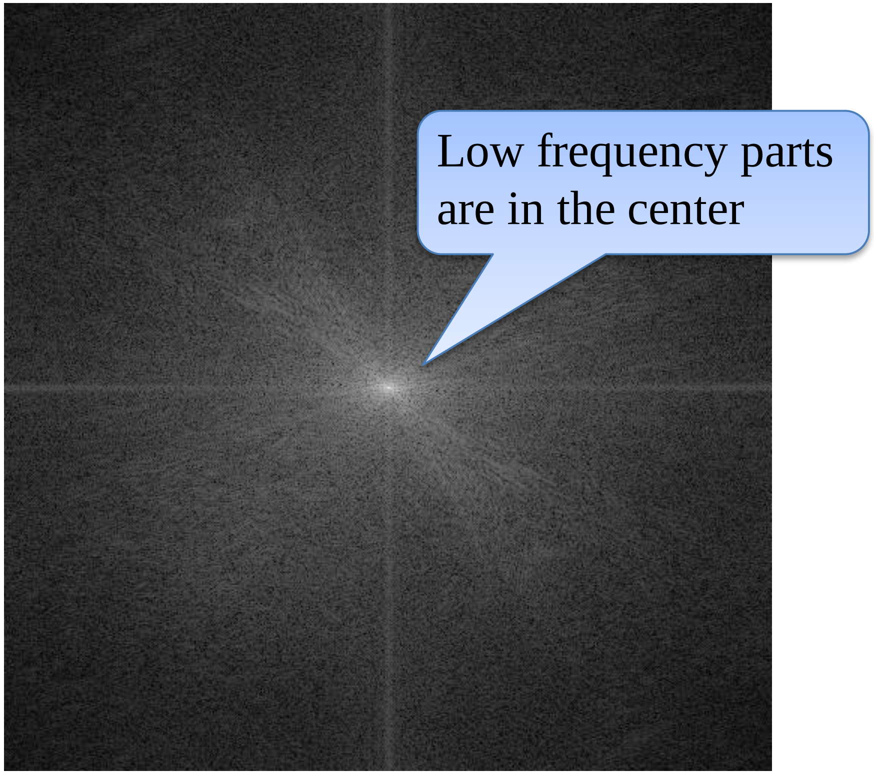
Low frequency parts are at corners

Low frequency parts are at corners

Fourier Transform in MATLAB

- Step 2: Use `fftshift()` to rearrange the frequency map

```
img = im2double(imread(‘images/lena.jpg’));
frequency_map = fft2(img);
frequency_map_shifted = fftshift(frequency_map);
```



Low frequency parts are in the center

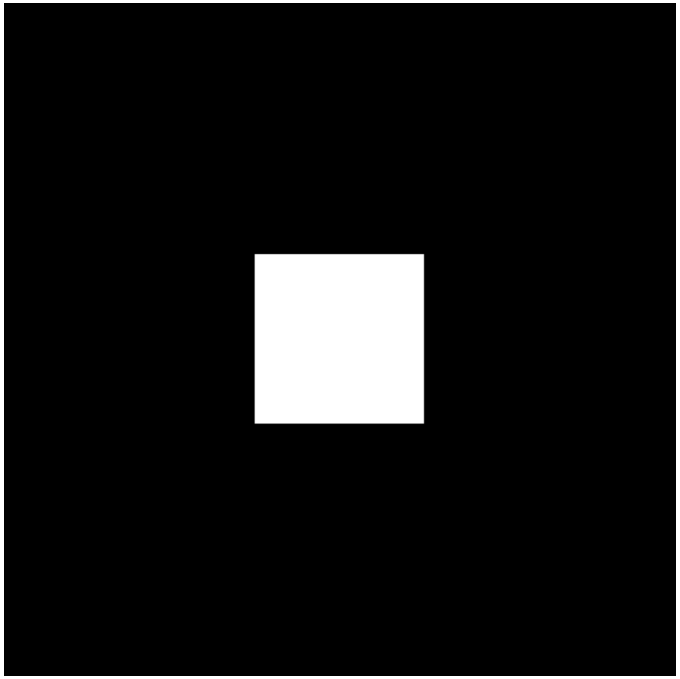
Fourier Transform in MATLAB

- Step 3: compute a low-frequency mask

```
y1 = ???
y2 = ???
x1 = ???
x2 = ???
mask = zeros(size(img));
mask(y1 : y2, x1 : x2, :) = 1;
```

Something related to ratio

Remember the third dimension



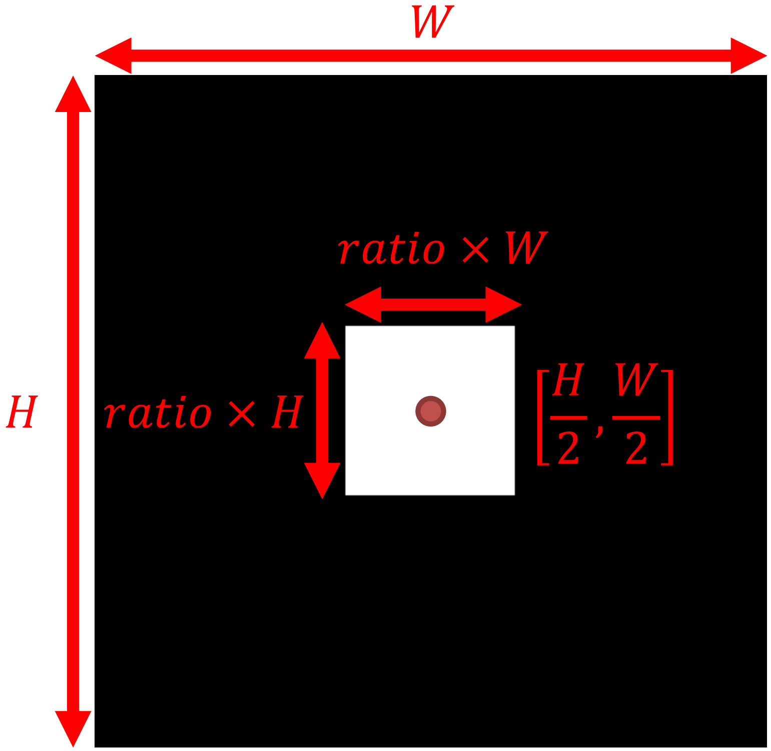
Fourier Transform in MATLAB

- Step 3: compute a low-frequency mask (TODO)

```
y1 = ???
y2 = ???
x1 = ???
x2 = ???
mask = zeros(size(img));
mask(y1 : y2, x1 : x2, :) = 1;
```

Something related to ratio

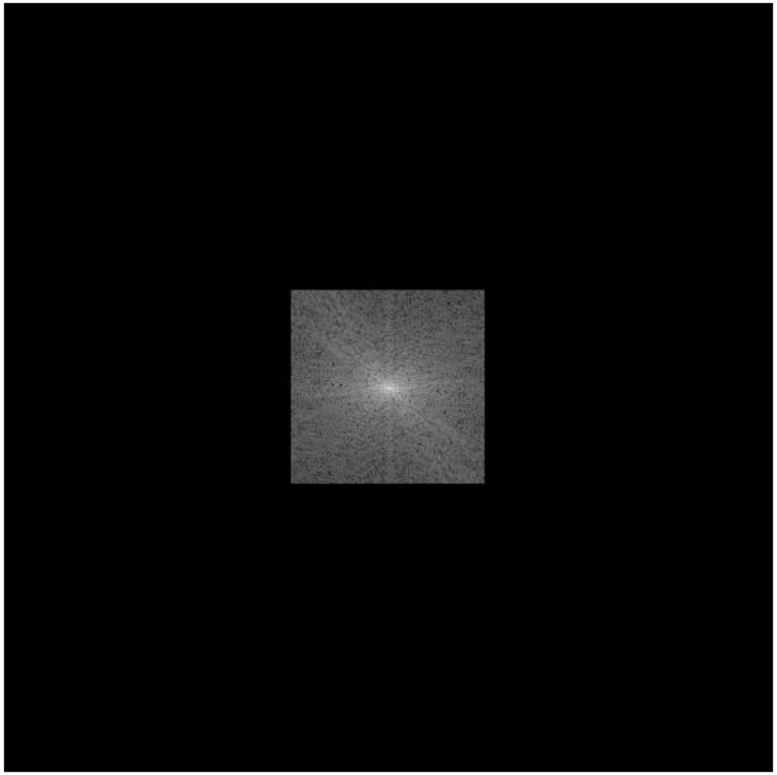
Remember the third dimension

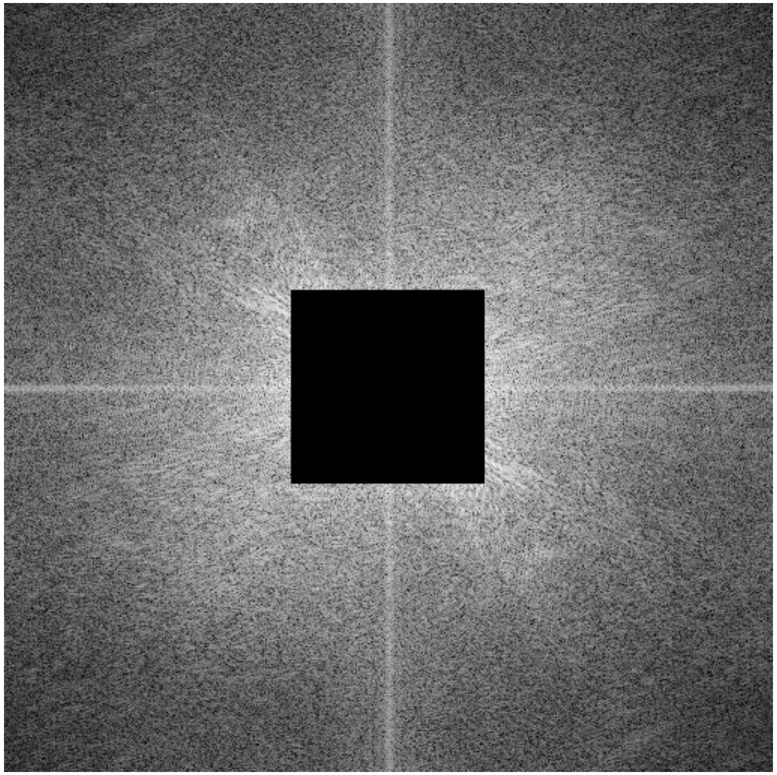


Fourier Transform in MATLAB

- Step 4: Split low-frequency and high-frequency parts

```
low_frequency_map_shifted =
    frequency_map_shifted .* mask;
high_frequency_map_shifted =
    frequency_map_shifted .* (1 - mask);
```

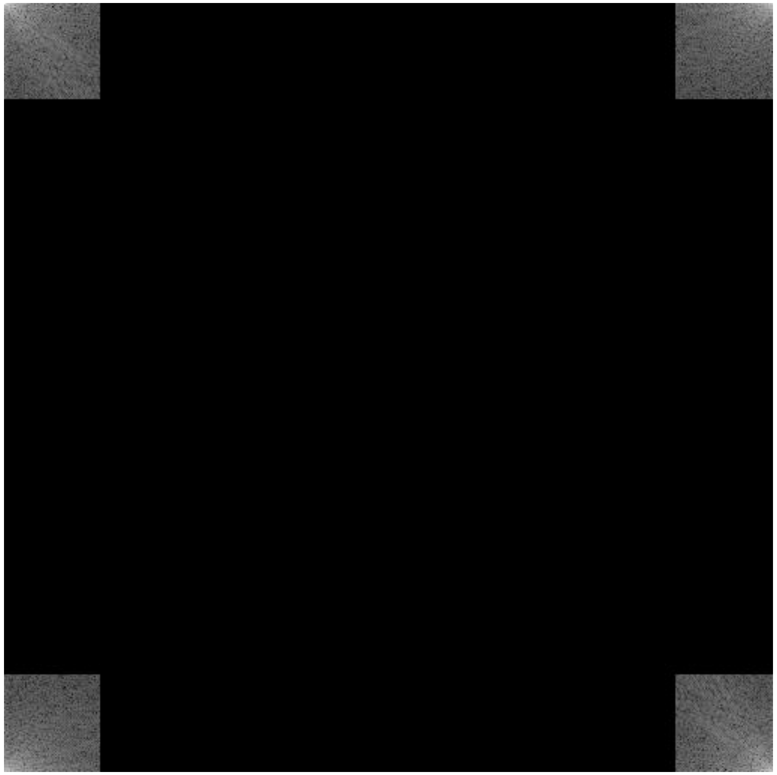


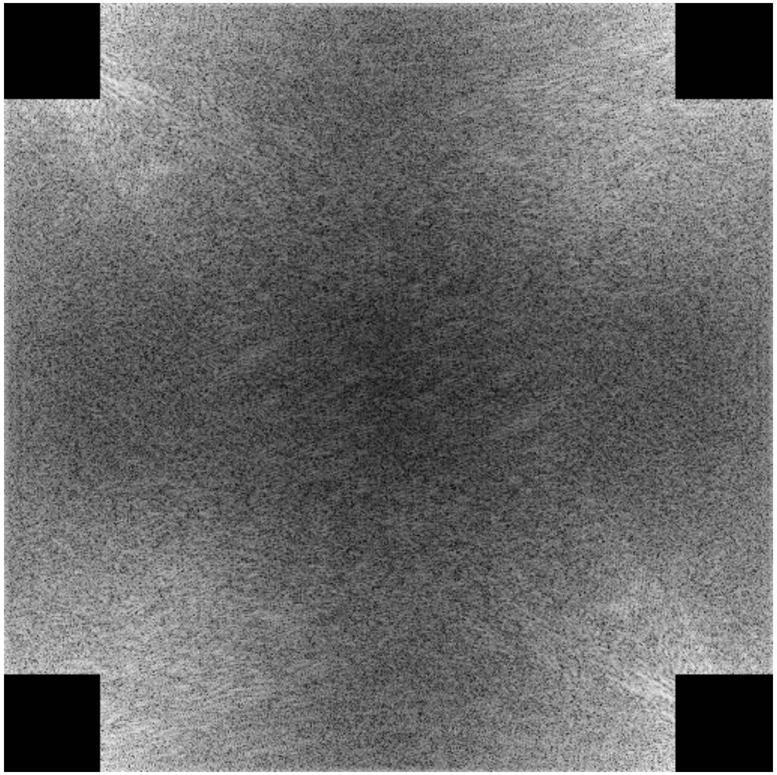


Fourier Transform in MATLAB

- Step 5: Use `ifftshift()` to shift frequency map back

```
low_frequency_map =
    ifftshift(low_frequency_map_shifted);
high_frequency_map =
    ifftshift(high_frequency_map_shifted);
```





Fourier Transform in MATLAB

- Step 6: Use `ifft2()` to convert from frequency domain to image domain

```
low_pass_img = real(ifft2(low_frequency_map));
high_pass_img = real(ifft2(high_frequency_map));
```

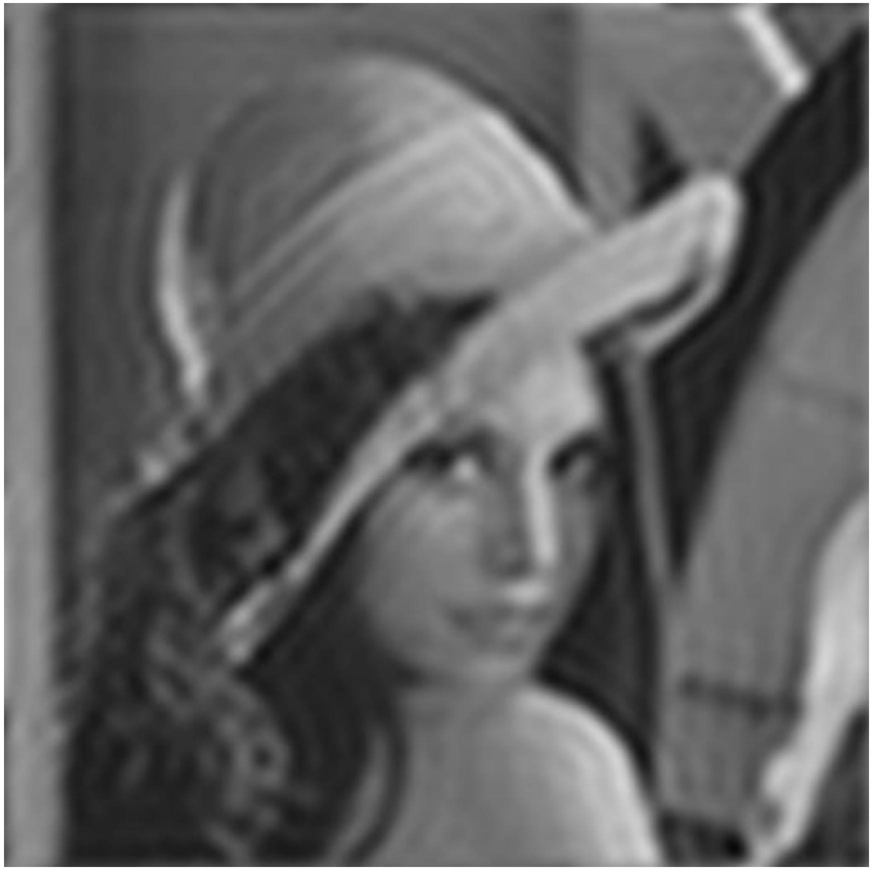
Take the real part only

- To show/save the high-frequency parts, add 0.5 offset for better visualization

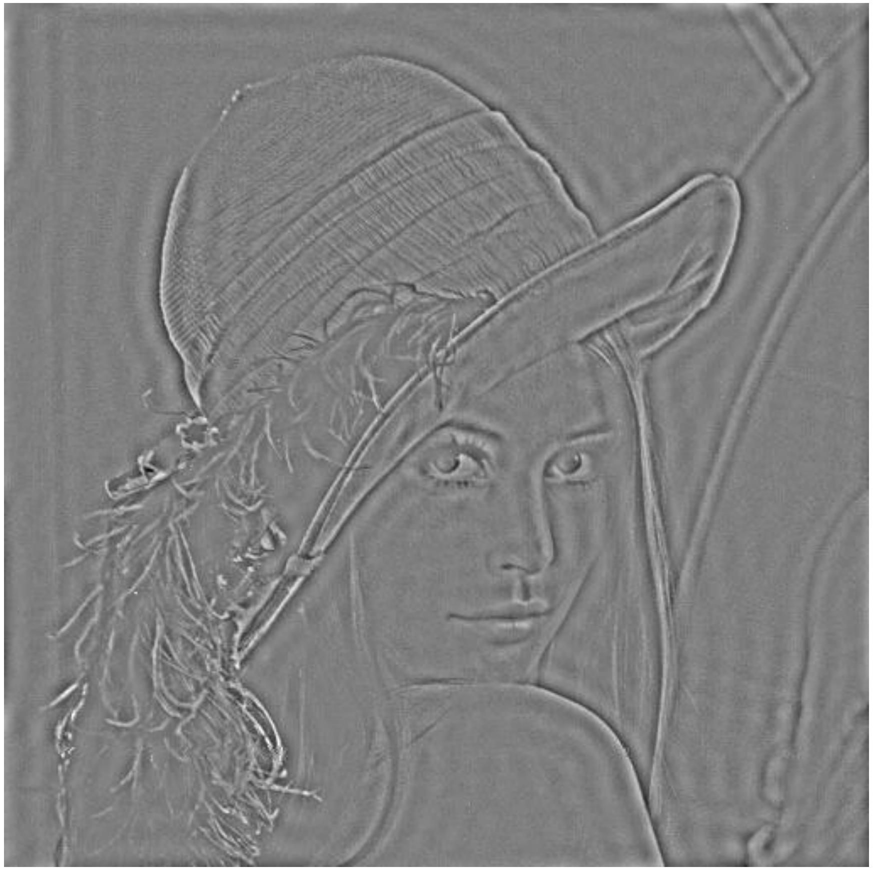
```
figure, imshow(low_pass_img);
figure, imshow(high_pass_img + 0.5);

imwrite(low_pass_img, ‘lena_low_pass.jpg’);
imwrite(high_pass_img + 0.5, ‘lena_high_pass.jpg’);
```

Split Frequency

- Ratio = 0.1



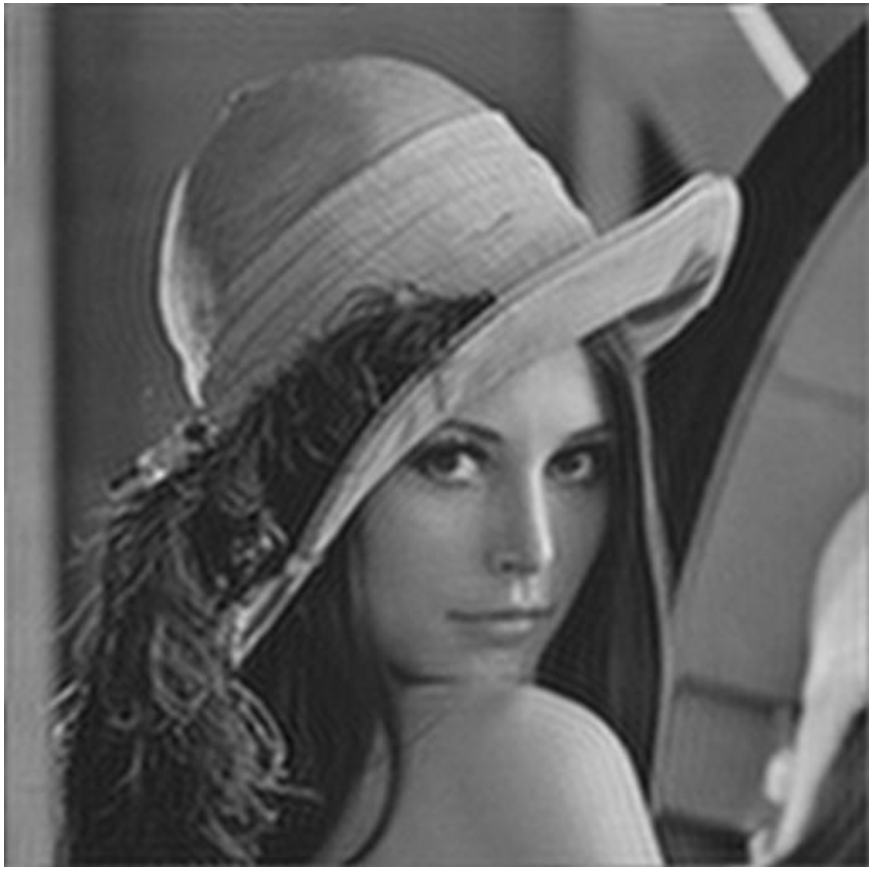
low-frequency part



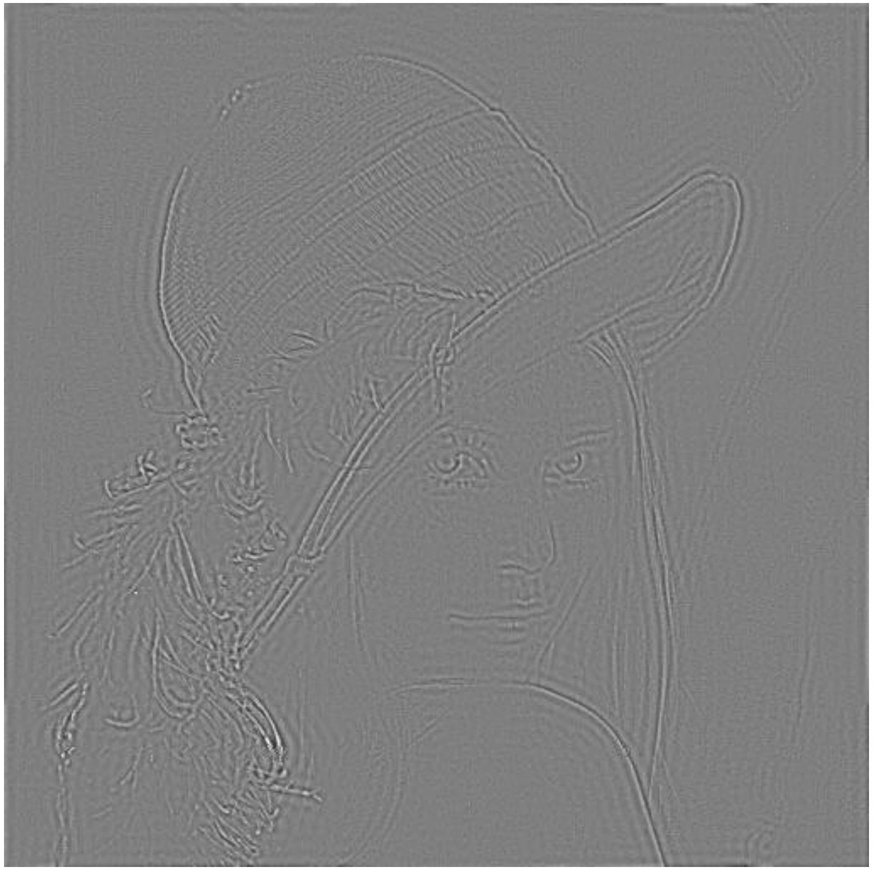
high-frequency part

Split Frequency

- Ratio = 0.25



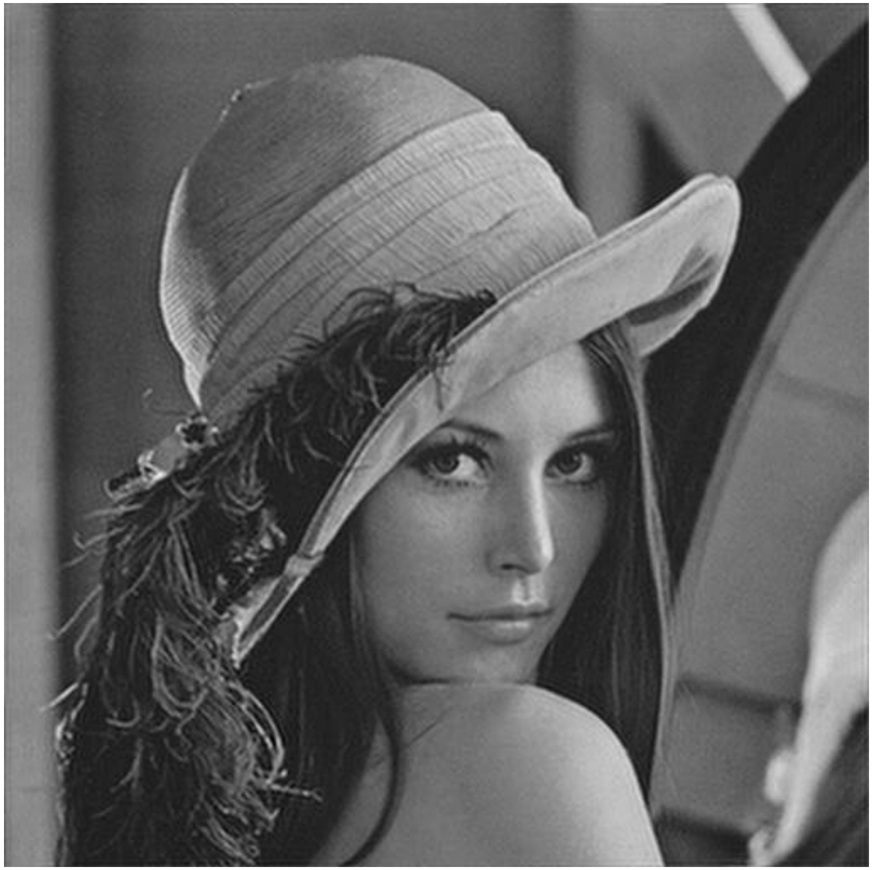
low-frequency part



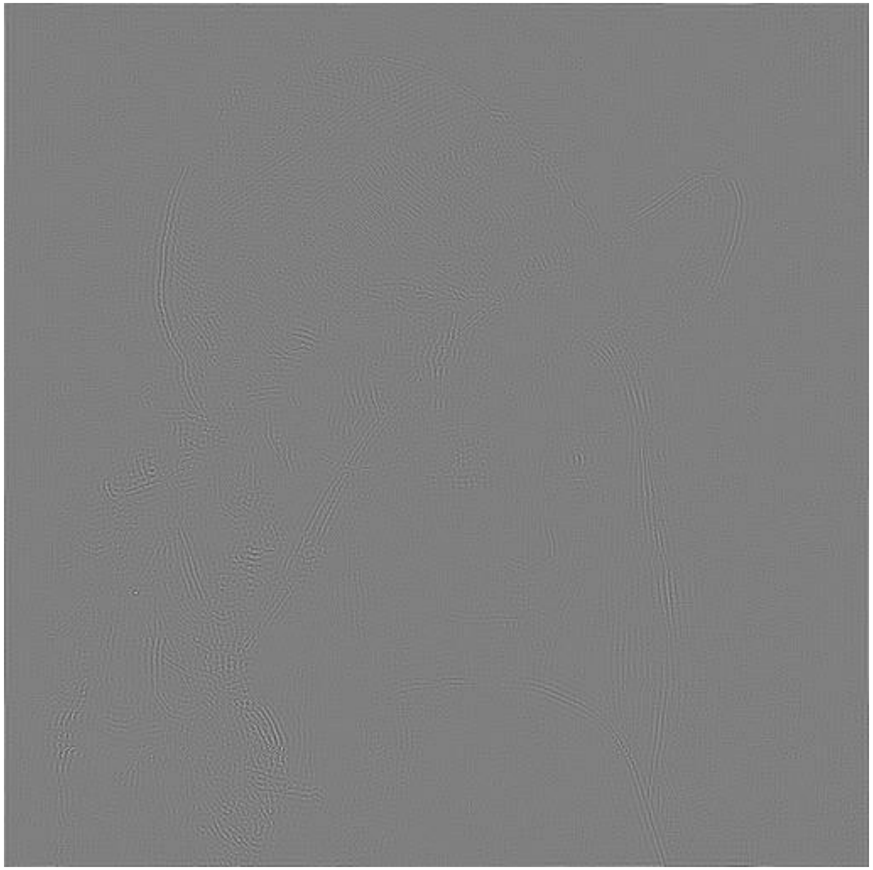
high-frequency part

Split Frequency

- Ratio = 0.5



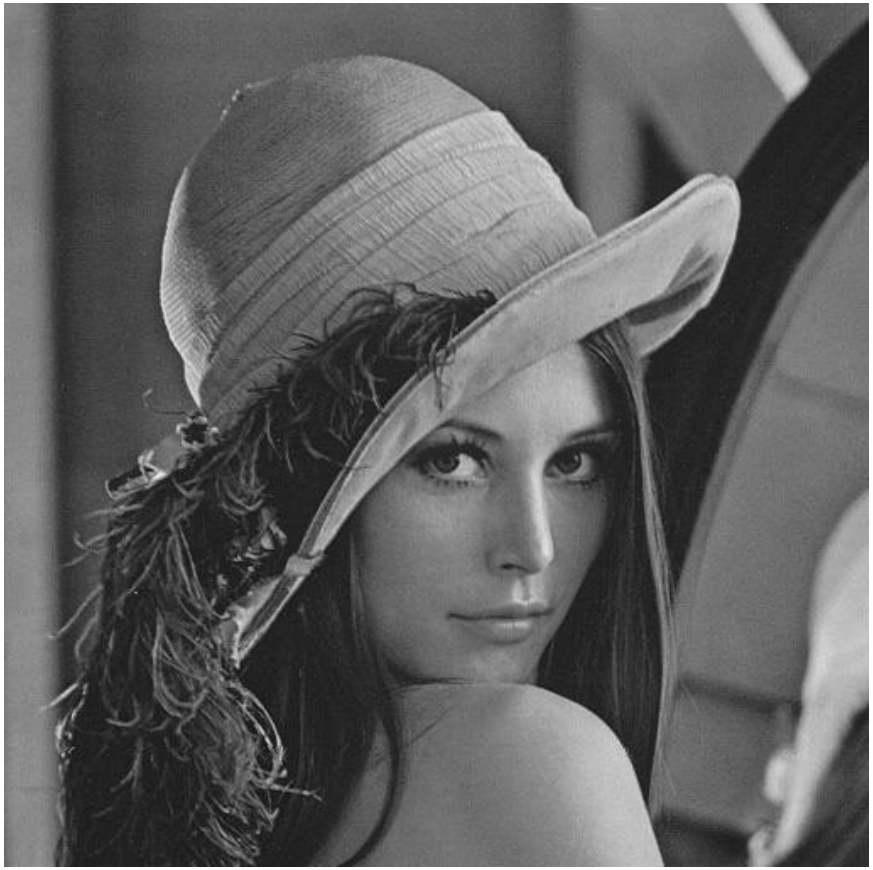
low-frequency part



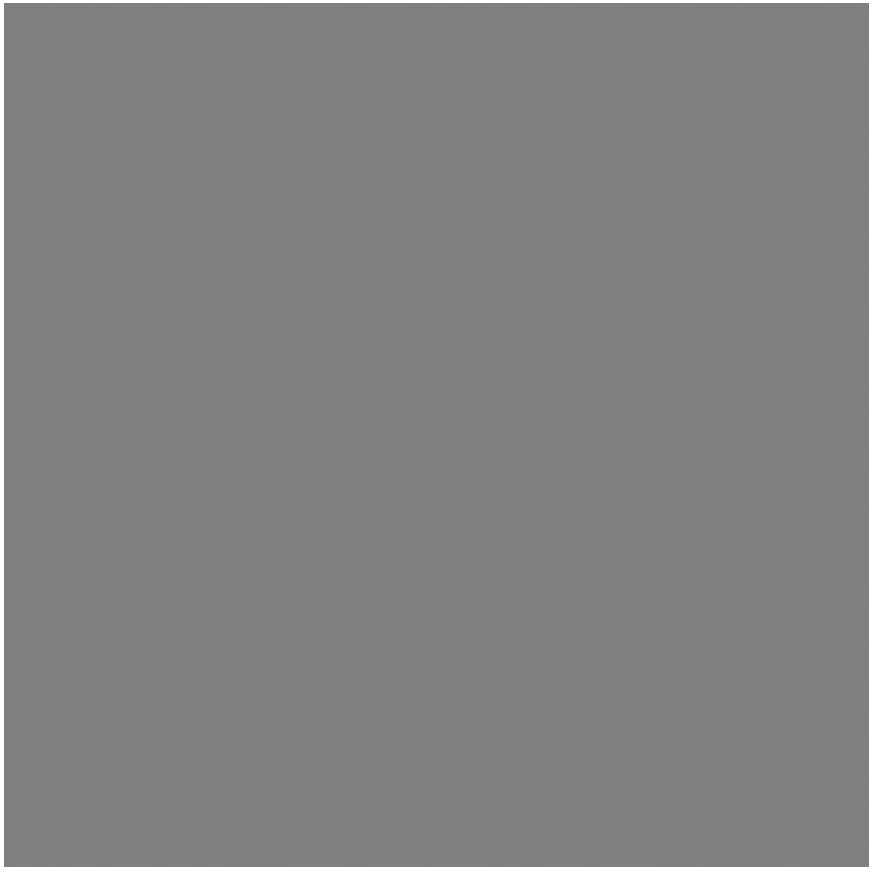
high-frequency part

Split Frequency

- Ratio = 1



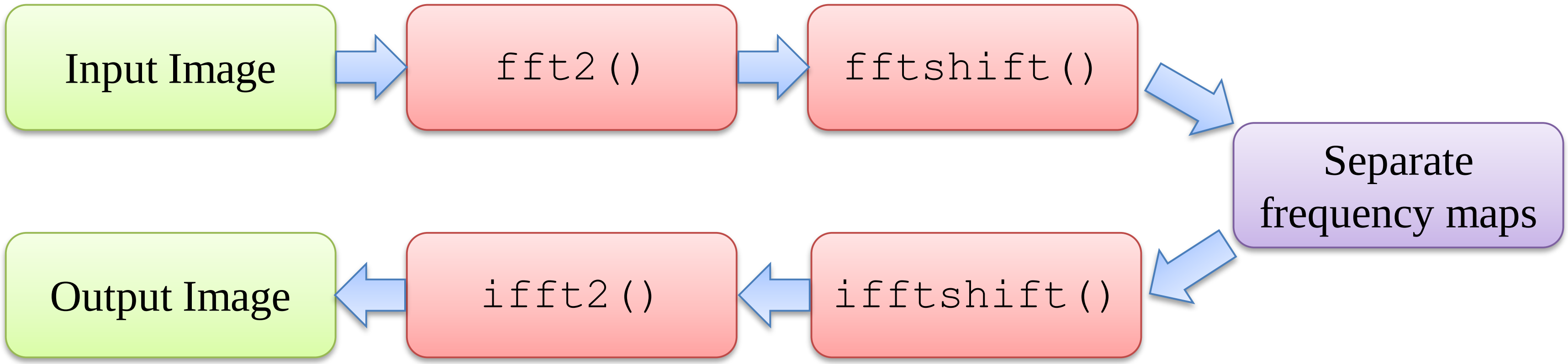
low-frequency part



high-frequency part

Summary of Task 1

1. Apply `fft2()` to input image
2. Use `fftshift()` to shift frequency map
3. Compute low-frequency mask
4. Separate low frequency and high frequency maps
5. Use `ifftshift()` to shift frequency maps back
6. Apply `ifft2()` to convert frequency maps to images



Summary of Task 1

- In `separate_frequency.m`:

```
function [low_pass_img, high_pass_img] =
                        separate_frequency(img, ratio)

    %% apply FFT
    frequency_map = fft2(img);

    %% shift the frequency map

    %% compute low-frequency mask

    %% separate low-frequency and high-frequency maps

    %% shift frequency maps back

    %% apply Inverse FFT
    low_pass_img = real(ifft2(frequency_map));
    high_pass_img = real(ifft2(frequency_map));
end
```

Replace frequency map with your low/high frequency maps

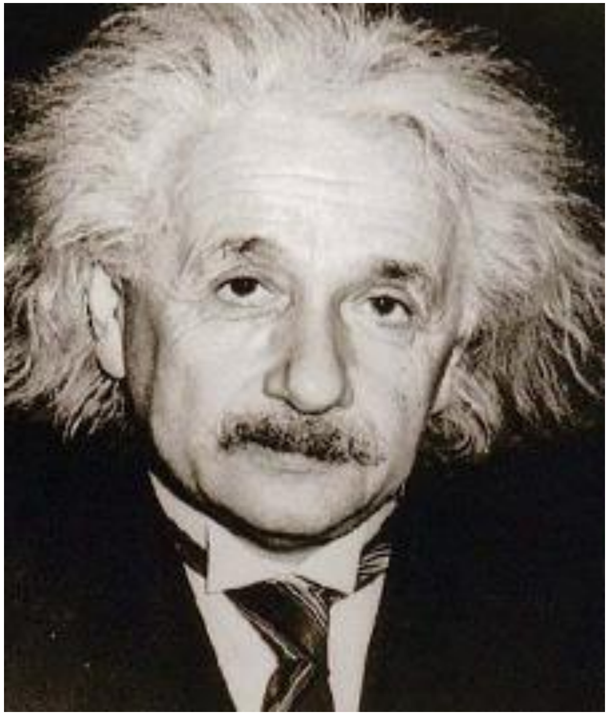
Summary of Task 1

- Create lab04.m:

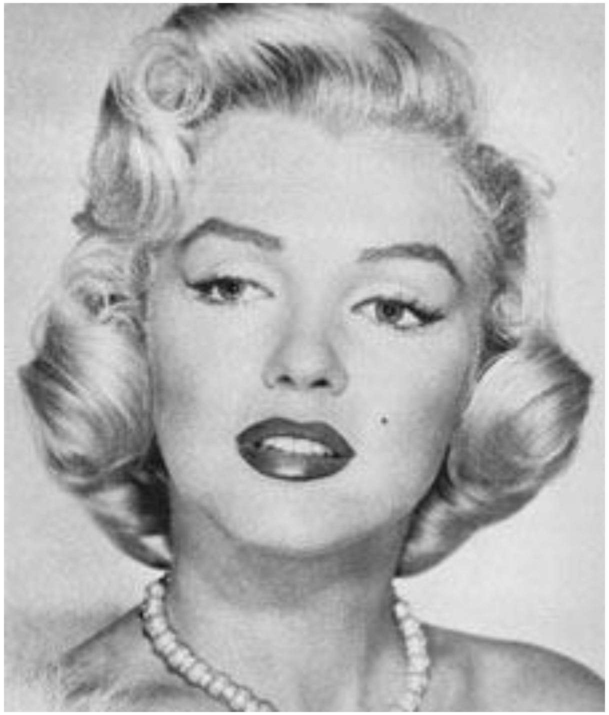
```
%% Task 1: Split Frequency
img = im2double(imread('images/cameraman.jpg'));

ratio = 0.1;

[low_pass_img, high_pass_img] =
                   separate_frequency(img, ratio);

imwrite(low_pass_img, ‘cameraman_low_pass.jpg');
imwrite(high_pass_img + 0.5,
‘cameraman_high_pass.jpg');
```

Hybrid Image

- Take 2 input images



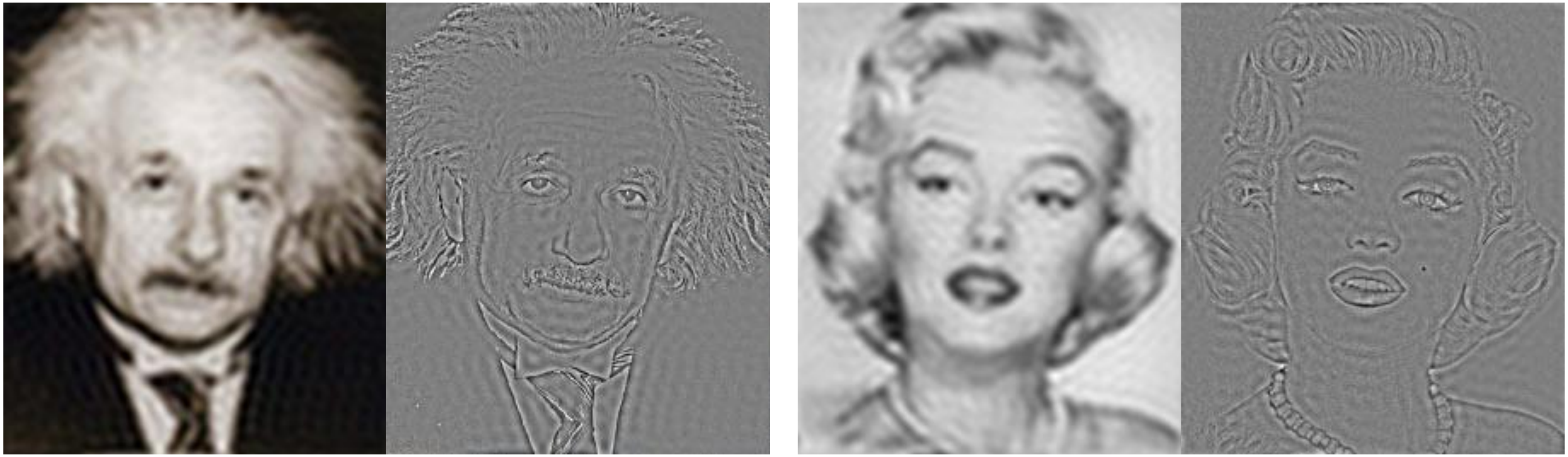
Input 1



Input 2

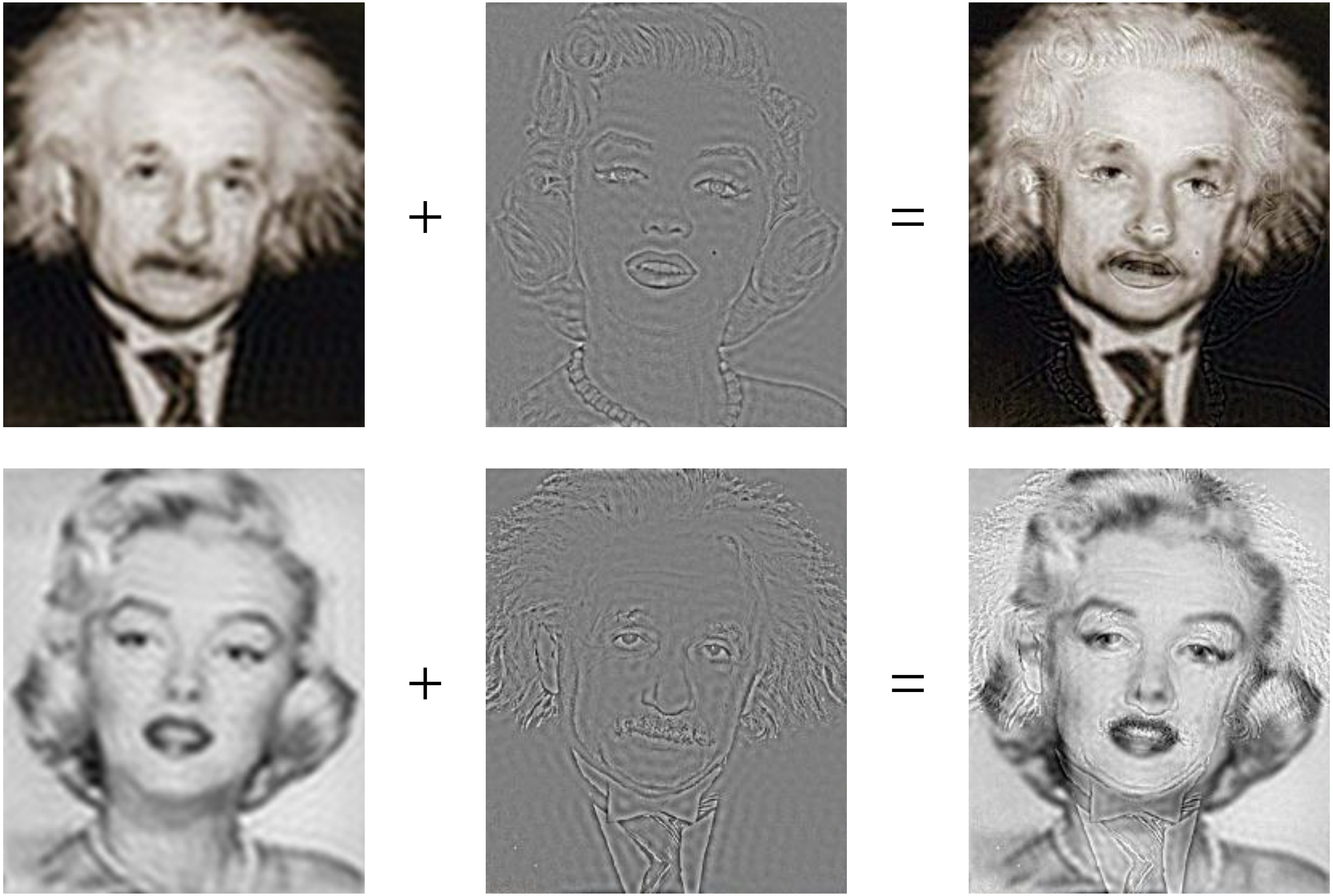
Hybrid Image

- Splitting into low/high-pass images



Hybrid Image

- Merge low and high frequency images



Hybrid Image

- Make use of your function `separate_frequency.m` in `hybrid_image.m`:

```
function img_merged = hybrid_image(img1, img2, ratio)

    %% split img1 and img2 into low/high frequency maps

    %% combine the low-frequency map of img1 with the high-frequency map of img2
    img_merged = ???;

end
```

Hybrid Image

- In lab04.m:

```
%% Task 2: Hybrid Image
name1 = 'images/cat.jpg';
name2 = 'images/dog.jpg';

img1 = im2double(imread(name1));
img2 = im2double(imread(name2));

ratio = 0.2;
img_merged = hybrid_image(img1, img2, ratio);

figure, imshow(img_merged);
```

Try other image pairs

Try other ratio

TODO

1. Implement separate_frequency.m
2. Use ratio = 0.1, and save the image as cameraman_low_0.1.jpg and cameraman_high_0.1.jpg
3. Use ratio = 0.2, and save the image as cameraman_low_0.2.jpg and cameraman_high_0.2.jpg
4. Implement hybrid_image.m
5. Use any ratio to merge the low-frequency of cat.jpg and the high frequency of dog.jpg, and save the image as hybrid_1.jpg
6. Use any ratio to merge the low-frequency of dog.jpg and the high frequency of cat.jpg, and save the image as hybrid_2.jpg
7. Upload your output images and separate_frequency.m, hybrid_image.m, and lab04.m

One more thing…

- Display gray-scale frequency map:

```
figure, imshow(log(abs(frequency_map) + 1), []);
```

- Display color frequency map:

```
figure, imagesc(log(abs(frequency_map) + 1)), colormap jet;
```

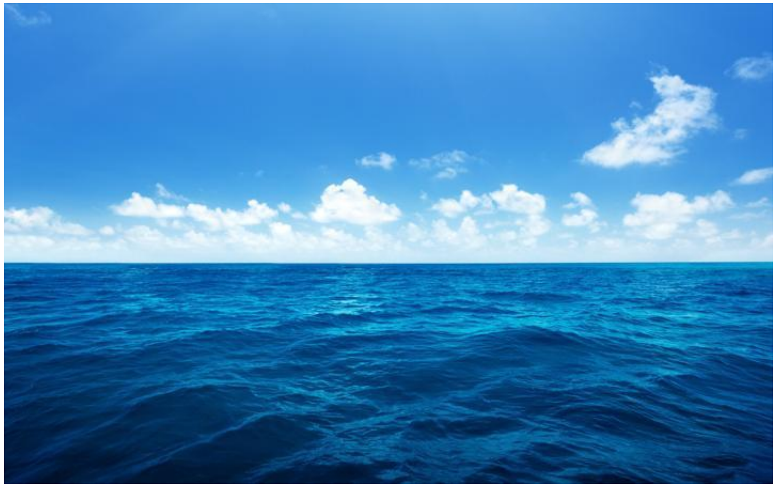
Should be single channel

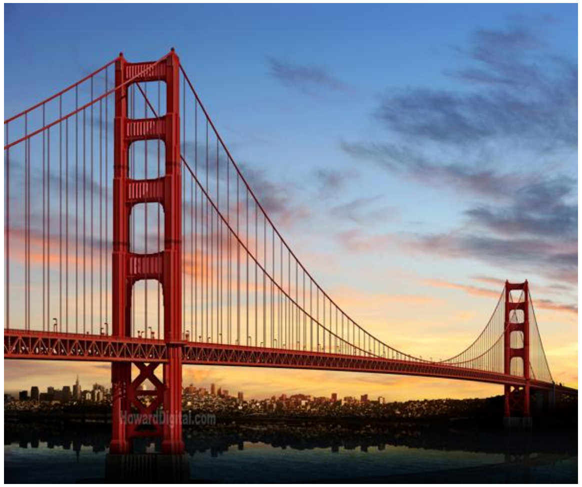
- Try to display the frequency of square.bmp, gaussian.bmp, sea.bmp, and bridge.bmp

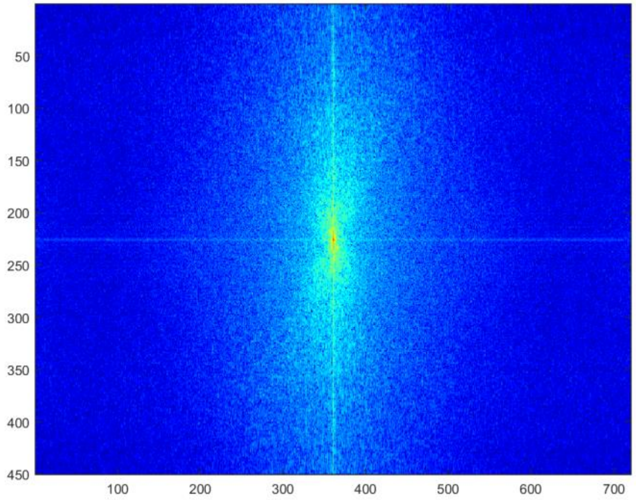
```
img = im2double(imread(‘images/square.bmp’));
frequency_map = fftshift(fft2(img));
figure, imagesc(log(abs(frequency_map)+ 1)), colormap jet;
```

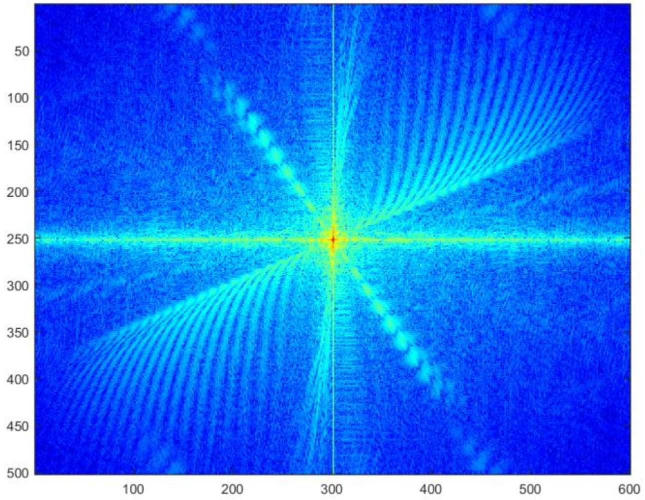
One more thing…

- More horizontal edges ⇒ higher response on vertical direction



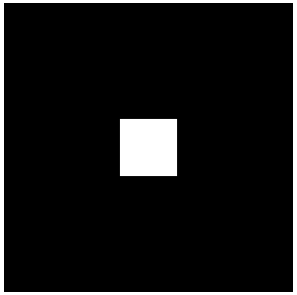


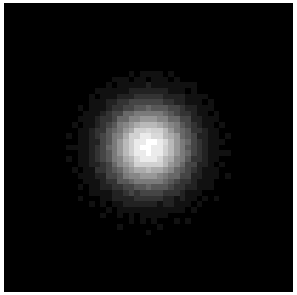


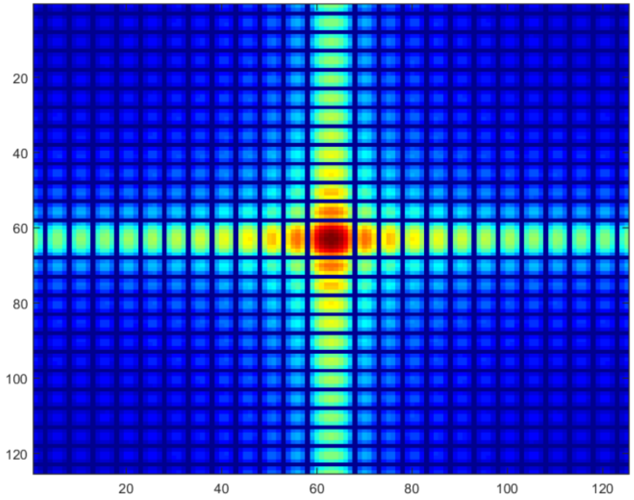


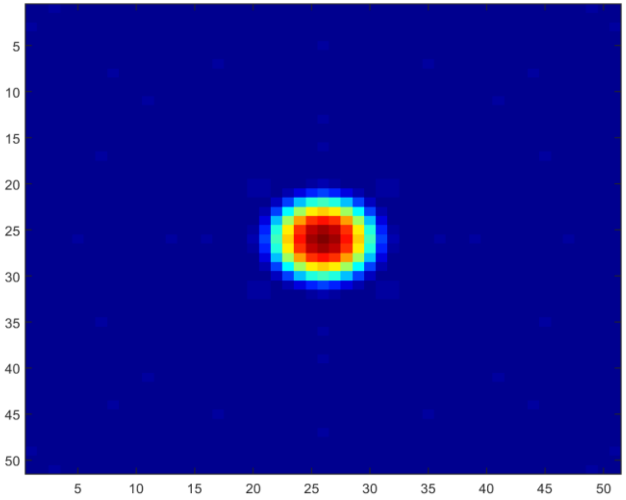
One more thing…

- The Fourier transform of a box signal is a sinc function
- The Fourier transform of a Gaussian signal is still a Gaussian function






20
40
60
80
100
120
20
40
60
80
100
120


5
10
15
20
25
30
35
40
45
50
5
10
15
20
25
30
35
40
45
50