Illinois Institute of Technology

ECE 565 Multidimensional Digital Signal Processing
(Fall '98)

Project 1: Fourier Representation of Images

11/10/1998

Jovan Brankov

Step by step procedure with results

1. In writing the routine *fft6464* for 2D-DFT, for block size 64x64, I used the theory from the textbook on pg.75, topic 2.3.2. Row-Column Decomposition equations (2.73a-b). From these equations it can be seen that we can compute a 2D-DFT by decomposing an image into row and column DFTs. First it is to compute the DFT of the each column of the image, and after that we perform another DFT on each row of this intermediate results.

   Note: The task was to write a routine for 64x64 2-D DFT but *fft6464* can do it for any NxN size image, where $N=2^k$, $k \in Z^+$.

2. For this task I wrote two routines *fftn* and *fftn2*. In routine *fftn* I performed decimation in spatial domain until the image block is 64x64 and then I used the routine written in task 1. In routine *fftn2* I performed the full decimation in spatial domain until the image block is 2x2 for which is easy to compute the DFT. For both routine I used the material from the textbook on pg.76, topic 2.3.3 Vector-Radix Fast Fourier Transform and equations (2.77a-e) and (2.78a-d).

   Note: Both routines work for any NxN size image, where $N=2^k$, $k \in Z^+$.

   Here are presented execution times of four DFT routines for 512x512 pixels gray scale images of Lena:

Description	origin	function name	execution time
Row-Column Decomposition	matlab	*fft2*	1.8900
decimation in “time” until 64x64 and then 64x64 Row-Column Decomposition	mine	*fftn*	5.3910
decimation in “time” until 2x2	mine	*fftn2*	95.2030
decimation in “time” until 2x2	given example	*vrfft2*	more then I was willing to wait

Bad performance of *fftn2* can explained by the recursive character of program. It takes additional time for work space saving for each call of the routine.

Bad performance of *vrfft2*, in my opinion, can be explained by the matlab implementation of for .. end construction. To illustrate that, I tried to compare execution times of two small programs that perform the same thing: make a product $\underline{y}^T\underline{y}$ where y is a row vector of size 512 whose elements are uniform distributed numbers between 0 and 1.

I used this matlab function for time measurement:	TIC	Start a stopwatch timer.
	TOC	Read the stopwatch timer.

The sequence of commands : TIC, operation, TOC prints the time required for the operation.

```
clear;
y = rand(1,512);                          make a random vector y
tic;   for  i=1:512
            for j=1:512;
            z(i,j)=y(i)*y(j);
            end;
       end;   toc;                        elapsed time =        24.8130
                                          at this point z is defined as 512x512 matrix
tic;   for  i=1:512
            for j=1:512;
            z(i,j)=y(i)*y(j);             same routine as above
            end;
       end;   toc;                        elapsed time =        10.1100

tic;   z2=y'*y; toc;                      elapsed time =   0.2660

                                          at this point z2 is defined as 512x512 matrix
tic;   z2=y'*y; toc;                      elapsed time =   0.2500
```

We can see here that the performance is better if the matrix is already defined. This can be explained by the mechanism of dynamic allocating of memory by matlab. This is true just for big matrices.

3. Done.

In following experiment, a gray scaled image is used. Each pixel is presented as an 8-bit number with a range from 0 to 255. The image that is used is an image of Lena with size 512x512 pixels and a range of pixel values from 25 to 245, but for the purpose of displaying the image, it is scaled to a full range from 0 to 255.

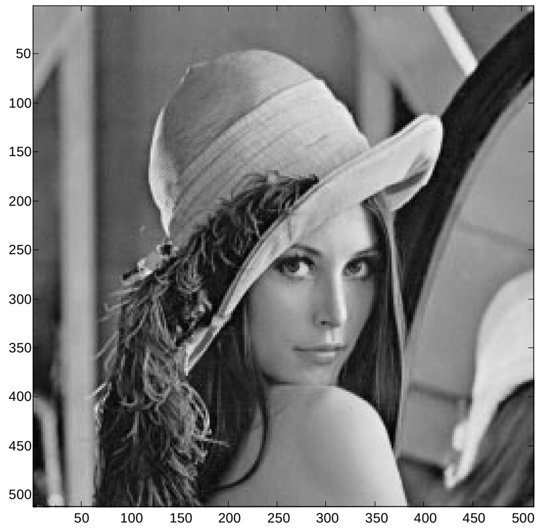

Figure 1. Gray scale image of Lena

4. By running the routine *fftn*, which gave us a 2D-DFT of the image, using the base 10 log scaled magnitude of it, and rearranging the elements, we got the following picture:

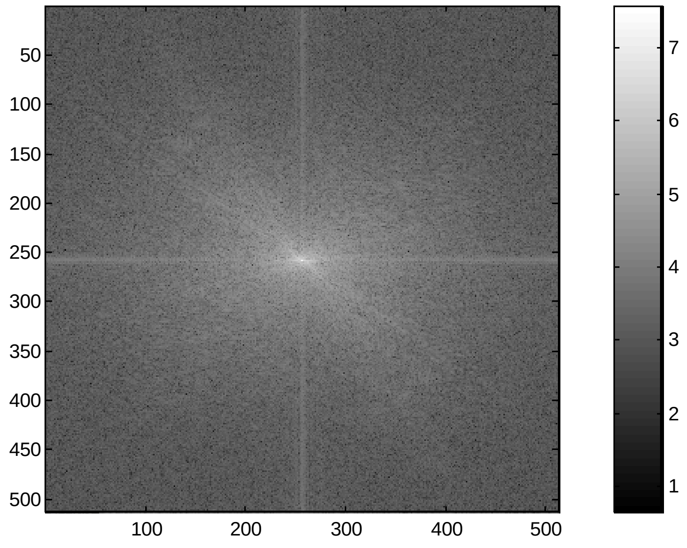

Figure 2. Base 10 log scaled magnitude spectrum of the gray scale image of Lena
The origin of the frequency plane is in the center of the plot

It can be seen that there is a lot of spectral energy near axises as well as in 135° direction because of lot of lines in 45° direction on the Lena image.

5. Derivation for 2D-IDFT by using existing 2D-DFT routine, for N=M is given belou:

$$x(n_1,n_2)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[X(k_1,k_2)*W_N^{-k_1*n_1}*W_N^{-k_2*n_2}\right]$$

$$X(k_1,k_2)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[x(n_1,n_2)*W_N^{k_1*n_1}*W_N^{k_2*n_2}\right]$$

where $W_N = e^{-i*\frac{2*\pi}{N}}$

$$x(n_1,n_2)^*=\left[\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[X(k_1,k_2)*W_N^{-k_1*n_1}*W_N^{-k_2*n_2}\right]\right]^*$$

$$x(n_1,n_2)^*=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[X(k_1,k_2)*W_N^{-k_1*n_1}*W_N^{-k_2*n_2}\right]^*$$

$$x(n_1,n_2)^*=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[\left[X(k_1,k_2)\right]^**W_N^{k_1*n_1}*W_N^{k_2*n_2}\right]$$

From the last equation it can be seen that 2D-IDFT can be performed by 2D-DFT with preprocessing the $X(k_1,k_2)$ and postprocessing 2D-DFT routine result, simply by conjugation in both cases:

- conjugate the input $X(k_1,k_2)$ and,
- use a existing 2D-DFT routine
- conjugating the result of 2D-DFT.

6. Image synthesis by using just the phase spectrum

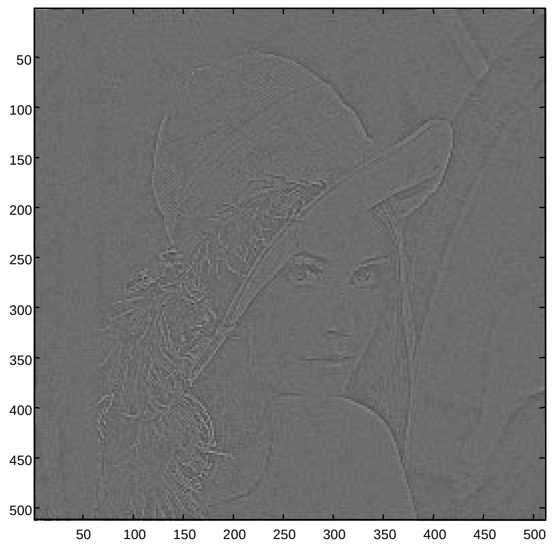

Figure 3. Image of Lena produced by IDFT by using just the phase spectrum of Lena for fixed magnitude spectrum at 1.

7. Image synthesis by using just the magnitude spectrum

The picture has too big dynamic to be shown as a gray scale image, so the base 10 log scale was used:

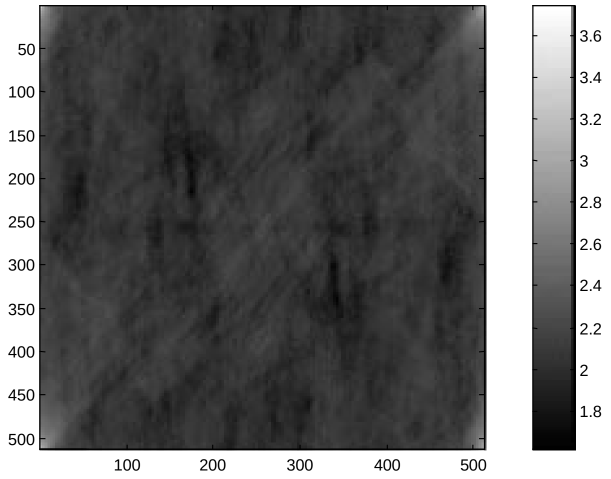

Figure 4. Log scaled image of Lena produced by IDFT by using just the magnitude spectrum of Lena

We can see that the central part of image has relatively small dynamic. If we scale the image according to the values that are in the central part of it e.g. values of region from 100th pixel to 400th pixel are mapped into values in range 0-255, then we get:

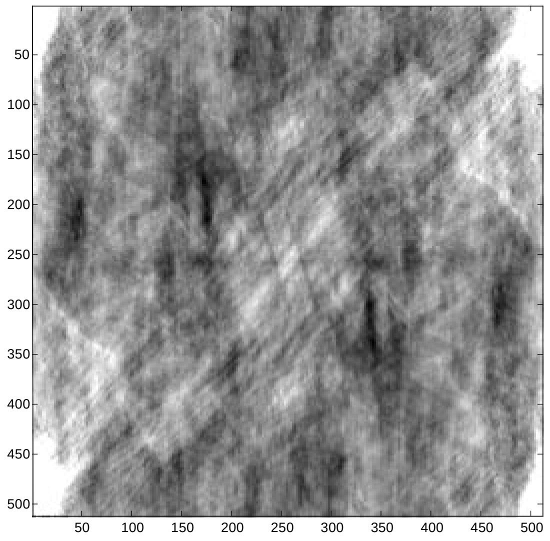

Figure 5. Image of Lena produced by IDFT by using just the magnitude spectrum of Lena scaled on the central part of the image. (region from 100:400 in both direction)

For zero phased signal we have big value at corners because we made a sum, of zero phased cos-components, that has its maximum at corners.

In following paragraph I will try to explain why is phase more important then magnitude. I picked up an example from 1-D case just for illustration. It is straight forward to apply that to 2D-case.

For humans to recognize the object from the image, it is more important to get an idea of a shape and couture, than to get the whole texture of the object. First humans recognize the a shape and after that they analyze the texture.

The edge is a sharp transition from one gray level to another. So what happens when we use just a phase or just a magnitude for IDFT?

If we imagine that we have just two sin functions, what kind of value we can expect for some point?
Lets define:

$$f_1 = A_1 * \sin(\omega + \phi_1)$$
$$f_2 = A_2 * \sin(\sqrt{2}\omega + \phi_2)$$

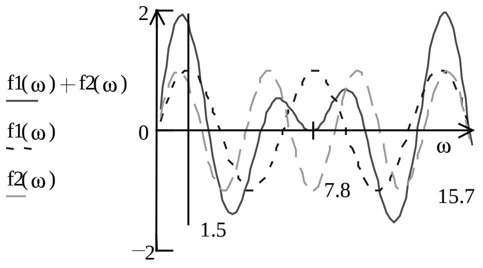

Figure 6. Two sin functions and their sum.

Let us fix ω=1.571 and consider the possible values at that point. If we change just the magnitude of amplitude A_1 and A_2, in range form 0..10, and plot contours for f_1+f_2=Const_value we have Figure 7., but if we change a phase ϕ_1 and ϕ_2 in range form -π to π and plot contours for f_1+f_2=Const_value we have Figure 8.

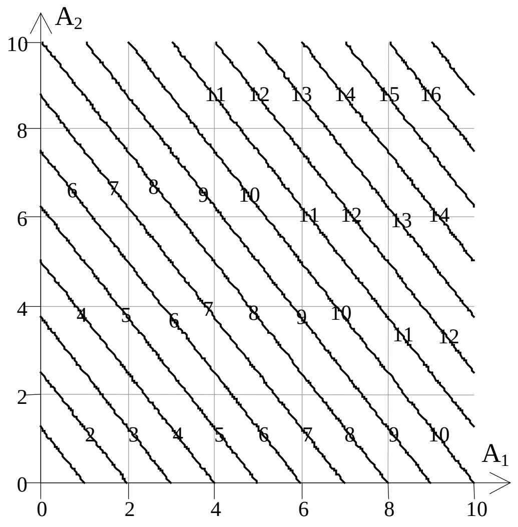

Figure 7. Contours in the magnitude plane for f_1+f_2 for different $A_{1,}$ A_2 and fixed $\phi_{1,}$ ϕ_2 to zero.

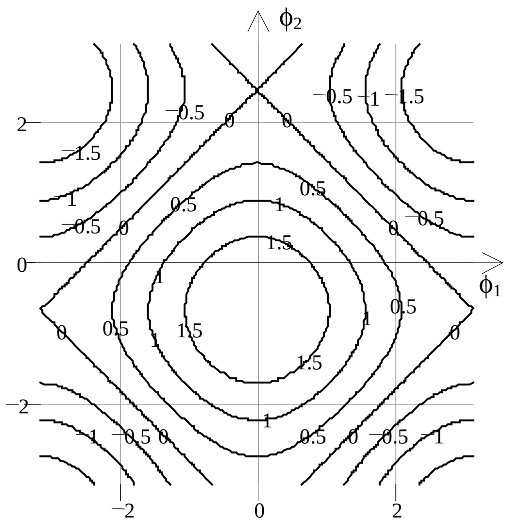

Figure 8. Contours in the phase plane for f_1+f_2 for different $\phi_{1,}$ ϕ_2 and $A_{1,}$ A_2 fixed to one.

We can see that by varying the phase of sin we can achieve transition at any point, that is not possible by varying the magnitude only.

8. Image synthesis by using the magnitude spectrum augmented by sing of the real part.

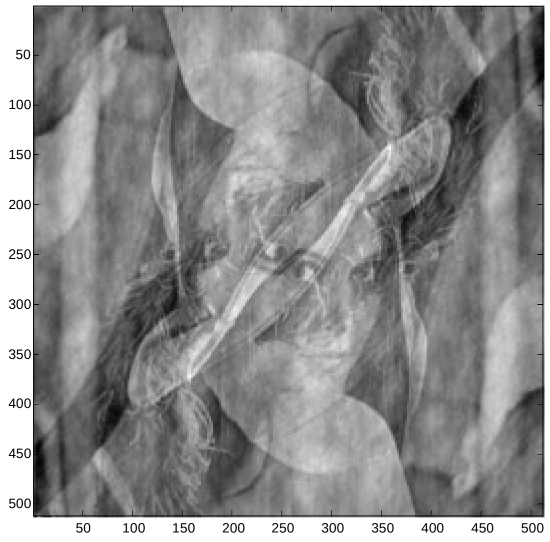

Figure 9. Image of Lena produced by IDFT by using the magnitude spectrum of Lena augmented by sing of the real part of the spectral amplitude.

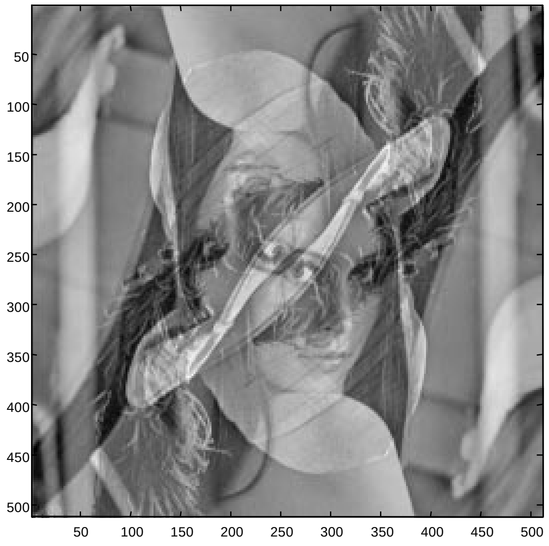

Figure 10. Image of Lena produced by IDFT by using the real part of the spectral amplitude.

The image on the Figure 9. , produced by IDFT by using the magnitude spectrum of Lena augmented by sing of the real part of the spectral amplitude, is a degenerated version of image on Figure 10. produced by IDFT using the real part of the spectral amplitude. From the textbook pg.35 equations (1.110a-b) and (1.111a-b) we can see the relation between the real and the imaginary part of the Fourier transform and the original image. Since :

$$\Im(x(n_1,n_2)) = X(\omega_1,\omega_2),$$

and
$$\Im\left[\frac{1}{2}\left(x(n_1,n_2)+x^*(-n_1,-n_2)\right)\right] = \mathrm{Re}\left[X(\omega_1,\omega_2)\right],$$

and
$$\Im\left[\frac{1}{2}\left(x(n_1,n_2)-x^*(-n_1,-n_2)\right)\right] = j\,\mathrm{Im}\left[X(\omega_1,\omega_2)\right],$$

we have image on Figure 10. being a sum of the original image and its flipped version.
If we quantize the phase to +j\pi and -j\pi on the imaginary axis instead of the real axis and then synthesize, the images are:

Image synthesis by using the magnitude spectrum augmented by sing of the imaginary part.

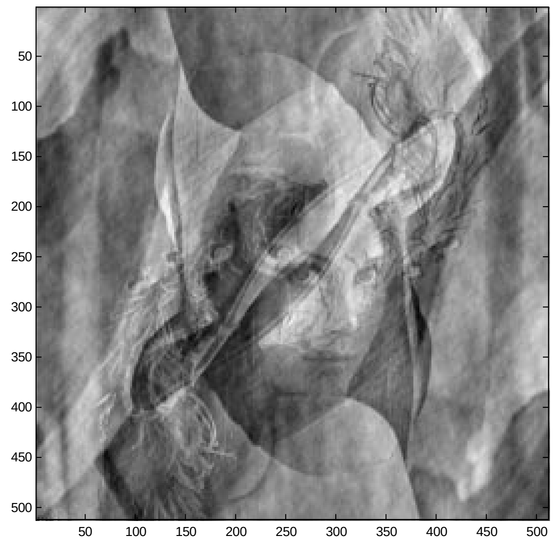

Figure 11. Image of Lena produced by IDFT by using just the magnitude spectrum of Lena augmented by sing of the imaginary part of spectral amplitude.

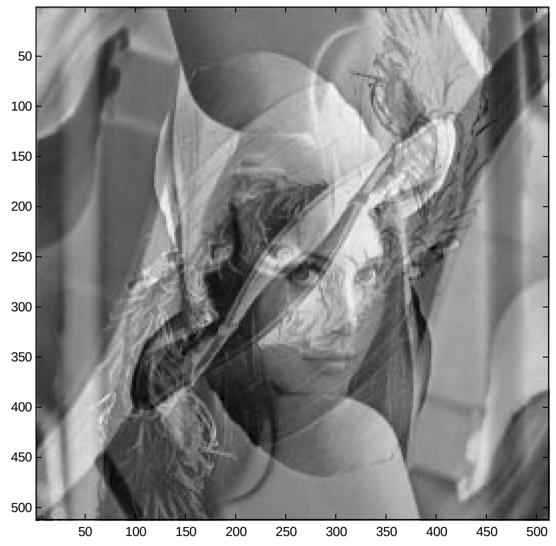

Figure 12. Image of Lena produced by IDFT by using just imaginary part of the spectral amplitude

.

We can see that the image is flipped and subtracted from the original, like the equations tell us.

Conclusion:

Regarding the question, which one is most visually informative, I cannot really give an answer. I am probably biased, by being an engineer, and I ask: What do we want from the picture? Shape, size, brightness...or just Lena.

If we want the exact shape and the size of the object on the image, then the Figure 3. is the best, but if we want to get some feeling of texture or material then probably the Figure 9. is better.

Bonus Work

1. Image synthesis by replacing the magnitude spectrum with random numbers

We can play with magnitude spectrum as much as we want and we will get almost the same results, unless we use some distribution that goes below zero. Negative value change the phase value and after that we cannot recognize our Lena any more.

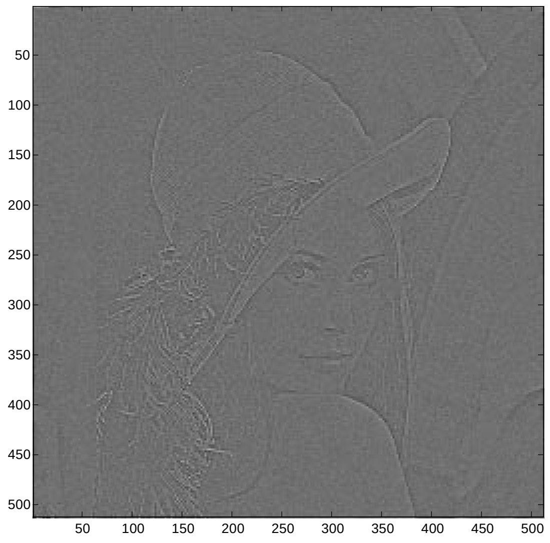

Figure 11. Image of Lena produced by IDFT by using just the phase spectrum of Lena and replacing the magnitude spectrum with uniform distributed random numbers between 0 and 50, μ=25, σ=625.

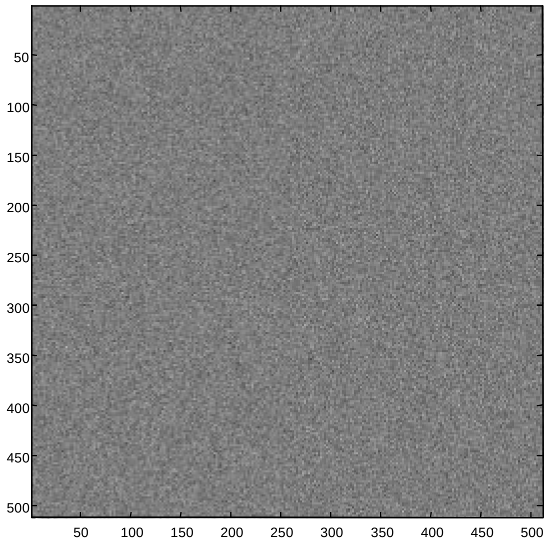

Figure 12. Image of Lena produced by IDFT by using just the phase spectrum of Lena and replacing the magnitude spectrum with normal distributed random numbers, μ=0, σ=1.

As we can see from the Figure 12., even for very "weak" normal random process, with σ=1, there is no more Lena on the image.

2. Image synthesis by filtering in spatial domain

If we agree that our image in task 6, reconstructed just from the phase, is the edge of the image, in some sense, we need to find the edge detector function. That can be some filter that makes difference between current point and neighborhood points.

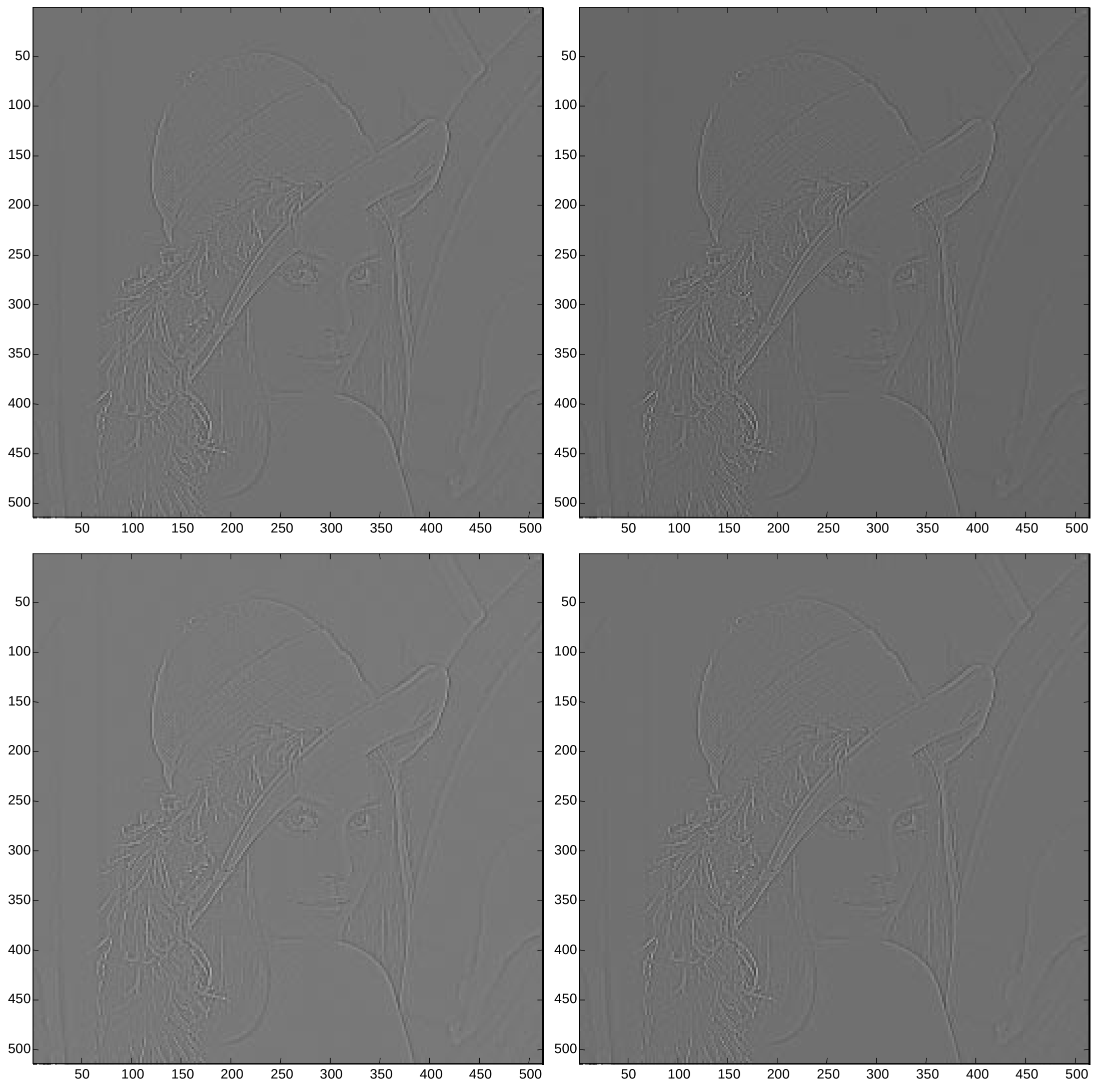

Figure 12. Image of Lena produced by filtering in spatial domain
by four types of filters receptively.

Four types of filters	
1) $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	2) $\begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$
3) $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	4) $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} + \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$

We can see that the filter 3x3, for N required to be an odd number, can make the image to look like the phase-only synthesized image.