Practica 3: Convolución lineal y circular utilizando la DFT

Aguilera Palacios Luis Ernesto, leapvader.1998@hotmail.com
Padilla Castillo Aaron Samir, samir.castill@gmail.com
Facultad de Ingeniería, Universidad Nacional Autónoma de México

***Resumen*— En el siguiente trabajo se mostrará como se llevan a cabo las transformaciones en el dominio de Fourier para el procesamiento de imágenes y cuáles son las implicaciones de utilizar una convolución lineal o una circular.**

***Abstract*— The following work shows how the transformations in the Fourier domain are carried out for image processing and what are the implications of using a linear or a circular convolution.**

I. Objetivos

- Calcular la convolución lineal y la convolución circular de una imagen y explicar sus diferencias.
- Comprender la relación f(x)⊗h(x) F←→F(k)×H(k)
- Comprobar que la mayor concentración de la energía se encuentra en los primeros coeficientes de la DFT.

II. Introducción

DFT y FFT

Dada la limitación computacional para procesar imágenes, debemos utilizar la FFT que es un algoritmo con el cuál podemos calcular con un computador la DFT.

Dado que una DFT 2-dimensional puede separarse en transformaciones 1-dimensionales podemos decir que:

$$F(u,v) = \sum_{x=0}^{M-1} e^{-j2\pi ux/M} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi vy/N}$$

$$= \sum_{x=0}^{M-1} F(x,v) e^{-j2\pi ux/M}$$

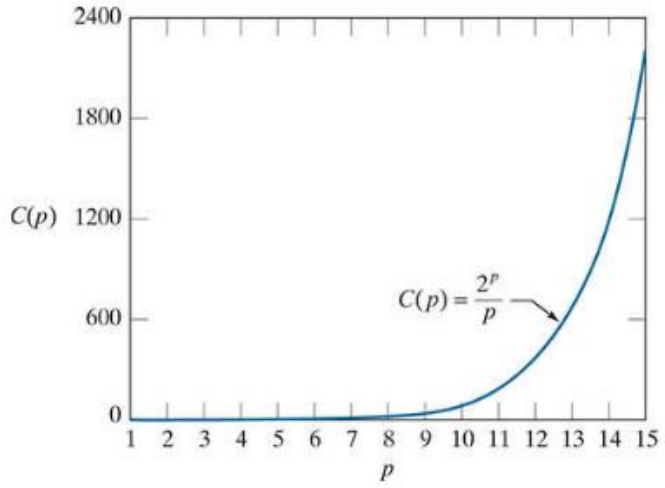
En donde:

$$F(x,v) = \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi vy/N}$$

Esto nos permite computar una DFT 2-dimensional como un conjunto de transformaciones 1-dimensionales sobre f(x,y) por cada fila y luego ejecutando otro ciclo sobre las columnas obtenidas, lo que es una reducción importante de complejidad por que solo tenemos que usar una variable a la vez. Sin embargo esto aún es una complejidad muy elevada, pues se requieren del orden de $(MN)^2$ operaciones para calcular la DFT, por ejemplo en una imagen de 2048*2048 el número de operaciones se encuentra aporx. Cerca de los 17 billones de operaciones, sin embargo si utilizamos la FFT el número se reduce a tan solo 92 millones, pues pasamos a un orden de CNlog(N) para una constante K.[1]

La FFT es un algoritmo computacional que tiene distintas implementaciones pero todas tienen un propósito; permitir hacer un cálculo bastante rápido de la DFT, sin embargo el primero en proponer un algoritmo para realizar la FFT fue Gauss, desde entonces varios métodos más han sido descritos y todos son válidos para calcular la DFT.[3]

En la siguiente tabla se muestra como un algoritmo de FFT puede mejorar el rendimiento con respecto a la DFT, en un cálculo 1-dimensional[1]:



Filtrado de imágenes

Una aplicación importante de la DFT es que al transformar al dominio de la frecuencia podemos también utilizar el teorema de convolución pues este es complicado de calcular en el dominio del tiempo, pero no en el da la frecuencia, lo que hace que filtrar imágenes en el dominio de la frecuencia sea preferido computacionalmente para ahorrar una cantidad significativa de recursos.[2]

Algunos de los usos más extendidos son los filtros gaussianos, pues estos sirven bastante bien para restaurar imágenes como la

*

siguiente[4]:

Original image

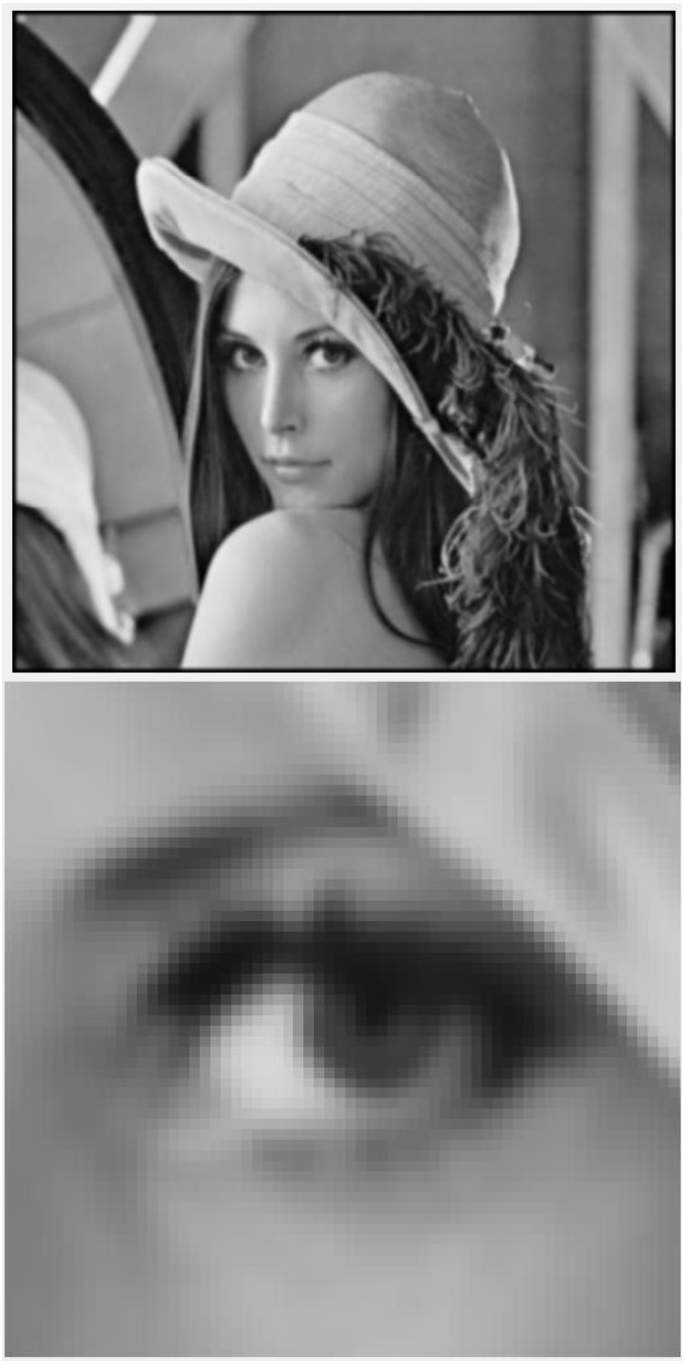
Image with illumination interference

Image restored by homomorphic filtering

III. DESARROLLO

1. Obtener la convolución lineal (comando MATLAB conv2 y argumentos 'full', 'same' y 'valid' de la imagen con un filtro paso bajas (filtro de bloque). Usar 2 o 3 tamaños diferentes de filtros, por ejemplo: 7x7, 9x9 y 11x11. Desplegar las imágenes resultantes.

Fig. 1. Imagen original



Figs. 2 y 3. Filtrado con filtro binomial de 7*7, convolucion lineal en dominio espacial.

Figs. 4 y 5. Filtrado con filtro binomial de 11*11, convolucion lineal en dominio espacial.

Fig. 6 y 7. Filtrado con parámetro “same”, filtros 7*7 y 11*11.

Fig. 8 y 9. Filtrado con parámetro "valid", filtros 7*7 y 11*11.

2. Obtener la DFT de la imagen original y desplegarla de manera amplificada utilizando el logaritmo del módulo de la DFT para dicha amplificación. Cambiar el eje de coordenadas (comando MATLAB fftshift) y nuevamente amplificar.

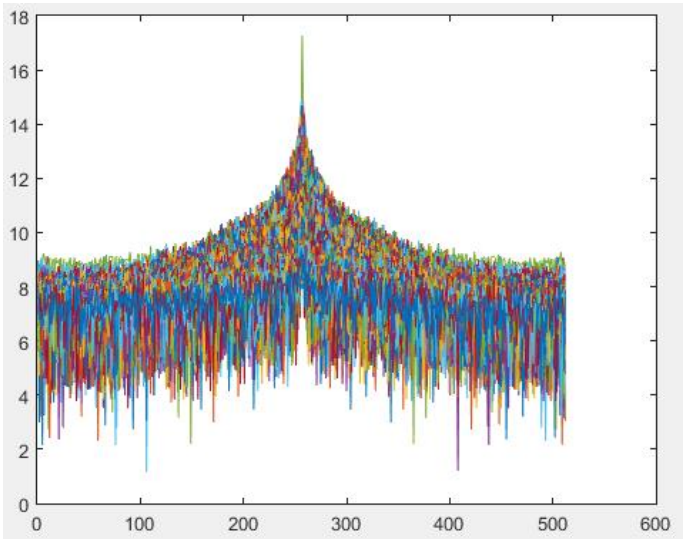


Fig. 10. Representación en Fourier de la original.

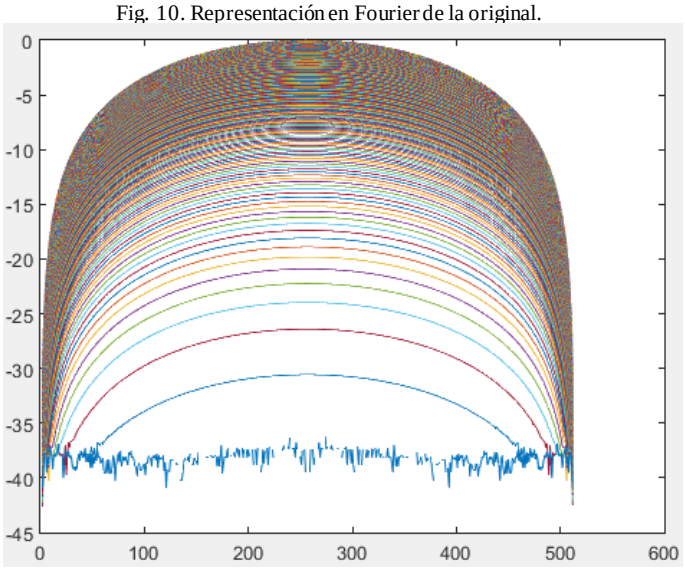


Fig. 11. Filtro bonomial en Fourier de 7*7.

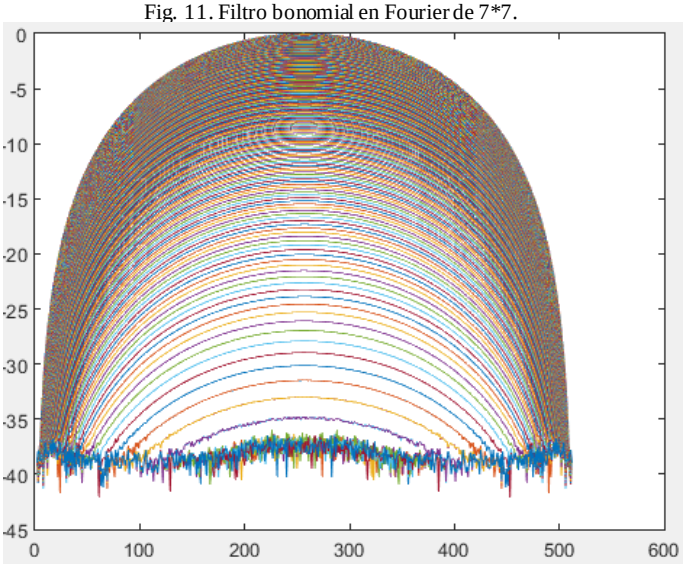


Fig. 12. Filtro bonomial en Fourier de 11*11.

3. Obtener la convolución circular (⊗) de la imagen con el filtro paso bajas a través de la DFT. Usar también diferentes tamaños de filtros. Desplegar las imágenes resultantes.

Fig. 13. Transformada inversa de la convolucion circular, filtro de 7*7.

Fig. 14. Transformada inversa de la convolucion circular, filtro de 11*11.

Fig. 15. Convolución lineal con FFT, filtro de 7*7.

Fig. 16. Convolución lineal con FFT, filtro de 11*11.

4. Obtener la convolución lineal (∗) de la imagen con el filtro paso bajas a través de la DFT (comandos MATLAB fft2 y ifft2). Usar también diferentes tamaños de filtros. Desplegar las imágenes resultantes. (Recordar que $f(x) \otimes h(x) \; F \longleftrightarrow F(k) \times H(k)$ y $* \; \square = \otimes$)

5. Comparar los resultados obtenidos en los puntos 1,3 y 4 desplegando para un mismo tamaño de filtro, las 3 convoluciones, por ejemplo: convolución lineal filtro 9x9, convolución lineal (DFT) filtro 9x9, convolución circular (DFT) filtro 9x9.

Fig. 17. Esquina superior izq. conv. lineal en el espacio, filtro de 11*11.

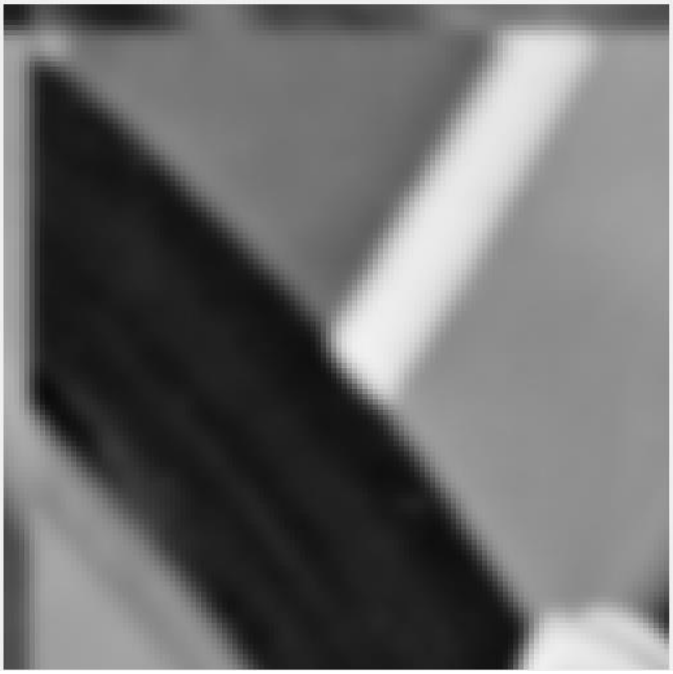


Fig. 18. Esquina superior izq. conv. circular en frecuencia, filtro de 11*11.

Fig. 19. Esquina superior izq. conv. lineal en frecuencia, filtro de 11*11.

IV. Conclusiones

A pesar de que el resultado final es muy parecido, se logran observar pequeñas diferencias entre los 3 tipos de convolución y se aprecia como en el borde la imagen obtiene ruido que no estaba en la imagen original, por lo que debemos tomarlo en cuenta a la hora de filtrar imágenes para nuestros sistemas, pues este ruido puede afectar los resultados.

V. Código fuente

```
%Practica de convolución en Fourier

%obtenemos nuestros datos
imagen=imread('lenna.jpeg', 'jpg');
img_size = size(imagen);
padding_size = img_size(1);
%img_t=rgb2gray(imagen);
figure(1);
imshow(imagen);

%generamos los filtros
filter_7=generate_filter(7);
%G_7 = fspecial('gaussian',[7 7],2); %filtro gaussiano con desviaci□n
%est□ndar
filter_7_padding=zero_padding(filter_7, padding_size);
fft_7=fft2(filter_7_padding);
filter_11=generate_filter(11);
%G_7 = fspecial('gaussian',[11 11],2);
filter_11_padding=zero_padding(filter_11, padding_size);
fft_11=fft2(filter_11_padding);
```

Fig. A. Carga de imagen y generación de filtros

```
%original en Fourier
fft_lenna=fft2(imagen);
figure(2);
plot(log(abs(fftshift(fft_lenna))));

%filtros en Fourier
figure(3);
plot(log(abs(fftshift(fft_7))));
figure(4);
plot(log(abs(fftshift(fft_11))));

%Imagenes convolucionadas
convl_7=imfilter(imagen, filter_7, 'full');
figure(5);
imshow(convl_7);
convl_11=imfilter(imagen, filter_11, 'full');
figure(6);
imshow(convl_11);
```

Fig. B. Convolución lineal en el espacio y transformadas de Fourier.

```
%multiplicacion en dominio de Fourier(conv. circular)
fourier_7=fft_lenna.*fft_7;
figure(7);
plot(log(abs(fftshift(fourier_7))));
figure(8);
imshow(ifft2(fourier_7));
fourier_11=fft_lenna.*fft_11;
figure(9);
plot(log(abs(fftshift(fourier_11))));
figure(10);
imshow(ifft2(fourier_11));

%desplegamos las convoluciones circulares
ifft_7=ifft2(fourier_7);
ifft_7=uint8(ifft_7);
figure(11);
imshow(ifft_7);
ifft_11=ifft2(fourier_11);
ifft_11=uint8(ifft_11);
figure(12);
imshow(ifft_11);
```

Fig. C. Convolución circular en el dominio de la frecuencia

```
%convoluciones lineales con DFT
padding_size_l = padding_size+7-1;
lenna_padding = zero_padding(imagen, padding_size_l);
fft_lenna_lin=fft2(lenna_padding);
filter = zero_padding(filter_7, padding_size_l);
fft_filter_lin=fft2(filter);
fft_lin=fft_lenna_lin.*fft_filter_lin;
ifft_7_lin=ifft2(fft_lin);
ifft_7_lin=uint8(ifft_7_lin);
figure(13);
imshow(ifft_7_lin);

padding_size_l = padding_size+11-1;
lenna_padding = zero_padding(imagen, padding_size_l);
fft_lenna_lin=fft2(lenna_padding);
filter = zero_padding(filter_11, padding_size_l);
fft_filter_lin=fft2(filter);
fft_lin=fft_lenna_lin.*fft_filter_lin;
ifft_11_lin=ifft2(fft_lin);
ifft_11_lin=uint8(ifft_11_lin);
figure(14);
imshow(ifft_11_lin);
```

Fig. D. Convolución lineal en el dominio de la frecuencia con la FFT.

```
%function to generate square gussian filter of size x

function [m]=generate_filter(size)
    A=pascal_vector(size);
    B=A.';
    m=B*A;
    m=m/sum(sum(m));
end
```

Fig. E. Función para generar el filtro cuadrado gaussiano.

```
%Zero padding function to fill with zeros a matrix

function [m] = zero_padding(original_m, final_size)
    original_m(final_size, final_size)=0;
    m=original_m;
end
```

Fig. F. Función para rellenar con zeros hacia la esquina inferior derecha.

```
%Function that retrieves a vector from Pascal's triangle level

function [row] = pascal_vector(m)
    if m == 1
        row = 1;
    elseif m == 2;
        row = [1 1];
    else
        row = [1,1];
          for i=2:m-1
            row = conv([1 1],row);
          end
    end
end
```

Fig. G. Función que devuelve un nivel del triángulo de pascal.

REFERENCIAS

[1] R. E. Woods y R. C. Gonzalez, *Digital Image Processing*, Fourth. Pearson, 2018.

[2] A. Broughton y K. Bryan, *Discrete Fourier Analysis and Wavelets Applications to Signal and Image Processing*, Second. Wiley, 2018.

[3] W. Briggs y H. Van Emden, The DFT: An Owners' Manual for the Discrete Fourier Transform , First. Society for Industrial Mathematics, 1987.

[4] K. Rao, D. Kim, y J. Hwang, Fast Fourier Transform - Algorithms and Applications , First. Springer Netherlands, 2010.