Procesamiento de Imágenes

Yemel Angel Jardi Bello (49150) — Santiago José Samra (49030)

Resumen

El objetivo del presente artículo es analizar espectralmente una imagen y utilizar la transformada discreta de Fourier para realizar filtrados espaciales.

1. Introducción

El pepe digital de imágenes es el conjunto de técnicas que se aplican a imágenes digitales con el objetivo de mejorar su calidad o facilitar la búsqueda de información en ellas. El proceso de filtrado es el conjunto de técnicas englobadas dentro del preprocesamiento de imágenes cuyo objetivo fundamental es obtener, a partir de una imagen original, otra final cuyo resultado sea más adecuado para una aplicación específica mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella. Los principales objetivos que se persiguen con la aplicación de filtros son: suavizar la imagen, eliminar ruido, realzar bordes y detectar bordes.

La motivación de este artículo es difundir el uso de la Transformada de Fourier para la implementación de filtros dentro del campo del procesamiento digital de imágenes. En la sección 2.1 se presentará la Transformada de Fourier sobre dos dimensiones, luego en la sección 2.2 se computarán las imágenes correspondientes a la amplitud y fase de una imagen arbitraria. En la sección 2.3 se presentará el concepto de filtro, su utilización y algunos ejemplos. Por último en la sección 3 se presentan los resultados y conclusiones.

2. Desarrollo

2.1. Transformada de Fourier en dos dimensiones

La Transformada de Fourier puede ser generalizada a varias dimensiones. En nuestro caso, una imagen puede interpretarse como una señal de dos dimensiones, por tanto utilizaremos una generaliación bidimensional de la Transformada de Fourier sobre variables discretas (Ecuación 1). Análogamente, para el proceso de anti-transformación utilizamos una generalización bidimensional de la anti-transformada sobre variables discretas (Ecuación 2). Las funciones 5 y 5 del anexo muestran una simple implementación de las ecuaciones anteriores respectivamente. Sin embargo utilizando diferentes propiedades es posible realizar optimizaciones a las mismas para lograr una mayor eficiencia de cómputo. Es por esto que a fines prácticos utilizamos las funciones `fft2` y `ifft2` de *Octave* que calculan la Transformada Rápida de Fourier y la Antitransformada Rápida de Fourier respectivamente.

$$X_{l,k} = \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x_{n,m} e^{-i\frac{2\pi}{N}(nl+mk)} \tag{1}$$

$$x_{n,m} = \frac{1}{N^2} \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} X_{l,k} e^{+i\frac{2\pi}{N}(nl+mk)} \tag{2}$$

2.2. Imágenes en amplitud y fase

A modo de ejemplo tomamos una imagen arbitraria, en una escala de 256 grises, llamada `saturno` (1). A la misma le aplicamos la Transformada de Fourier en dos dimensiones y remapeando el resultado al intervalo entero $[0, 255]$ obtuvimos su representación en fase (2), al mismo tiempo aplicando módulo al resultado resultó su representación en amplitud (3). Para comprobar el proceso podríamos aplicar la Antitransformada de Fourier para obtener nuevamente la imagen original.

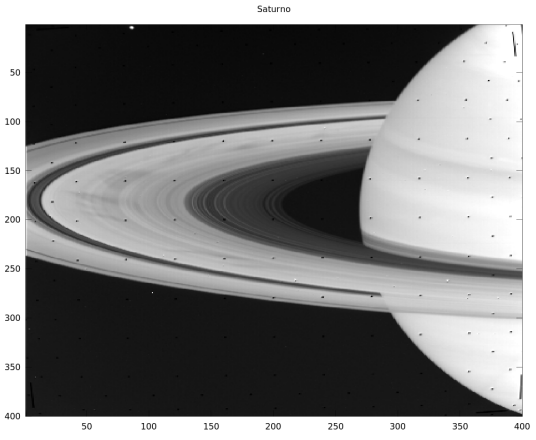

Figura 1: Imagen original.

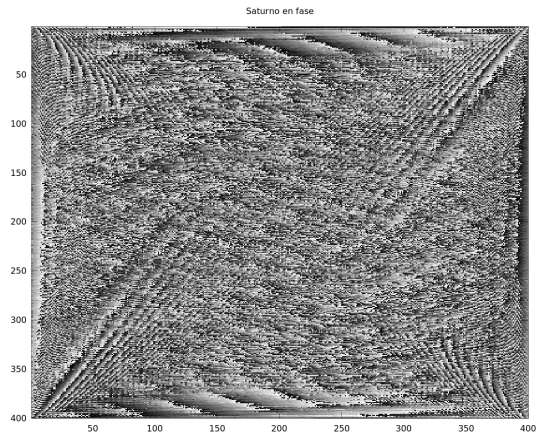

Figura 2: Imagen original transformada en fase.

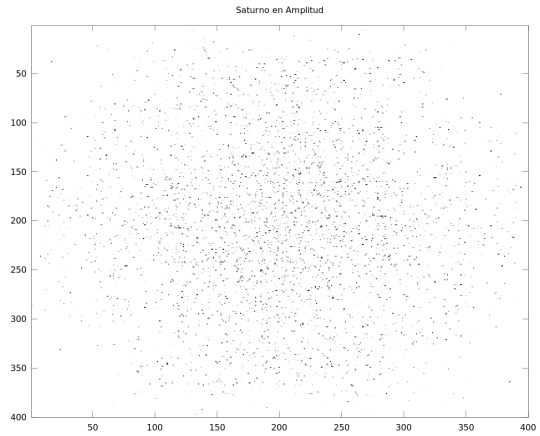

Figura 3: Imagen original transformada en amplitud.

2.3. Filtros de imágenes utilizados

Un filtro es una función que opera contra la representación en fase de una señal, existiendo así diferentes tipos de filtros según su comportamiento. Los filtros pasa bajos son aquellos que eliminan las altas frecuencias y, en el campo de las imágenes, permiten suavizar una imagen, eliminar ruido y detalles pequeños de poco interés ya que sólo afecta a zonas con muchos cambios. Por otro lado, los filtros pasa alto, quienes eliminan las bajas frecuencias, intensifican detalles, bordes y cambios de alta frecuencia y atenúan las zonas de tonos uniformes.

Para aplicar un filtro a una imagen es necesario representarlo inicialmente como una matriz, como así también la imagen con la sobre la que se operará. Luego debemos aplicar la Transformada de Fourier a esta imagen, multiplicar elemento a elemento el resultado contra el filtro y, finalmente aplicar la Antitransformada de Fourier al resultado para recuperar la imagen filtrada. Es decir, los filtros se aplican en la frecuencia. Si x es la matriz que contiene a la imagen, X será su transformada y $xfil$ será la imagen con el filtro aplicado. Se denomina H al filtro y F a la Transformada Discreta de Fourier Bidimensional.

$$\begin{aligned} X &= F(x) \\ xfil &= F^{-1}(H * X) \end{aligned}$$

Para ejemplificar la utilización de los filtros implementamos tres diferentes y su definición se presenta a continuación:

- $$H_{k,l} = \begin{cases} 0 & \text{si } 0 \leq k \leq 400, 190 \leq l \leq 210 \\ 0 & \text{si } 190 \leq k \leq 210, 0 \leq l \leq 400 \\ 1 & \text{en otro caso} \end{cases} \quad (3)$$

- Filtro Gaussiano
$$H_{k,l} = e^{-0{,}01(k^2+l^2)} \quad (4)$$

- El damero
$$H_{k,l} = \begin{cases} 0 & \text{si } l+k \text{ es par} \\ 1 & \text{si } l+k \text{ es par} \end{cases} \quad (5)$$

3. Resultados y conclusiones

El resultado de aplicar el primer filtro a la imagen `saturno` se puede observar en la Figura 4. En la misma no apreciamos grandes cambios respecto a la imagen original. En cambio, al aplicar el filtro Gaussiano obtuvimos un efecto de `blur` sobre la imagen original, esto se puede observar en la Figura 5. Por último el filtro Damero produce un efecto de simetría sobre la imagen superponíendola con ella misma, este resultado puede verse en la Figura 6.

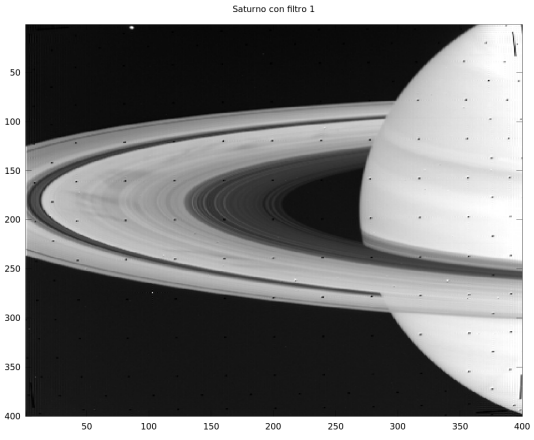

Figura 4: La imagen de saturno con el filtro 1 aplicado.

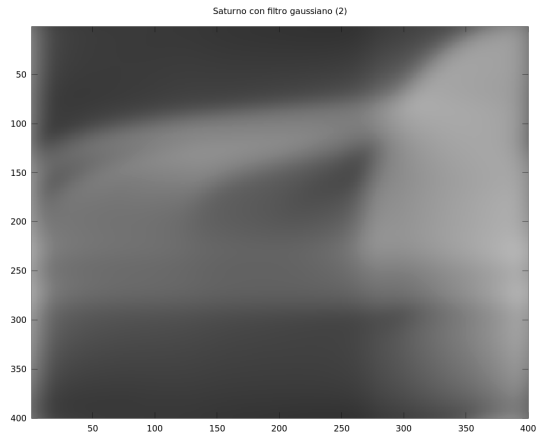

Figura 5: La imagen de saturno con el filtro Gaussiano aplicado.

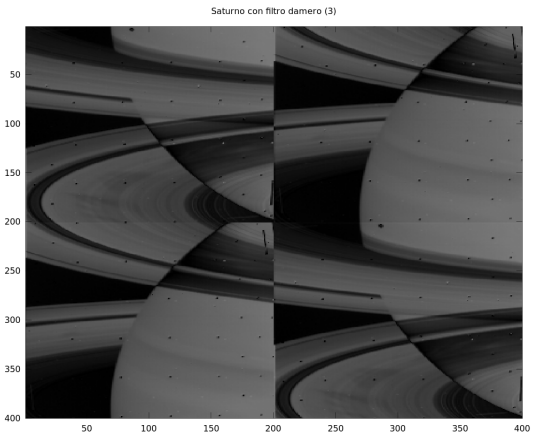

Figura 6: La imagen de saturno con el filtro Damero aplicado.

Por lo observado pordemos concluir que realizar el filtrado digial de imágenes mediante la Transformada Discreta de Fourier bidimensional es de baja complejidad en su implementación. Al mismo tiempo proporciona flexibilidad en el diseño de soluciones de filtrado y rapidéz si se utiliza una primitiva eficiente como la que ofrece *Octave*.

4. Bibliografía

- Fierens, Pablo. *Guía 2*, Métodos Numéricos Avanzados. ITBA, 1er Cuatrimestre 2011
- C. Pinilla, A. Alcalá y F. J. Ariza. *Filtrado de imágenes en el dominio de la frecuencia*, Departamento de Ingeniería Cartográfica, Geodésica y Fotogrametría. Universidad de Jaén. URL: `http://www.aet.org.es/revistas/revista8/AET8_5.pdf` - Accedido por última vez el 9 de junio del 2011.
- González, R.C., Wintz, P. *Procesamiento digital de imágenes* URL: `http://dmi.uib.es/~catalina/docencia/PDS/tema3.pdf` - Accedido por última vez el 9 de junio del 2011.

5. Anexo

Aquí se pueden ver las funciones de *GNU Octave* utilizadas para este análisis.

El *script* `filter1.m` implementa el filtro 1.

```
filter1.m


function H = filter1()
    H = ones(400, 400);
    for i=1:400
        for j=190:210
            H(i,j) = 0;
        endfor
    endfor

    for i=190:210
        for j=1:400
            H(i,j) = 0;
        endfor
    endfor

endfunction
```

El *script* `filter2.m` implementa el filtro Gaussiano.

```
filter2.m


function H = filter2()
    H = zeros(400, 400);
    for i=1:400
        for j=1:400
            H(i,j) = exp(-0.01*(i^2 + j^2));
        endfor
    endfor
endfunction
```

El *script* `filter3.m` implementa el filtro Damero.

```
filter3.m


function H = filter3()
    H = zeros(400, 400);
    for i=1:400
        for j=1:400
            H(i,j) = mod(i+j, 2);
        endfor
    endfor

endfunction
```

El *script* `loadData.m` carga la imagen en memoria.

loadData.m

```
function [image map] = loadData(file)
map = colormap(gray(255));
image = load(file);
endfunction
```

El *script* `main.m` tiene por objetivo cargar la imagen `saturno` en memoria, computar la Transformada Discreta de Fourier de esta imagen, su inversa y aplicarle 3 diferentes filtros. Deja los archivos resultado en el mismo directorio.

main.m

```
[x y] = loadData('../datos/saturno');
colormap(gray(255));

% Se muestra la imagen original
image(x');
title("Saturno");
print("-dpng", "../images/saturnoVisible.png");

% Se muestra la imagen transformada en phase
X = fftn(x);
phase = angle(X);
mx = max(max(phase));
mn = min(min(phase));

m = 255/(mx-mn);
b = -m*mn;

nphase = floor(m*phase + b);
image(nphase');
title("Saturno en fase");
print("-dpng", "../images/saturnoEnFase.png");

% Se muestra la imagen transformada en amplitud
image(abs(X)');
title("Saturno en Amplitud");
print("-dpng", "../images/saturnoEnAmplitud.png");


% Se muetra la imagen original tras las transformaciones
xprima = ifftn(X);
image(abs(xprima)');
title("Saturno recuperado");
print("-dpng", "../images/saturnoRecuperado.png");


% Se aplica el primer filtro
Xfilter1 = filter1().*X;
xfilter1 = abs(ifftn(Xfilter1))';
image(xfilter1);
title("Saturno con filtro 1");
print("-dpng", "../images/saturnoFiltrado1.png");

% Se aplica el segundo filtro
Xfilter2 = filter2().*X;
xfilter2 = abs(ifftn(Xfilter2))';
image(xfilter2);
title("Saturno con filtro gaussiano (2)");
print("-dpng", "../images/saturnoFiltrado2.png");

% Se aplica el segundo filtro
Xfilter3 = filter3().*X;
xfilter3 = abs(ifftn(Xfilter3))';
image(xfilter3);
title("Saturno con filtro damero (3)");
print("-dpng", "../images/saturnoFiltrado3.png");
```

El *script* `suma.m` implementa una suma necesaria para calcular la Transformada Discreta de Fourier de una secuencia bidimensional.

suma.m

```
function h = suma(x, l, k)
h = 0;
N = columns(x);
exp = - i * 2 * pi / N;
for n = 1:N
for m = 1:N
h += x(n,m) * e^(exp * (n*l + m*k));
endfor
endfor
endfunction
```

El *script* `sumaInv.m` implementa una suma necesaria para calcular la Transformada Discreta de Fourier Inversa de una secuencia bidimensional.

sumaInv.m

```
function h = sumaInv(x, l, k)
h = 0;
N = columns(x);
exp = i * 2 * pi / N;
for n = 1:N
for m = 1:N
h += x(n,m) * e^(exp * (n*l + m*k));
endfor
endfor
endfunction
```

El *script* `ftbi.m` implementa la Transformada Discreta de Fourier de una secuencia bidimensional.

ftbi.m

```
function X = fftbi(x)
for i=1:columns(x)-1
for j=1:columns(x)-1
X(i,j) = suma(x,i,j);
endfor
endfor
endfunction
```

El *script* `ftbiInv.m` implementa la Transformada Discreta de Fourier Inversa de una secuencia bidimensional.

ftbiInv.m

```
function x = fftbiInv(X)
for i=1:columns(X)-1
for j=1:columns(X)-1
x(i,j) = sumaInv(X, i, j);
endfor
endfor
x = x * (1/(columns(X)^2));
endfunction
```