**Instituto Politécnico Nacional**

**Escuela Superior De Cómputo**


# Entregable 15

# Señal Estacionaria

**Nombre de los integrantes:**

Hernandez Rodriguez Juan Uriel

Vergara Martinez Brenda

García Quiroz Gustavo Ivan

Gutiérrez Jiménez Cinthia Nayelli

Ramírez Carrillo José Emilio

Iturbide Serrano Uriel

**Grupo:** 5CV1

**Procesamiento Digital de Señales**

**Nombre del profesor:** José Antonio Flores Escobar

# Índice de Contenidos

Marco teórico ..... 1

Wavelet ..... 1

Wavelet Compression ..... 1

Desarrollo ..... 2

Simulación ..... 3

Conclusión ..... 4

Referencias ..... 4

Anexo ..... 5

# Marco teórico

## Wavelet

Es una función matemática de corta duración y oscilatoria que se utiliza para descomponer una señal en componentes de diferente frecuencia, permitiendo un análisis detallado tanto en el dominio del tiempo como en el dominio de la frecuencia. A diferencia de la Transformada de Fourier, que solo proporciona información de frecuencia, la Transformada Wavelet proporciona información simultánea de tiempo y frecuencia, lo que es particularmente útil para el análisis de señales no estacionarias o transitorias. Las wavelets se aplican en una variedad de campos, como procesamiento de señales, compresión de datos, y análisis de imágenes, debido a su capacidad para capturar características locales y detalles finos de la señal.

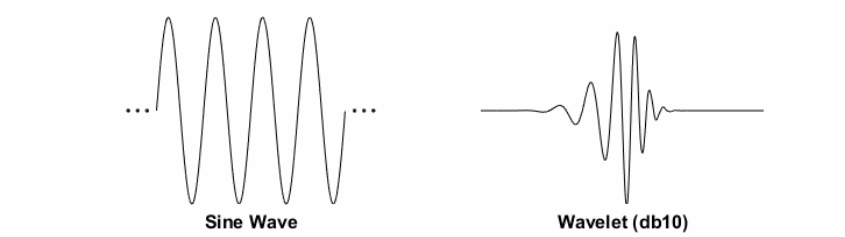


*Ilustración 1 - Diferencia entre señal sinusoidal y wavelet*

## Wavelet Compression

Es una forma de compresión de datos muy adecuada para la compresión de imágenes (a veces también para la compresión de vídeo y de audio). Las implementaciones notables son JPEG 2000, DjVu y ECW para imágenes fijas, JPEG XS, CineForm y Dirac de la BBC. El objetivo es almacenar datos de imágenes en el menor espacio posible en un archivo. La compresión wavelet puede ser sin pérdidas o con pérdidas. Utilizando una transformada de wavelet, los métodos de compresión wavelet son adecuados para representar transitorios, como sonidos de percusión en audio, o componentes de alta frecuencia en imágenes bidimensionales. Esto significa que los elementos transitorios de una señal de datos pueden representarse mediante una cantidad menor de información que si se hubiera utilizado alguna otra transformada, como la más extendida transformada discreta del coseno.

# Desarrollo

Para el desarrollo de esta práctica se debe de cargar una imagen en escala de grises y se convierte a formato *double* para facilitar las operaciones matemáticas. La variable *delta* es un parámetro que determina el umbral para la compresión. Los valores de la transformada que estén por debajo de este umbral serán considerados como ruido y serán eliminados. Posteriormente se definen tres matrices *H1, H2* y *H3* que corresponden a las transformaciones básicas de la wavelet de Haar. Teniendo eso, se normalizan las matrices para asegurar que las columnas de la matriz resultante sean ortonormales. Si es exitoso, se crean matrices de ceros *y0* e *y* con el mismo tamaño que la imagen original. En el bucle se aplica la transformada wavelet de Haar en bloques de 8x8 píxeles de la imagen ingresada.

Para el siguiente paso se realiza la compresión umbralizando los valores pequeños (absolutos menores que *delta*) en las matrices transformadas y se cuenta el número de elementos no nulos antes y después de la compresión. Una vez hecho eso, se debe de aplicar la transformada inversa para reconstruir la imagen comprimida a partir de los bloques de 8x8. Finalmente, se calcula el factor de compresión, que es la proporción de los valores no nulos después de la compresión en comparación con antes de la compresión.

```
% A continuacion se realiza la transformada de la matriz H
% En este caso se va a operar en bloques de 8x8
for i=0:8:r-8
    for j=0:8:c-8
        p = i+1;
        q = j+1;
        yo (p:p+7, q:q+7) = Ho' * imagen(p:p+7, q:q+7)*Ho;
        y (p:p+7, q:q+7) = H' * imagen(p:p+7, q:q+7)*H;
    end
end
```

*Ilustración 2 - Transformada de la matriz Haar*

```
for i=0:8:r-8
    for j=0:8:c-8
        p = i + 1;
        q = j + 1;
        zo(p:p+7,q:q+7) = Ho * yo(p:p+7,q:q+7)*Ho';
        z(p:p+7,q:q+7) = inv(H') * y(p:p+7,q:q+7)*inv(H);
    end
end
```

*Ilustración 3 - Reconstrucción de la imagen comprimida*

# Simulación

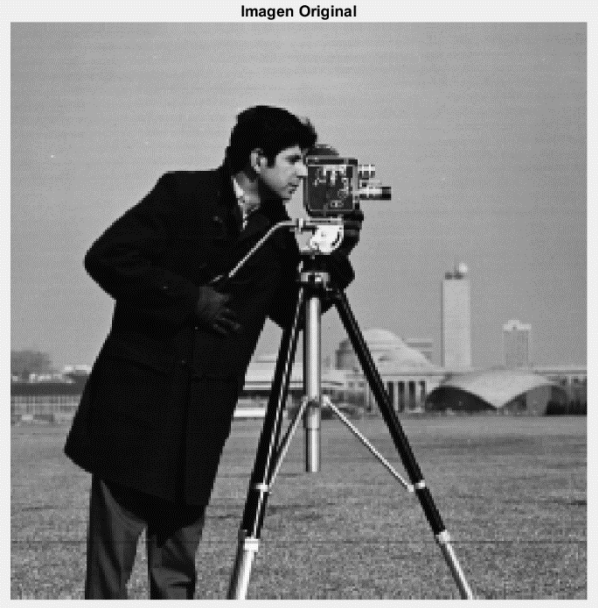


*Ilustración 4 - Imagen original "tif.png"*


*Ilustración 5 - Imagen comprimida con factor delta = 0.05*


*Ilustración 6 - Imagen comprimida con factor delta = 0.005*

## Conclusión

Tras realizar esta práctica pudimos observar cómo se puede comprimir una imagen utilizando únicamente una wavelet, en este caso la de Haar. Este método aprovecha las propiedades de las wavelets para descomponer la imagen en componentes de diferentes frecuencias, permitiendo la eliminación de los componentes que no son significativas (ruido) y la retención de las características esenciales de la imagen. A pesar de la eliminación de algunos detalles menores, la imagen principal se conservó adecuadamente, lo que demuestra la efectividad del método de compresión wavelet. Con esos resultados es fácil saber por qué se usa este método en sitios web para cargar imágenes de manera sencilla y eficiente sin afectar su desempeño, así como en como en el almacenamiento de grandes bases de datos de imágenes y la transmisión de imágenes en tiempo real.

## Referencias

[1] Wikipedia contributors. (2024c, junio 15). Wavelet. Wikipedia. https://en.wikipedia.org/wiki/Wavelet

[2] De B Harrington, P. (2016). Multivariate Curve Resolution of Wavelet Compressed Data. En Data handling in science and technology (pp. 311-332). https://doi.org/10.1016/b978-0-444-63638-6.00009-7

[3] Wavelet Compression | CloudInary. (2024, 6 junio). Cloudinary. https://cloudinary.com/glossary/wavelet-compression

# Anexo

## Código.

```
% Entregable    17 – Wavelet Compression
% Grupo         5CV1
% Equipo        7
% Profesor: José Antonio Flores Escobar
% Integrantes:
%              Hernandez Rodriguez Juan Uriel
%              Vergara Martinez Brenda
%              García Quiroz Gustavo Ivan
%              Gutiérrez Jiménez Cinthia Nayelli
%              Ramírez Carrillo José Emilio
%              Iturbide Serrano Uriel
% Compresion de imagenes por medio de una wavelet
% La wavelet a utilizar es la de Haar
clear all;
close all;
clc;

% Lectura de la imagen original
imagen = double(imread("tif.png"));
%figure(1)
%imshow(imagen);

% Para este ejemplo, el valor de delta debe estar entre 0 y 1
%delta = 0.0001;
delta = 0.05;

% Definir la wavelet Haar
% Mediante 3 matrices: H1, H2, H3
H1=[0.5 0 0 0 0.5 0 0 0;0.5 0 0 0 -0.5 0 0 0;0 0.5 0 0 0 0.5 0 0 ;0 0.5 0 0 0 -0.5 0
0 ;0 0 0.5 0 0 0 0.5 0;0 0 0.5 0 0 0 -0.5 0;0 0 0 0.5 0 0 0 0.5;0 0 0 0.5 0 0 0 -
0.5;];
H2=[0.5 0 0.5 0 0 0 0 0;0.5 0 -0.5 0 0 0 0 0;0 0.5 0 0.5 0 0 0 0;0 0.5 0 -0.5 0 0 0
0;0 0 0 0 1 0 0 0;0 0 0 0 0 1 0 0;0 0 0 0 0 0 1 0;0 0 0 0 0 0 0 1;];
H3=[0.5 0.5 0 0 0 0 0 0;0.5 -0.5 0 0 0 0 0 0;0 0 1 0 0 0 0 0;0 0 0 1 0 0 0 0;0 0 0 0
1 0 0 0;0 0 0 0 0 1 0 0;0 0 0 0 0 0 1 0;0 0 0 0 0 0 0 1;];

% Normalizar cada una de las columnas de las 3 matrices
% Con esto se asegura que los resultados de la normalizacion
% son ortonormales para cada columna de la matriz
H1o = (H1.*(2^0.5));
H2o = (H2.*(2^0.5));
H3o = (H3.*(2^0.5));

% Muliplicarlas por las tres matrices
H = (H1o*H2o*H3o);
% Esta seria la matriz resultante, normalizada
Ho = norm(H);
%Multiplicar las tres matrices de Haar
H = (H1*H2*H3);
```

```
% Tamano de la imagen
len = length(size(imagen));

% ESTE EJEMPLO NO APLICA PARA UNA IMAGEN RGB
if len~=2
    error('Para este ejemplo necesita una imagen en escala de grises')
    % Si se desea aplicar el ejemplo a una imagen RGB, usar un ejemplo con
    % wavelet RGB (haar_wv_rgb)
end

% Crear dos imagenes con valores 0
yo = zeros(size(imagen));
y = yo;
% Se obtienen los renglones y las columnas de la imagen original
[r,c] = size(imagen);

% A continuacion se realiza la transformada de la matriz H
% En este caso se va a operar en bloques de 8x8
for i=0:8:r-8
    for j=0:8:c-8
        p = i+1;
        q = j+1;
        yo (p:p+7, q:q+7) = Ho' * imagen(p:p+7, q:q+7)*Ho;
        y (p:p+7, q:q+7) = H' * imagen(p:p+7, q:q+7)*H;
    end
end

% Mostrar la imagen original
figure(1);
imshow(imagen/255)
title('Imagen Original');

% Determinar el numero de valores no-ceros de y
n1 = nnz(y);

% Crear una nueva imagen
zo = yo;
m = max(max(yo));

yo = yo/m;
% Valores dentro de +delta y -delta en y, se reemplazan por ceros
% Esto es un parametro que ayuda en la compresion
yo(abs(yo)<delta) = 0;
yo = yo*m;

% Crear una nueva imagen
z = y;
y = y/m;
% Valores dentro de +delta y -delta en y, se reemplazan por ceros
% Esto es un parametro que ayuda en la compresion
y(abs(y)<delta) = 0;
y = y*m;

% Determinar el numero de valores no-ceros de y
```

```
n2 = nnz(y);

%% SEGUNDA PARTE DE LA PRACTICA
for i=0:8:r-8
    for j=0:8:c-8
        p = i + 1;
        q = j + 1;
        zo(p:p+7,q:q+7) = Ho * yo(p:p+7,q:q+7)*Ho';
        z(p:p+7,q:q+7) = inv(H') * y(p:p+7,q:q+7)*inv(H);
    end
end
figure(2);
imshow(z/255);
title("Imagen Comprimida")

% Por ultimo, obtener un formato de compresion aproximado
factorcomp = n2/n1;

% Guardar las imagenes
imwrite(imagen/255, 'original.tif');
imwrite(z/255, 'comprimida.tif');

plot(ifftfhat2); title("Señal reconstruida a partir del espectro ruidoso");
```