Instituto Politécnico Nacional

Escuela Superior De Cómputo

Entregable 4

DFT Imagen

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

Introducción 1

Marco teórico 2

Transformada de Fourier discreta 2

Definición 2

Propiedades 3

La DFT unitaria 3

Desarrollo 5

Compresión de una imagen 5

Expansión de una imagen 6

Simulación 9

Compresión de una imagen 9

Expansión de una imagen 9

Conclusión 11

Referencias 12

Anexo. Código. 13

Introducción

Por supuesto, aquí tienes una introducción para el proyecto "Entregable 4 DFT Imagen":

El procesamiento digital de imágenes es un campo fundamental en la ingeniería moderna, con aplicaciones que abarcan desde la compresión de datos hasta el análisis médico y la visión por computadora. Este proyecto, titulado "Entregable 4 DFT Imagen", se enfoca en explorar los conceptos básicos de la compresión y expansión de imágenes digitales utilizando MATLAB como herramienta de programación.

El objetivo principal de este trabajo es desarrollar y analizar dos algoritmos: uno para la compresión de imágenes y otro para su posterior expansión. Estos algoritmos se basan en técnicas de promediado de píxeles y replicación, respectivamente, permitiendo manipular la resolución y el tamaño de las imágenes de manera controlada.

A través de esta práctica, se busca no solo implementar estos algoritmos, sino también comprender los efectos que tienen sobre la calidad de la imagen y el tamaño del archivo resultante. Este entendimiento es crucial en un mundo donde la optimización del almacenamiento y la transmisión de datos visuales es cada vez más importante.

El proyecto utiliza una imagen de prueba estándar (Lena) y permite al usuario especificar el grado de compresión y expansión, proporcionando así una experiencia interactiva y educativa. Al finalizar, se espera que los participantes hayan adquirido una comprensión práctica de los desafíos y compromisos involucrados en el procesamiento de imágenes digitales.

Marco teórico

Transformada de Fourier discreta

La **transformada discreta de Fourier** o **DFT** (Discrete Fourier Transform) es un tipo de transformada discreta utilizada en el análisis de Fourier. Transforma una función matemática en otra, obteniendo una representación en el dominio de la frecuencia, siendo la función original una función en el dominio del tiempo. Pero la DFT requiere que la función de entrada sea una secuencia discreta y de duración finita. Dichas secuencias se suelen generar a partir del muestreo de una función continua, como puede ser la voz humana. Al contrario que la transformada de Fourier en tiempo discreto (DTFT), esta transformación únicamente evalúa suficientes componentes frecuenciales para reconstruir el segmento finito que se analiza. Utilizar la DFT implica que el segmento que se analiza es un único período de una señal periódica que se extiende de forma infinita; si esto no se cumple, se debe utilizar una ventana para reducir los espurios del espectro. Por la misma razón, la DFT inversa (IDFT) no puede reproducir el dominio del tiempo completo, a no ser que la entrada sea periódica indefinidamente. Por estas razones, se dice que la DFT es una transformada de Fourier para análisis de señales de tiempo discreto y dominio finito. Las funciones sinusoidales base que surgen de la descomposición tienen las mismas propiedades.

Los algoritmos FFT se utilizan tan habitualmente para calcular DFTs que el término FFT muchas veces se utiliza en lugar de DFT en lenguaje coloquial. Formalmente, hay una diferencia clara: DFT hace alusión a una transformación o función matemática, independientemente de cómo se calcule, mientras que FFT se refiere a una familia específica de algoritmos para calcular DFTs.

Definición

La secuencia de N números complejos $x_0, ..., x_{N-1}$ se transforma en la secuencia de N números complejos $X_0, ..., X_{N-1}$ mediante la DFT con la fórmula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \dots, N-1$$

donde i es la unidad imaginaria y $e^{-\frac{2\pi i}{N}}$ es la N-ésima raíz de la unidad. (Esta expresión se puede escribir también en términos de una matriz DFT; cuando se escala de forma apropiada se convierte en una matriz unitaria y X_k puede entonces ser interpretado como los coeficientes de x en una base ortonormal.)

La transformada se denota a veces por el símbolo F, igual que en $\mathbf{X} = \mathcal{F}\{\mathbf{x}\} \circ \mathcal{F}(\mathbf{x}) \circ \mathcal{F}\mathbf{x}$.

La transformada inversa de Fourier discreta (IDFT) viene dada por

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \qquad n = 0, \ldots, N-1.$$

Una descripción simple de estas ecuaciones es que los números complejos X_k representan la amplitud y fase de diferentes componentes sinusoidales de la señal de entrada x_n. La DFT calcula X_k a partir de x_n, mientras que la IDFT muestra cómo calcular x_n como la suma de componentes sinusoidales $\left(\frac{1}{N}\right) X_k e^{\frac{2\pi i}{N} kn}$ con una frecuencia de $\frac{k}{N}$ ciclos por muestra.

Propiedades

Tabla 2: PROPERTIES OF THE DTFT

Propiedad	Secuencia	Transformada de Fourier
	$x[n]$	$X(\Omega)$
	$x_1[n]$	$X_1(\Omega)$
	$x_2[n]$	$X_2(\Omega)$
Periodicidad	$x[n]$	$X(\Omega + 2\pi) = X(\Omega)$
Linealidad	$a_1 x_1[n] + a_2 x_2[n]$	$a_1 X_1(\Omega) + a_2 X_2(\Omega)$
Desplazamiento en el tiempo	$x[n - n_0]$	$e^{-j\Omega n_0} X(\Omega)$
Desplazamiento en la frecuencia	$e^{j\Omega_0 n} x[n]$	$X(\Omega - \Omega_0)$
Conjugación	$x^*[n]$	$X^*(-\Omega)$
Inversión del tiempo	$x[-n]$	$X(-\Omega)$
Escalamiento en el tiempo	$x_{(m)}[n] = \begin{cases} x[n/m] & \text{si } n = km \\ 0 & \text{si } n \neq km \end{cases}$	$X(m\Omega)$
Diferenciación en la frecuencia	$nx[n]$	$j\frac{dX(\Omega)}{d\Omega}$

La DFT unitaria

Otra forma de interpretar la DFT es dándose cuenta de que puede expresarse como una matriz de Vandermonde:

$$\mathbf{F} = \begin{bmatrix} \omega_N^{0\cdot 0} & \omega_N^{0\cdot 1} & \dots & \omega_N^{0\cdot (N-1)} \\ \omega_N^{1\cdot 0} & \omega_N^{1\cdot 1} & \dots & \omega_N^{1\cdot (N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_N^{(N-1)\cdot 0} & \omega_N^{(N-1)\cdot 1} & \dots & \omega_N^{(N-1)\cdot (N-1)} \end{bmatrix}$$

donde

$$\omega_N = e^{-2\pi i/N}$$

es una raíz de la unidad. La transformada inversa viene entonces dada por la inversa de la matriz anterior:

$$F^{-1} = \frac{1}{N} F^*$$

Desarrollo

El proyecto DFT de una imagen consta de dos programas principales desarrollados en MATLAB: "comprimir.m" y "expandir.m". Estos programas tienen como objetivo manipular imágenes mediante técnicas de compresión y expansión.

Compresión de una imagen

El primer programa, "comprimir.m", se encarga de reducir el tamaño de una imagen. Comienza leyendo una imagen de entrada en formato BMP y permite al usuario especificar cuántos píxeles desea promediar por renglón y por columna. El programa utiliza un algoritmo de promediado de píxeles para comprimir la imagen. Recorre la imagen original en bloques del tamaño especificado, calcula el promedio de los valores de píxel en cada bloque y asigna este valor promedio a un único píxel en la imagen comprimida. El resultado es una imagen de menor resolución que se guarda con el nombre "imagenfinalcomprimida.bmp".

```
% Entregable    4 - DFT Imagen
% 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

%Programa: comprimir.m
%Descripción: Comprime una imagen, el usuario especifica cuantos pixeles
%desea comprimir por renglon y por columna, en caso de no querer comprimir
%el numero que se introduce es 1. Los numeros a introducir deben ser y 2, 4, 8, 16, 32, 64, 128
clear all;
clc;
images=imread('lena512x512.bmp');
tam=size(images);
images=double(images);
fprintf('\nIMAGEN DE %d x %d PIXELES\n\n',tam(1),tam(2));
numren=input('Cuantos pixeles se promedian por renglon?: ');
numcol=input('Cuantos pixeles se promedian por columna?: ');
picfinal=[];
for renglon=1:numren:tam(1)
    for columna=1:numcol:tam(2)
```

```
vector=[];
vector=images(renglon:renglon+(numren-1),columna:columna+(numcol-1));
picfinal((renglon+numren-1)/numren,((columna+numcol-1))/numcol)=sum(sum(vector))/(numcol*numren);
    end
end
picfinal=uint8(picfinal);
tam=size(picfinal);
fprintf('\nLa imagen que se ha comprimido es de %d x %d\n',tam(1), tam(2));
imwrite(picfinal,'imagenfinalcomprimida.bmp');
figure (1)
imshow(picfinal)
```

Figura 1 Programa comprimir.m

```
IMAGEN DE 512 x 512 PIXELES

Cuantos pixeles se promedian por renglon?: 2
Cuantos pixeles se promedian por columna?: 2

La imagen que se ha comprimido es de 256 x 256
>>
```

Figura 2 Simulación de la compresión de una imagen

Expansión de una imagen

El segundo programa, "expandir.m", realiza la operación inversa. Lee la imagen comprimida generada por el primer programa y permite al usuario especificar cuántos píxeles desea expandir por renglón y por columna. El algoritmo de expansión funciona replicando cada píxel de la imagen comprimida en un bloque de píxeles del tamaño especificado en la imagen expandida. Esto resulta

en una imagen de mayor resolución, aunque con una calidad visual reducida debido a la pérdida de información durante el proceso de compresión. La imagen expandida se guarda con el nombre "imagenfinalexpandida.bmp".

```
% Entregable    4 - DFT Imagen
% 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

%Programa: expandir.m
%Descripción: Expande una imagen, el usuario especifica cuantos pixeles
%desea expandir por renglon y por columna, en caso de no querer expandir
%el numero que se introduce es 1.
clear all;
clc;
images=imread('imagenfinalcomprimida.bmp');
tam=size(images);
images=double(images);
fprintf('\nIMAGEN DE %d x %d PIXELES\n\n',tam(1),tam(2));
numren=input('Cuantos pixeles se expanden por renglon?: ');
numcol=input('Cuantos pixeles se expanden por columna?: ');
picfinal=[];
for renglon=1:tam(1)
    for columna=1:tam(2)
vector=zeros(numren,numcol);
vector=vector+images(renglon,columna);
picfinal((renglon*numren)-(numren-1):renglon*numren,(columna*numcol)-(numcol-1):columna*numcol)=vector
    end
end
picfinal=uint8(picfinal);
tam=size(picfinal);
fprintf('\nLa imagen que se ha expandido es de %d x %d\n',tam(1), tam(2));
imwrite(picfinal,'imagenfinalexpandida.bmp');
figure (1)
imshow(picfinal)
```

Figura 3 Programa expandir.m

```
IMAGEN DE 256 x 256 PIXELES

Cuantos pixeles se expanden por renglon?: 2
Cuantos pixeles se expanden por columna?: 2

La imagen que se ha expandido es de 512 x 512
>>
```

Figura 4 Simulación expansión de una imagen

Simulación

Compresión de una imagen

```
IMAGEN DE 512 x 512 PIXELES

Cuantos pixeles se promedian por renglon?: 2
Cuantos pixeles se promedian por columna?: 2

La imagen que se ha comprimido es de 256 x 256
>>
```

Figura 5 Simulación de la compresión de una imagen

Expansión de una imagen

```
IMAGEN DE 256 x 256 PIXELES

Cuantos pixeles se expanden por renglon?: 2
Cuantos pixeles se expanden por columna?: 2

La imagen que se ha expandido es de 512 x 512
>>
```

Figura 6 Simulación expansión de una imagen

Conclusión

Este proyecto demuestra la implementación práctica de técnicas básicas de compresión y expansión de imágenes utilizando MATLAB. A través de los programas desarrollados, se logró manipular eficazmente el tamaño y la resolución de imágenes digitales. La técnica de promediado de píxeles empleada en la compresión permite reducir significativamente el tamaño de la imagen, aunque a costa de cierta pérdida de detalle. Por otro lado, el proceso de expansión, si bien aumenta las dimensiones de la imagen, no puede recuperar la información perdida durante la compresión, resultando en una imagen de mayor tamaño pero con calidad reducida. Estos resultados subrayan el compromiso inherente entre el tamaño del archivo y la calidad de la imagen en los procesos de compresión digital, y proporcionan una valiosa perspectiva sobre los fundamentos del procesamiento de imágenes y sus aplicaciones prácticas en el campo de la ingeniería y la informática.

Referencias

[1] L. Tan, *Digital Signal Processing*. Academic Press, 2008.

[2] Wikipedia contributors, “Transformada de Fourier discreta”, *Wikipedia, The Free Encyclopedia*. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Transformada_de_Fourier_discreta&oldid=148495486.

Anexo. Código.

```
% Entregable    4 – DFT Imagen
% 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

%Programa: comprimir.m
%Descripción: Comprime una imagen, el usuario especifica cuantos pixeles
%desea comprimir por renglon y por columna, en caso de no querer comprimir
%el numero que se introduce es 1. Los numeros a introducir deben ser y 2, 4, 8,
16, 32, 64, 128
clear all;
clc;
images=imread('lena512x512.bmp');
tam=size(images);
images=double(images);
fprintf('\nIMAGEN DE %d x %d PIXELES\n\n',tam(1),tam(2));
numren=input('Cuantos pixeles se promedian por renglon?: ');
numcol=input('Cuantos pixeles se promedian por columna?: ');
picfinal=[];
for renglon=1:numren:tam(1)
    for columna=1:numcol:tam(2)
vector=[];
vector=images(renglon:renglon+(numren-1),columna:columna+(numcol-1));
picfinal((renglon+numren-1)/numren,((columna+numcol-
1))/numcol)=sum(sum(vector))/(numcol*numren);
    end
end
picfinal=uint8(picfinal);
tam=size(picfinal);
fprintf('\nLa imagen que se ha comprimido es de %d x %d\n',tam(1), tam(2));
imwrite(picfinal,'imagenfinalcomprimida.bmp');
figure (1)
imshow(picfinal)
```

```
% Entregable    4 – DFT Imagen
% 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

%Programa: expandir.m
%Descripción: Expande una imagen, el usuario especifica cuantos pixeles
%desea expandir por renglon y por columna, en caso de no querer expandir
%el numero que se introduce es 1.
clear all;
clc;
images=imread('imagenfinalcomprimida.bmp');
tam=size(images);
images=double(images);
fprintf('\nIMAGEN DE %d x %d PIXELES\n\n',tam(1),tam(2));
numren=input('Cuantos pixeles se expanden por renglon?: ');
numcol=input('Cuantos pixeles se expanden por columna?: ');
picfinal=[];
for renglon=1:tam(1)
    for columna=1:tam(2)
        vector=zeros(numren,numcol);
        vector=vector+images(renglon,columna);
        picfinal((renglon*numren)-(numren-1):renglon*numren,(columna*numcol)-
(numcol-1):columna*numcol)=vector;
    end
end
picfinal=uint8(picfinal);
tam=size(picfinal);
fprintf('\nLa imagen que se ha expandido es de %d x %d\n',tam(1), tam(2));
imwrite(picfinal,'imagenfinalexpandida.bmp');
figure (1)
imshow(picfinal)
```