Compresión de modelos 3D usando Transformadas de Fourier

De los Santos, Pol

INS Jaume Vicens Vives, curso 2021-2022

Grupo: 2n de Bachillerato A

Tutoría: Albert Falgueras

Índice de contenidos

1. INTRODUCCIÓN

En este trabajo de recerca se explora las series de Fourier desde un punto de vista matemático. Junto con esta investigación teórica también se estudiará las aplicaciones prácticas del Análisis de Fourier, en concreto, siguiendo la pregunta: ¿Cómo se puede utilizar las Transformadas de Fourier para aproximar objetos 3D y cómo comparan con métodos tradicionales?

Como es una investigación matemática primero se plantearán inicialmente las Series de Fourier como concepto matemático, seguido de las Transformadas de Fourier, estudiando sus propiedades y como se define. A continuación, se usarán estos conocimientos para explicar sus diferentes aplicaciones, y seguidamente cumplir con una de ellas, la aplicación gráfica de las series de Fourier.

1.1. Objetivos

Los objetivos que quiero lograr a lo largo de este trabajo son los siguientes:

- Responder a las preguntas de investigación: “¿Que son las Series de Fourier?”, “¿Qué base matemática se encuentra detrás de las Series de Fourier?” y “¿Cómo se usan las Transformadas de Fourier para aproximar funciones?”
- Obtener los conocimientos necesarios para realizar el trabajo
- Aprender a organizar mi tiempo para hacer un trabajo externo
- Finalmente, hacer un trabajo del que este satisfecho

1.2. Metodología

He decidido dividir este trabajo en tres capítulos:

1. Marco Teórico: En este primer capítulo hay una introducción teórica de lo necesario para seguir el trabajo y de los conceptos que se usan a lo largo de este. Este apartado empieza con una definición de conceptos y seguidamente las propiedades de funciones que se usan durante el trabajo y son relevante a este.
2. Series Fourier: En este capítulo se contesta a las preguntas: “¿Que es el Análisis de Fourier?” y “¿Qué base matemática se encuentra detrás de las

Series de Fourier?". Se empieza con una breve introducción sobre su historia, seguida de una definición y a continuación, se explican las matemáticas detrás de este concepto.

3. Aplicaciones: El último capítulo de este trabajo empieza listando todas las posibles aplicaciones que tienen las Series de Fourier. Seguidamente, se responde a la pregunta: "¿Cómo se puede utilizar las Transformadas de Fourier para aproximar objetos 3D y cómo comparan con métodos tradicionales?" a través de un programa informático que aplica los conocimientos obtenidos en el capítulo anterior del trabajo.

Para poder representar las figuras y toda la información recopilada, se usarán una variedad de programas. Matlab, para representar figuras en el plano 3D, y para procesar la información obtenida se usarán programas escritos en C++.

1.3. Motivación personal

A principio de mi bachillerato, me encontré con un video en el que se podía ver una animación de la Serie Compleja de Fourier, en el que se unen muchos vectores que están girando a una frecuencia constante, para ver como la punta del ultimo vector dibuja una figura con el tiempo. Al ver este tipo de animación, decidí hacer recerca de las mates que había detrás. Sin saber dónde me metía, entre en el mundo del análisis de Fourier, el tipo de análisis que se usa más en el mundo moderno al procesar señales. Además, como había escoger un tema para el Trabajo de Recerca del instituto, decidí realmente entender y profundizar en este campo.

2. MARCO TEÓRICO

Seguidamente se esposarán todos los conocimientos previos necesarios para poder comprender lo que se tratara en el trabajo.

2.1. Funciones[1]

Dados dos conjuntos A y B, llamamos función a la correspondencia de A en la cual todos los elementos de A tienen a lo sumo una imagen en B, es decir una imagen o ninguna. Función real de variable real es toda correspondencia f que asocia a cada elemento de un determinado subconjunto de números reales, llamado dominio, otro número real.

$$f: D \to \mathbb{R}$$
$$x \to f(x) = y$$

El subconjunto en el que se define la función se llama dominio o campo existencia de la función. Se designa por D.

2.2.1. Impares y Pares [2]

Una función es par si para una función $f: A \to B, f(-x) = -f(x)$ para todo $x \in A$. También debe considerarse $-x \in A$.

Una función es par si para una función $f: A \to B, f(x) = f(-x)$ para todo $x \in A$. También se debe considerar $-x \in A$.

Integración de funciones parece impares sobre un dominio simétrico: Sea $p > 0$ un número fijo. Si f(x) es una función impar, entonces:

$$\int_{-p}^{p} f(x)\, dx = 0$$

Esto es debido a que la integral de la función en el periodo $[-p, 0]$ es igual a la integral de la funcion en el periodo $[0, p]$, pero con signo oposado. De esta manera se cancelan y queda como resultado 0.

Sea $p > 0$ un número fijo. Si f(x) es una función par, entonces:

$$\int_{-p}^{p} f(x)\, dx = 2\int_{0}^{p} f(x)\, dx$$

Esto es debido a que la integral de la función en el periodo $[-p, 0]$ es igual a la integral de la funcion en el periodo $[0, p]$. Entonces, la intefral en este dominio simetrico es igual al doble de la integral en al mitad de la smetria.

[1] www. superprof.es/apuntes/escolar/matematicas/calculo/funciones/funciones.html
[2] www.matesfacil.com/BAC/funciones/paridad/funcion-par-impar-paridad-propiedades-demostraciones.html

2.2.2. Periódicas[3]

Diremos que una función $f(x)$ es periódica cuando exista un número real T, tal que el valor de la función en el punto x coincida con el valor de la función en el punto $x + T$: $f(x) = f(x + T)$. Llamamos periodo al número real T que nos indica cada cuanto se repite la función.

2.2.3. Trigonométricas[4]

Las funciones trigonométricas f son aquellas que están asociadas a una razón trigonométrica.

Las razones trigonométricas de un ángulo α son las obtenidas entre los tres lados de un triángulo rectángulo. Es decir, las comparaciones por su cociente de sus tres lados a, b y c.

Durante el trabajo trabajaremos con 2 funciones trigonométricas:

Seno: El seno de un ángulo se define como la razón entre el cateto opuesto $(C.O.)$ y la hipotenusa (h) del triángulo rectángulo.

$$sen(x) = \frac{C.O.}{h}$$

La función $f(x) = sen(x)$ tiene un seguido de propiedades. Su dominio es de $x \in \mathbb{R}$ y su recorrido es de $x \in [-1, 1]$. Es una función periódica con periodo $T = 2\pi$, continua en todo su dominio e impar.

Coseno: El coseno de un ángulo se define como la razón entre el cateto contiguo $(C.C.)$ y la hipotenusa (h) del triángulo rectángulo.

$$\cos(x) = \frac{C.C.}{h}$$

La función $f(x) = cos(x)$ tiene un seguido de propiedades. Su dominio es de $x \in \mathbb{R}$ y su recorrido es de $x \in [-1, 1]$. Es una función periódica con periodo $T = 2\pi$, continua en todo su dominio y par.

[3] www.matematica.laguia2000.com/general/periodicidad-de-una-funcion
[4] www.universoformulas.com/matematicas/analisis/funciones-trigonometricas/

2.2. Números complejos

Al número $z = a + bi$ se le llama número complejo. En general, cualquier número complejo se denota por la letra z y donde i es la unidad imaginaria: $\sqrt{-1} = i$. Al número a se llama **parte real** del número complejo y se denota $a = Re(z)$, mientras que al número b se llama **parte imaginaria** del número complejo y se denota por $b = Im(z)$.

El conjunto de los números complejos se defino como: $\mathbb{C} = \{a + bi | a, b \in \mathbb{R}\}$[5]

En matemáticas, el **plano complejo** es una forma de visualizar y ordenar el conjunto de los números complejos. Puede entenderse como un plano cartesiano modificado, en el que la parte real está representada en el eje de abscisas y la parte imaginaria en el eje de ordenadas. De esta manera representas lo números complejos como vectores.[6]

2.3.1. Formula de Euler

La fórmula de Euler, atribuida a Leonhard Euler, establece que:

$$e^{i\varphi} = \cos(\varphi) + i\, sin(\varphi)$$

Para todo número real x, que representa un ángulo en el plano complejo. Aquí, e es la constate de Euler y i es la unidad imaginaria. Esta fórmula lo que nos dice es que $e^{i\varphi}$ es un vector de magnitud 1 con inicio en el centro de coordenadas y que tiene un ángulo φ (Figura 1). Así mismo, si $\varphi = \omega t$; donde t va variando, obtenemos un vector que rota sobre el circulo unitario del plano complejo.

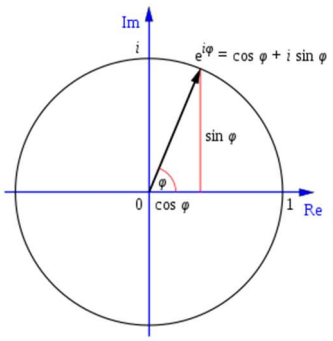


Figura 1: Circulo de la unidad para representar e^{ix}

2.3.2. Amplitud, Fase y Frecuencia de $e^{i\omega t}$

Nos referimos a el ángulo en que empieza a rotar como fase (ω), el módulo del vector como amplitud i a la rapidez del vector cuando gira como frecuencia (como de rápido varia t).

[5] www.superprof.es/apuntes/escolar/matematicas/aritmetica/complejos/numeros-complejos-resumen.html
[6] es.wikipedia.org/wiki/Plano_complejo

3. ANALISIS DE FOURIER

3.1. Introducción

Durante la extensión de este trabajo, trataremos dos temas, las series de Fourier y las transformadas de Fourier. Estos dos conceptos matemáticos se basan en una misma idea, representar un fragmento (o periodo) de una función como la suma de funciones trigonométricas (senos y cosenos). De esta manera obtenemos una aproximación de ese intervalo (o de toda la función si es periódica). Por un lado, las series de Fourier sirven para analizar funciones periódicas a través de la descomposición de dicha función en una suma infinita de funciones sinusoidales mucho más simples (como combinación de senos y cosenos con frecuencias enteras y su peso apropiado)[7]. En cambio, la transformada de Fourier es usada para funciones aperiódicas.

Estos dos conceptos constituyen una herramienta muy importante en la solución de problemas físicos en las que intervienen ecuaciones diferenciales ordinarias y parciales. El análisis de Fourier, es usado en muchas ramas de la ingeniería, como el análisis vibratorio, acústica, óptica, procesamiento de imágenes y señales, y comprensión de datos.

Antes de empezar, hemos de clarificar que el trabajo estará enfocado en el análisis de Fourier en un tiempo discreto (DT), a diferencia de un tiempo continuo (CT). Esto es importante de clarificar ya que existe una variación según el dominio de tiempo en ambos, la serie de Fourier y las transformadas de Fourier.

3.1.1. Historia

La introducción al Análisis de Fourier fue uno de los mayores avances jamás realizados en la física matemática y en sus aplicaciones en la ingeniería, ya que es utilizado en nuestro día a día, aunque no nos demos cuenta. [8]

El Análisis de Fourier fue aplicado por primera vez en el siglo XVIII por Jean-Baptiste Joseph Fourier con el propósito de resolver la ecuación de calor en 1807 *Mémoire sur la propagation de la chaleur dans les corps solides* y la publicación de la obra *Théorie*

[7] www. es.wikipedia.org/wiki/Serie_de_Fourier
[8] dmae.upct.es/~paredes/am_ti/apuntes/Tema%202.%20Series%20y%20transformadas%20de%20Fourier.pdf

analytique de la chaleur en 1822. Aunque fue inspirado por el trabajo anterior de Daniel Bernoulli. Leonhard Euler o Jean Le Rond d'Alembert. [9]

La ecuación de calor es una ecuación diferencial parcial, a la cual Fourier dio solución. Antes del trabajo de Fourier no se conocía ninguna solución excepto en los casos excepcionales en que la fuente de calor tomaba una forma de una oda sinusoidal o cosenoidal. Fourier, puesto de forma simple, decidió expresar cualquier otra función como la superposición o combinación lineal de simple ondas sinusoidales y cosenoidales., y obtener la solución a partir de estas. Esta sobreposición o combinación lineal de ondas recibe el nombre de series de Fourier.[10]

3.2. Series de Fourier[11]

Empecemos primero con un ejemplo. Consideremos la función definida de la siguiente manera:

$$f(x)\ en\ su\ periodo\ de\ 2 = \begin{cases} -1 & si\ -1 < x < 0 \\ 1 & si\ \ 0 < x < 1 \end{cases}$$ (Figura 2)

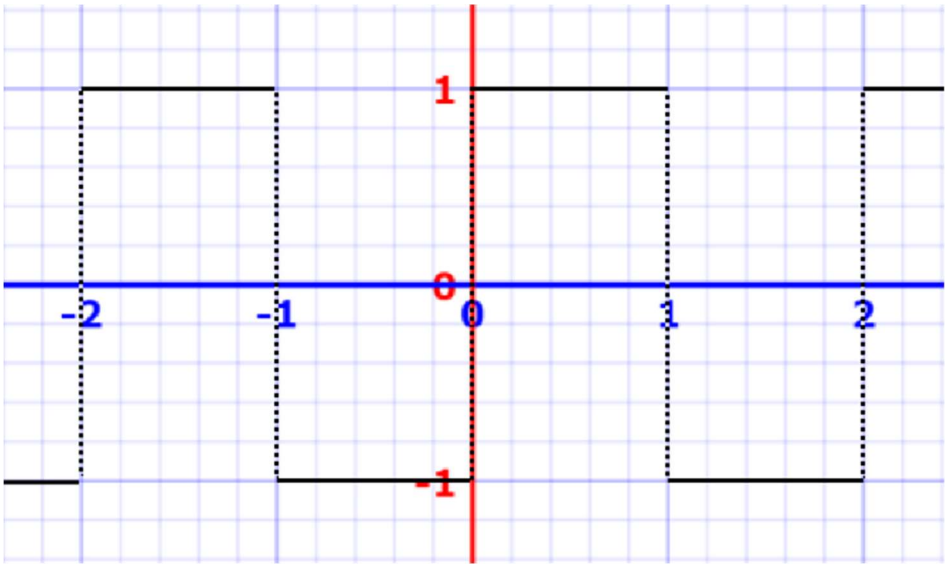


Figura 2: representación de f(x), conocida como "square wave function"

Considerando el intervalo $[-1, 1]$, nuestro objetivo con la Serie de Fourier es expresar esta función como la suma senos y cosenos cuya frecuencia es un numero entero,

[9] elpais.com/ciencia/2020-10-01/la-serie-que-cambio-el-mundo.html
[10] en.wikipedia.org/wiki/Fourier_series#History

pero, como es esto posible si esta función contiene ni una forma ondulatoria, e inclusión es discontinua.

Es importante remarcar que ningún tipo de suma finita de funciones ondulatorias llegaría a ser igual a este intervalo (Figura 3), pero la Serie de Fourier considera una suma infinita que se aproxima a la función original, por lo tanto, en su límite, es igual. En el caso de esta función de onda cuadrada obtenemos que la función es igual a:

$$f(x) = \frac{4}{\pi}\sum_{n=1}^{\infty} \frac{sen(\pi(2n-1)x)}{2n-1} = \frac{4}{\pi}\left(\frac{sen(\pi x)}{1} + \frac{sen(3\pi x)}{3} + \frac{sen(5\pi x)}{5} + \cdots\right)$$

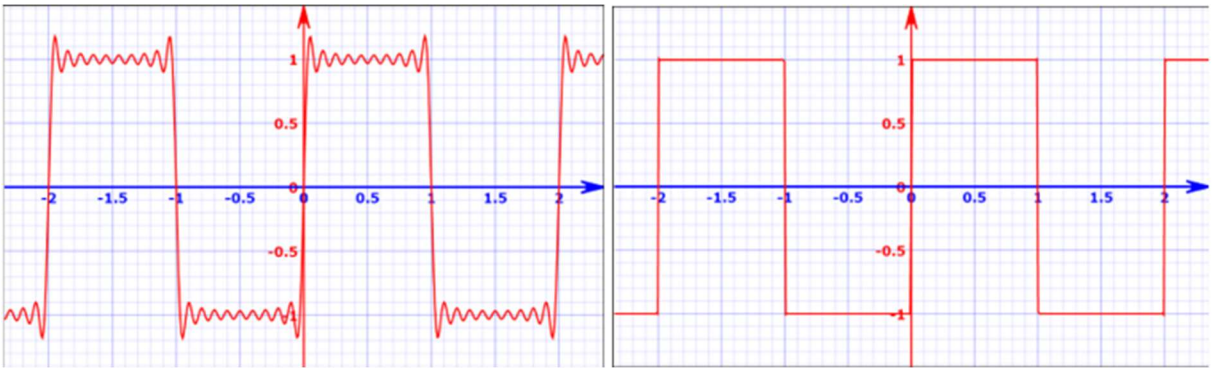


Figura 3: Aproximación de f(x) sumando 15 y 2000 funciones sinusoidales respectivamente.

Esta serie infinita se obtiene con la Serie de Fourier. esta afirma que si tenemos una función $f(t)$, de variable t, que es integrable en el intervalo $[t_0 - T/2, t_o + T/2\,]$ entonces se puede obtener el desarrollo de la serie de Fourier de f en ese intervalo. Fuera del intervalo la serie es periódica, con período T. Si $f(t)$ es periódica en toda la recta real, la aproximación por series de Fourier también será válida en todos los valores de t. Luego la serie de Fourier asociada a $f(t)$ es:[12]

$$f(t) = a_o + \sum_{n=1}^{\infty} (a_n \cdot \cos(n\omega_0 t) + b_n \cdot \sin(n\omega_0 t))$$

Donde a_0 es el promedio de la función $f(t)$ y es igual al área total en un periodo de tiempo dividido entre un periodo. Para calcular esto, integramos la función periódica en un periodo T.

$$a_0 = \frac{1}{T}\int_T^b f(t)\,dt$$

[12] es.wikipedia.org/wiki/Serie_de_Fourier

Donde a_n y b_n se denominan coeficientes de Fourier de la serie de Fourier $f(t)$, y se obtienen realizando la integral de la función $f(t)$ multiplicado por el coseno o el seno respectivo, de $\omega_n t$, en un periodo de tiempo y dividiendo por $T/2$.

$$a_n = \frac{2}{T}\int_T^b f(t)\cdot\cos(\omega_0 t)\,dt, \quad b_n = \frac{2}{T}\int_T^b f(t)\cdot\sin(\omega_0 t)\,dt$$

Usando la formula trigonométrica (1) de la Serie de Fourier, es necesario calcular los 3 coeficientes para así obtener la Serie de Fourier de la función periódica $f(t)$.

3.2.1. Condiciones de Dirichlet[13]

Anteriormente hemos dicho que las Series de Fourier se usa para funciones periódicas, pero esta afirmación no siempre es válida. Para que exista una Serie de Fourier la función ha de primero cumplir con unas condiciones que reciben el nombre de “Las condiciones de Dirichlet”:

1. In caso de que sea periódica, el número máximo y mínimo del rango del tiempo del periodo ha de ser finito.
2. Esta función periódica ha de tener un numero finito de discontinuidad en su periodo.
3. La función ha de ser absolutamente integrable en su rango de periodo:

$$\int |F_1(t)|\,dt < \infty$$

3.3. Formulación en forma compleja

Teniendo en cuenta la formula establecida anteriormente, la podemos transformar para obtener una forma de la serie de Fourier en el plano complejo:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n\cdot\cos(n\omega_0 t) + b_n\cdot\sin(n\omega_0 t))$$

Podemos usar la fórmula de Euler para substituir los cosenos y los senos de la fórmula:

$$\cos(x) = \frac{1}{2}e^{ix} + \frac{1}{2}e^{-ix}; \quad \sin(x) = -\frac{1}{2}ie^{ix} + \frac{1}{2}ie^{-ix};$$

[13] app.box.com/s/thpam69f04u5ygntpnemzxtme8hcig1b

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cdot \left(\frac{1}{2} e^{in\omega_0 t} + \frac{1}{2} e^{-in\omega_0 t} \right) + b_n \cdot \left(-\frac{1}{2} i e^{in\omega_0 t} + \frac{1}{2} i e^{-in\ _0 t} \right) \right] \Rightarrow$$

$$\Rightarrow f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[\frac{a_n}{2} \cdot \left(e^{in\omega_0 t} + e^{-in\omega_0 t} \right) - \frac{ib_n}{2} \cdot \left(e^{in\omega_0 t} - e^{-in\omega_0 t} \right) \right] =$$

$$= \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[\left(\frac{a_n}{2} - \frac{ib_n}{2} \right) e^{in\omega_0 t} \right] + \sum_{n=1}^{\infty} \left[\left(\frac{a_n}{2} + \frac{ib_n}{2} \right) e^{-in\omega_0 t} \right]$$

Considerando el ultimo sumatorio, podemos hacer una substitución de $n \to -n$ obtenemos:

$$\sum_{n=1}^{\infty} \left[\left(\frac{a_n}{2} + \frac{ib_n}{2} \right) e^{-in\omega_0 t} \right] = \sum_{n=-\infty}^{-1} \left[\left(\frac{a_{-n}}{2} + \frac{ib_{-n}}{2} \right) e^{in\omega_0 t} \right]$$

Haciendo una substitución de c_n donde:

$$c_n = \begin{cases} \frac{a_0}{2} & \text{n} = 0 \\ \frac{a_n - ib_n}{2} & n > 0 \\ \frac{a_{-n} + ib_{-n}}{2} & n < 0 \end{cases}$$

Notemos que de esta manera obtenemos que c_n y c_{-n} son conjugados complejos, por tanto $c_n = c_{-n}^*$, debido a que tenemos un cambio de signo de la parte imaginaria del número. Por tanto, la constante c_n es un numero complejo excepto en $n = 0$, y de esta manera obtenemos la Serie de Fourier en forma compleja.

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{in\omega_0 t}$$

Para obtener el coeficiente c_n, en un principio multiplicamos la expresión anterior por $e^{-im\omega_0 t}$ y realizamos una integral definida de dominio $[-\pi, \pi]$

$$f(t) \cdot e^{-im\omega_0 t} = \left(\sum_{n=-\infty}^{\infty} c_n e^{in\omega_0 t} \right) \cdot e^{-im\omega_0 t};$$

$$\int_{-\pi}^{\pi} f(t) \cdot e^{-im\omega_0 t}\, dt = \int_{-\pi}^{\pi} \left[\sum_{n=-\infty}^{\infty} \left(c_n e^{in\omega_0 t} \cdot e^{-im\omega_0 t} \right) \right] dt;$$

Asumiendo que la función $f(t)$ converge, podemos que la suma de las integrales es la integral de las sumas, entonces:

$$\int_{-\pi}^{\pi} f(t)\cdot e^{-im\omega_0 t}\,dt = \sum_{-n}^{n}\left[c_n\int_{-\pi}^{\pi}\left(e^{in\omega_0 t}\cdot e^{-im\omega_0 t}\right)dt\right] =$$

$$= \sum_{-n}^{n}\left[c_n\int_{-\pi}^{\pi}\left(e^{i(n-m)\omega_0 t}\right)dt\right]$$

Fijémonos en la integral de la derecha, esta solo tiene dos soluciones:

$$\int_{-\pi}^{\pi}\left(e^{i(n-m)\omega_0 t}\right)dt = \begin{cases} 0\ si\ n \neq m \\ 2\pi\ si\ n = m \end{cases}$$

Esto es debido a que la integral definida de e^{ix} en un domino de $[-\pi,\pi]$ es igual a 0, entonces, en el único punto que esto no es así es cuando $x = 0$. Entonces, teniendo en cuenta lo dio expandiendo el sumatorio tenemos lo siguiente:

$$\int_{-\pi}^{\pi} f(t)\cdot e^{-im\omega_0 t}\,dt = \cdots + c_{n-2}\cdot 0 + c_{m-1}\cdot 0 + c_m\cdot 2\pi + c_{m+1}\cdot 0 + \cdots = 2\pi\cdot c_m$$

Siendo c_m el coeficiente en que m y n se cancelan. Notemos que así $m = n$, y por tanto, si queremos obtener el valor de c_n, substituimos m por n y obtenemos:

$$c_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)\cdot e^{-in\omega_0 t}\,dt$$

Podemos ver que todos los coeficientes de la serie de Fourier compleja corresponden a la formula encontrada. Notemos que esto no solo se cumple con periodo 2π, si no que por todo periodo debido a que cualquier integral en un periodo simétrico de e^{ix} es igual a 0:

$$\int_{-p}^{p} e^{ix} = 0$$

Y como consecuencia obtenemos las mismas fórmulas sirven para un periodo T (seguiremos usando 2π para que sea más fácil de comprender):

$$c_n = \frac{1}{T}\int_{-T/2}^{T/2} f(t)\cdot e^{-in\omega_0 t}\,dt$$

Aun la definición anterior de c_n a trozos, podemos obtener una fórmula general de esta constante ya que:

Par n positivo, y sustituyendo por su fórmula obtenemos:

$$c_n = \frac{1}{2}(a_n - ib_n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)[\cos(n\omega_0 t) - isen(n\omega_0 t)]\,dt = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)\cdot e^{-in\omega_0 t}\ dt$$

Para un n negativo, repetimos el proceso y obtenemos:

$$c_n = \frac{1}{2}(a_{-n} + ib_{-n}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)[\cos(-n\omega_0 t) + isen(-n\omega_0 t)]\,dt$$

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)\cdot e^{-in\omega_0 t}\,dt$$

Finalmente, para $n = 0$:

$$c_0 = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)\,dt = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(t)\cdot e^{-it\omega_0\cdot 0}\,dt$$

Entonces, para todo c_n tenemos la misma fórmula.

Ahora notemos que implica la serie de Fourier. Si nos fijamos en la fórmula de Euler, nos está diciendo que cualquier función $f(t)$ que cumpla con los requisitos, se puede expresar como la suma de vectores modificados por un coeficiente c_n rotando en el plano complejo.

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{in\omega_0 t}$$

Así que, c_n modifica la magnitud y el ángulo inicial de cada uno de los vectores con que rota a una velocidad constante determinada por ω_0. Como ejemplo, imaginemos que uno de estos vectores empiece a 45 grados y que tenga una magnitud de 0.5 en vez de 1, podemos hacer lo siguiente:

$$c_n e^{in\omega_0 t} \rightarrow 0.5(e^{(\pi/4)i})e^{in\omega_0 t}$$

Esto se hace para cada uno de los vectores que componen la función.

Aquí termina nuestro estudio teórico de las Series de Fourier, pero para computaciones, ya que un ordenador no puede realizar una integral, usaremos lo que es conocido como las Transformadas de Fourier.

3.4. Transformadas de Fourier[14] [15]

Como mencionado anteriormente, las Series de Fourier solo sirven para funciones periódicas. Como consecuencia, cuando se aplica los conceptos establecidos en el Análisis de Fourier, normalmente se utiliza la Transformada de Fourier, que sirve para

[14] en.wikipedia.org/wiki/Discrete_Fourier_transform
[15] www.sciencedirect.com/topics/engineering/discrete-fourier-transform

funciones aperiódicas. Igual que la Serie de Fourier en su forma compleja, la Transformada de Fourier es una transformada de carácter complejo. De forma similar que la Serie de Fourier, la Transformada de Fourier en su forma discreta se trata de una transformación lineal:

$$f: \mathbb{C}^N \to \mathbb{C}^N$$

Donde $\mathbb{C}$ denota el conjunto de números complejos. Entonces, la transformada discreta de Fourier (DFT), transforma una secuencia de señales separados de manera uniforme $x_n \coloneqq x_0, x_1, \dots, x_{N-1}$ a coeficientes de naturaleza compleja $X_k \coloneqq X_0, X_1, \dots, X_{N-1}$, que incluyen información de tanto de amplitud como de fase. Entonces, la DFT transforma una señal en el domino del tiempo a dominio de frecuencia. La transformada discreta de Fourier se define como:

$$\boxed{X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x_n \left[\cos\left(\frac{2\pi kn}{N}\right) - i\,\sin\left(\frac{2\pi kn}{N}\right)\right]}$$

$$N = numero\ de\ muestras$$

$$n = muestra\ actual$$

$$k = frecuencia\ actual\ k \in [0, N-1]$$

$$x_n = valor\ de\ la\ senal\ en\ la\ muestra\ n$$

$$X_k = coeficientes\ de\ la\ DFT$$

La transformada de Fourier, a veces se puede denotar con la $\mathcal{F}$, así que $X = \mathcal{F}\{x\}$.
Así mismo, existe otra transformada de Fourier conocida como la transformada inversa (IDFT). Esta hace lo contrario que la DFT, usa los mismos coeficientes que en la DFT para pasar del dominio de frecuencia al dominio del tiempo.

$$\boxed{x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k \cdot e^{\frac{i2\pi kn}{N}} = \frac{1}{N}\sum_{k=0}^{N-1} X_k \cdot \left[\cos\left(\frac{2\pi kn}{N}\right) + i\,\sin\left(\frac{2\pi kn}{N}\right)\right]}$$

Si aplicamos una DFT a una señal y después usas los coeficientes y se realiza una IDFT, obtendremos la misma señal que el inicio.

4. APLICACIONES[16]

La transformada discreta de Fourier (DFT) se aplica ampliamente al procesamiento d datos discretos en ciencia e ingeniería, por ejemplo, para procesamiento de señales ópticas y acústicas, análisis de espectro, etc. separación, medición de amplitud y fase, y filtrado de datos discretos. Por ejemplo, algunas aplicaciones de las transformadas y series de Fourier en el mundo real son[17]:

1. **Procesamiento de señales:** Puede que sea la mejor aplicación del análisis de Fourier. Se usa en casi todo dispositivo electrónico que recibe señales.
2. **Teoría de la aproximación:** Usamos series de Fourier para escribir una función como un polinomio trigonométrico.
3. **Teoría del control**: La serie de funciones de Fourier en la ecuación diferencial a menudo proporciona alguna predicción sobre el comportamiento de la solución de la ecuación diferencial. Son útiles para conocer la dinámica de la solución.
4. **Ecuaciones diferenciales parciales**: Se usamos para resolver ecuaciones diferenciales parciales de orden superior mediante el método de separación de variables.

La lista es casi infinita ya que las transformadas de Fourier están casi presentes en todo dispositivo electrónico que usamos hoy en día. Esto es debido a que cuando trabajamos con señales, la transformada de Fourier es imprescindible.

4.1. Computación

En este apartado vamos a usar la Transformada de Fourier para comprimir un tipo de archivo que describe un objeto 3D, un archivo PLY.

4.1.1. Archivo PLY [18]

Un simple objecte PLY consisteix en una col·lecció d'elements per a la representació de l'objecte. Consiste en una lista de (x, y, z) triples de vértices y una lista de caras que en realidad son índices en la lista de vértices. Los vértices y las caras son dos

[16] researchgate.net/publication/3730641_Application_of_discrete_Fourier_transform_to_electronic_measurements
[17] math.stackexchange.com/questions/579453/real-world-application-of-fourier-series
[18] docs.fileformat.com/3d/ply/

ejemplos de elementos y la mayoría del archivo PLY consta de estos dos elementos. También se pueden crear y adjuntar nuevas propiedades a los elementos de un objeto. Otras propiedades que també se puedo almacenar con el Objeto incluyen:

- Textura
- Color
- Transparencia
- Tango de confianza de datos
- Propiedades para el anverso y el reverso de un polígono

Un objeto representado por el formato PLY puede ser el resultado de varias fuentes, como objetos digitalizados a mano u objetos poligonales de aplicaciones de modelado. Se puede formato binario, así como con texto ASCII, nosotros usaremos archivos (.ply) en formato ASCII.

Un archivo PLY se estructura de la siguiente forma:

Encabezado de archivo: Esta al inicio del archivo y describe sus características, empieza con la palabra: “ply”, para reconocer el formato ply y termina con la palabra clave: “end_header”. Los comentarios en un archivo PLY empiezan con la palabra “comment”. En el encabezamiento están presente la palabra clave “element”. Le sigue el número de ese tipo de elemento, las propiedades para ese tipo de elemento específico donde cada propiedad tiene su tipo de propiedad y el orden especificado como se muestra a continuación:

$element\ vertex\ 8$	$\{define\ el\ elemento$ vertex, $y\ el\ numero\ que\ hay\}$
$property\ float\ x$	{"vertex" $tiene\ una\ propiedad\ "x"\ de\ tipo\ "float"\}$
$property\ float\ y$	{"vertex" $tiene\ una\ propiedad\ "y"\ de\ tipo\ "float"\}$
$property\ float\ z$	{"vertex" $tiene\ una\ propiedad\ "z"\ de\ tipo\ "float"\}$

Estas son las propiedades necesarias, pero como mencionado anterior mente también se puede añadir más propiedades. En un archivo PLY casi siempre hay 2 “element”, “element vertex” y “element face”.

4.1.2. Computación en 2D y 3D

Antes de tractar con objetos en 3D, averiguaremos como tratar con un dibujo en 2D. Ya que el trabajo está orientado en objetos 3D, usaremos como ejemplo contenido de fuente no propia. [19]

Recodamos que lo queremos obtener es una aproximación de nuestro dibujo con una transformada de Fourier. Para esto consideremos que nuestro dibujo es un recorrido cerrado de coordenadas (x, y) (Figura 4).

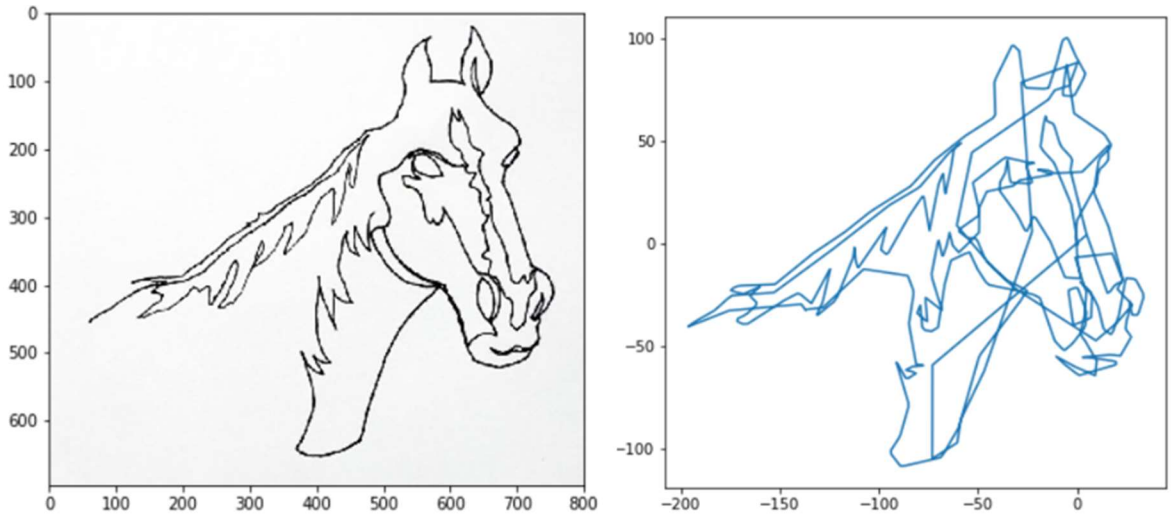


Figura 4: Dibujo expresado en coordenadas de puntos (x, y) y su recorrido cerrado respectivo

Seguidamente, podemos parametrizar el recorrido de cada una de las coordenadas de nuestro dibujo en función del tiempo (Figura 5). Así, obtenemos que si dibujo inicial es: $f(t)$, este compuesto de dos funciones: $f(x) = (x(t),\ y(t))$. Así obtenemos dos funciones de muestras que están separadas de manera uniforme, a las que podemos aplicar una DFT. De esta manera hemos obtenido la transformada de Fourier de nuestro dibujo inicial. Es decir, si realizamos una transformada Inversa, a las dos funciones transformadas, obtenemos nuestro dibujo inicial.

[19] stackoverflow.com/questions/57117182/drawing-rendering-3d-objects-with-epicycles-and-fourier-transformations-animati/57127276

Figura 5: Parametrización de las funciones f(x) y f(y) de la figura 5

Esto se puede mejorar, ya que si en vez de considerar $f(x)$ como la composición de dos funciones, consideramos que: $f: \mathbb{C}^N \to \mathbb{C}^N$, podemos obtener directamente la DFT del dibujo. De esta manera, las funciones $x(t), y(t)$ se juntan y $x(t)$ pasa a ser la parte real de $f(t)$ y $y(t)$ la imaginaria.

En conclusión, considerando que el dibujo se sitúa en el plano complejo podemos usar sus coordenadas como función para obtener la transformada que queremos.

Ahora, en vez de descomponer $f(t)$ en 2 señales $(x(t), y(t))$, $f(t)$ se descompone en 3 señales $f(t) = (x(t), y(t), z(t))$. Teniendo en cuenta que sabemos realizar una transformada cuando $f(t)$ se descompone en dos señales podemos separar estas 3 señales en dos funciones descritas por 2 señales: XY(t) y YZ(t). De esta manera obtenemos 2 DFT que describen nuestra función $f(t)$. (Figura 6)

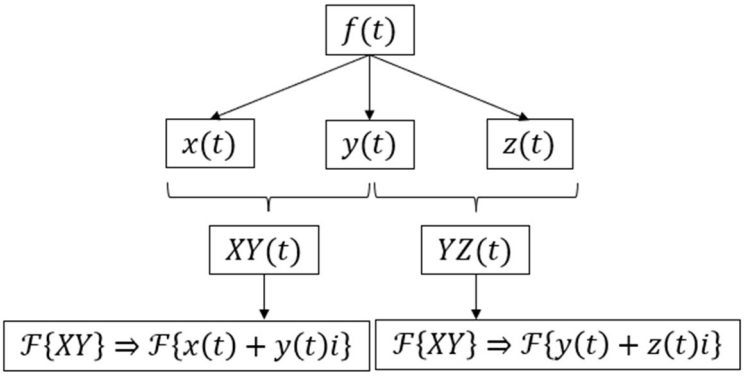


Figura 6: Esquema de la computación 3d de la serie de Fourier

4.1.4. Compresión de un archivo PLY

Para poder comprimir nuestros modelos 3D, primero hemos de averiguar cómo obtener la amplitud y fase de nuestro coeficiente de Fourier X_k obtenido con la DFT. Ya que el coeficiente de Fourier está en forma binómica, para obtener lo que buscamos, necesitamos pasar los coeficientes a forma polar.

$$X_n = a + bi \Rightarrow \alpha = \tan^{-1}\left(\frac{b}{a}\right) \; y \; |X_n| = \sqrt{a^2 + b^2}$$

Así mismo, como hemos dicho durante todo el trabajo, la transformada de Fourier transforma una función del dominio temporal al dominio de frecuencia. (Figura 7) Esto la hace descomponiendo la función inicial en diversas ondas senoidales con una frecuencia, fase y amplitud determinada.

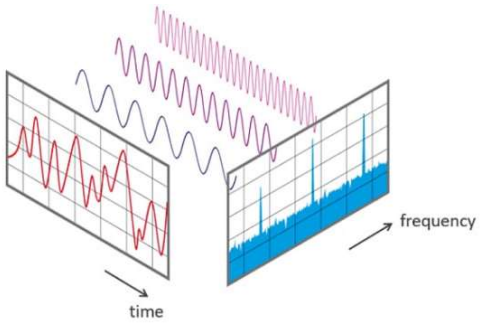


Figura 7: Representación gráfica del domino de frecuencia i el domino temporal

Así entonces después de aplicar la transformada de Fourier Discreta y pasar los coeficientes a forma polar, obtenemos la amplitud, frecuencia y fase. Como más grande es la amplitud de una onda, más afecta en la transformada inversa. Así mismo, si eliminamos las amplitudes más bajas (Figura 8), y realizamos la transformada inversa, obtenemos nuestra función inicial, pero con valores un poco distorsionados.

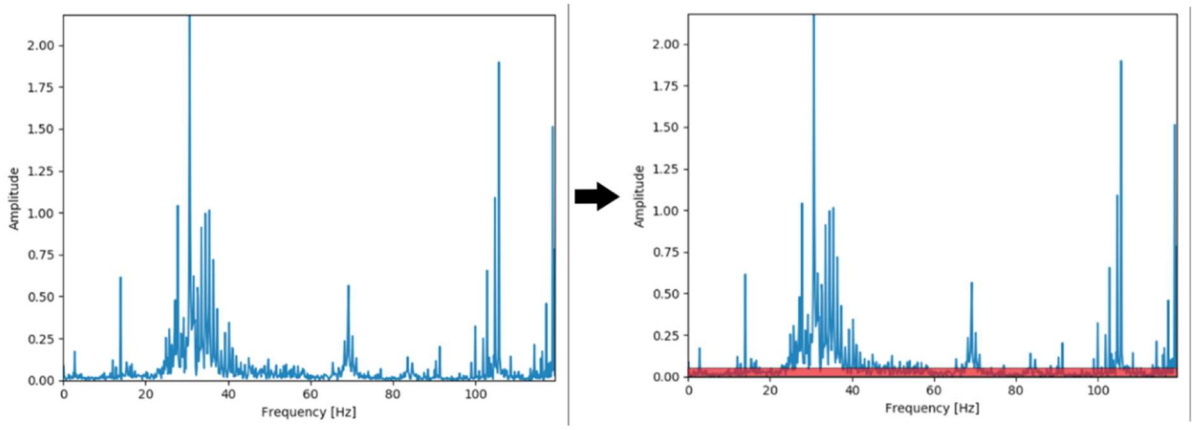


Figure 8: Representación del proceso de compresión eliminando la frecuencia de menor amplitud

Entonces, nuestro proceso de compresión consta de los pasos que se pueden observar en la Figura 9:

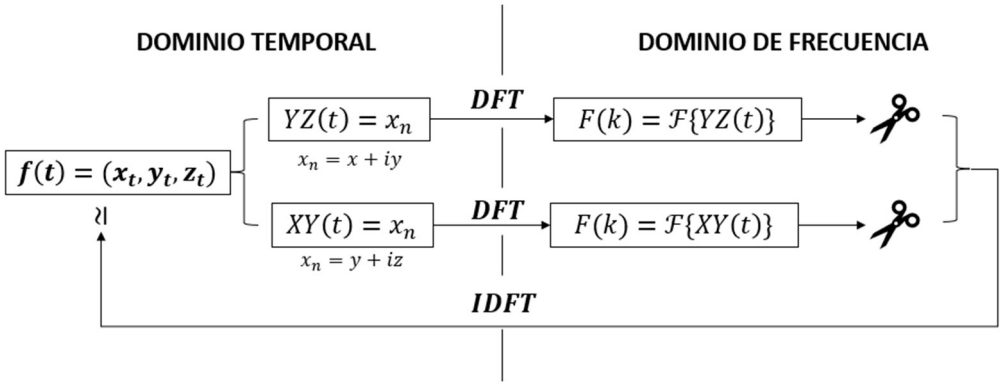


Figure 9: Esquema de todo el proceso de compresión

Para comparar los diferentes modelos que obtenemos usaremos la siguiente fórmula:

$$\Delta(xyz)_i = \sqrt{(x'_i - x_i)^2 + (y'_i - y_i)^2 + (z'_i - z_i)^2}$$

$$Error = \left(1 + \frac{1}{r^3}\sum_{i=0}^{N-1}(\Delta(xyz)_i)^3\right)^{-1};$$

Donde $\Delta(xyz)_i$ es la distancia entre los 2 puntos correspondientes de $f(t)$ y $f'(t)$ siendo $f'(t)$ la aproximación obtenida y r es la distancia entre el centro de masas i el punto de f(t) más lejos de este.

4.2. Resultados y análisis de la compresión

Los resultados se pueden apreciar en el anexo del trabajo (Anexo 1, 2, 3, 4).

Aunque se ha demostrado que se puede hacer esta compresión, no creo que sea una forma efectiva de realizar una compresión, ya que encuentro que la precisión que se pierde es mucho más valiosa que los megas que ganamos. Aun así, dependiendo del modelo que se está comprimiendo, se puede observar que cuando el modelo tiene menos puntos y sea más curvado, mejor será la compresión. Esto es debido a que gracias a que sea suave, no tiene amplitudes muy altas y como consecuencia al hacer la compresión, no afecta tanto.

5. CONCLUSIÓN

Con esto llegamos al final del Trabajo de Recerca y, al cabo de todos estos meses puedo afirmar que con mi trabajo he conseguido llegar a todos los objetivos que me propuse.

El camino no ha sido anda fácil, durante el trabajo me he encontrado con problemas constantes que he intentado superar al mejor de mis habilidades. Aun así, muchos de estos errores me han ayudado a crear un trabajo del cual estoy orgulloso y a terminar de comprender el concepto del análisis de Fourier.

No obstante, creo que aún hay lugar de mejor en mi trabajo. En un principio, si los puntos del archivo PLY estuviesen ordenados de manera óptima, creo que la transformada de Fourier hubiese sido más precisa al reducir los coeficientes, ya que la señal obtenida de este archivo hubiese sido más suabe. Para solucionar esto creo que se pudiese haber aplicado una variación en 3D de un algoritmo conocido como TSP (Traveling Salesman Problem), en el cual se calcula el camino óptimo que se puede realizar dados un conjunto de puntos en el espacio. Modificando este algoritmo para trabajar en puntos 3D y crear un gráfico con las arestas de los polígonos descritos por el archivo hubiese sido una buena solución para obtener una señal más suave. El problema con este algoritmo es que la forma más optimizada de este algoritmo tiene complexidad (número de operaciones que ha de realizar el ordenador) $O(n^2 \cdot 2^n)$ con una cantidad grande de puntos es imposible computar tal algoritmo. Otra propuesta de mejora que me hago es encontrar una mejor manera de cuantificar el error entre el

archivo comprimido y el original, ya que la fórmula que presentada hace una aproximación muy pagana del volumen y no nos da un valor realmente significativo.
No obstante, todo esto, estoy contento del trabajo y recerca que he hecho.

6. BIBLIOGRAFÍA Y WEBGRAFÍA

[1] “Funciones”. *Superproof,* 2019.
Web:
www.superprof.es/apuntes/escolar/matematicas/calculo/funciones/funciones.html
[Última consulta: 30/9/2021]
[2] “Paridad de funciones”. *Matesfacil,* any.
Web: www.matesfacil.com/BAC/funciones/paridad/funcion-par-impar-paridad-propiedades-demostraciones.html [Última consulta: 30/9/2021]
[3] “Periocidad de una función”. *La Guía,* 2013.
Web: https://matematica.laguia2000.com/general/periodicidad-de-una-funcion
[Última consulta: 30/9/2021]
[4] “Funciones Trigonométricas”. *Universo Formulas,* 2021.
Web: www.universoformulas.com/matematicas/analisis/funciones-trigonometricas/
[Última consulta: 30/9/2021]
[5] “Plano Complejo”. *Wikipedia,* 2021.
Web: https://es.wikipedia.org/wiki/Plano_complejo [Última consulta: 30/9/2021]
[6] “Serie de Fourier”. *Wikipedia,* 2021.
Web: https://es.wikipedia.org/wiki/Serie_de_Fourier [Última consulta: 30/9/2021]
[7] “Series y Transformadas de Fourier”. *Universitat Politécnica de Cartagena,* 2016.
Web:www.dmae.upct.es/~paredes/am_ti/apuntes/Tema%202.%20Series%20y%20transformadas%20de%20Fourier.pdf [Última consulta: 30/9/2021]
[8] “Applications de la Serie de Fourier”. *El País,* any. Web:
www.elpais.com/ciencia/2020-10-01/la-serie-que-cambio-el-mundo.html [Última consulta: 30/9/2021]
[9] “Historia de la Serie de Fourier”. *Wikipedia,* 2021.
Web: https://en.wikipedia.org/wiki/Fourier_series#History [Última consulta: 30/9/2021]
[10] “Serie de Fourier”. *Wikipedia,* 2021.
Web: https://es.wikipedia.org/wiki/Serie_de_Fourier [Última consulta: 30/9/2021]
[11] “Rendering 2D and 3D bodies with Fourier Transforms”. *Shultz, C.* 2016.

Trabajo: https://app.box.com/s/thpam69f04u5ygntpnemzxtme8hcig1b [Última consulta: 30/9/2021]

[12] “Transformada de Fourier”. *Wikipedia,* 2021.
Web: https://en.wikipedia.org/wiki/Discrete_Fourier_transform [Última consulta: 30/9/2021]

[13] “Applications of the DFT”. *Science Direct,* 2021.
Web: www.sciencedirect.com/topics/engineering/discrete-fourier-transform [Última consulta: 30/9/2021]

[14] “Application of DFt to electronic mesurements”. *ResearchGate,* 1997.
Web:https://researchgate.net/publication/3730641_Application_of_discrete_Fourier_transform_to_electronic_measurements [Última consulta: 30/9/2021]

[15] “Real world aplications of Foureir seires”. *StackExchange,* 2013.
Web: https://math.stackexchange.com/questions/579453/real-world-application-of-fourier-series [Última consulta: 30/9/2021]

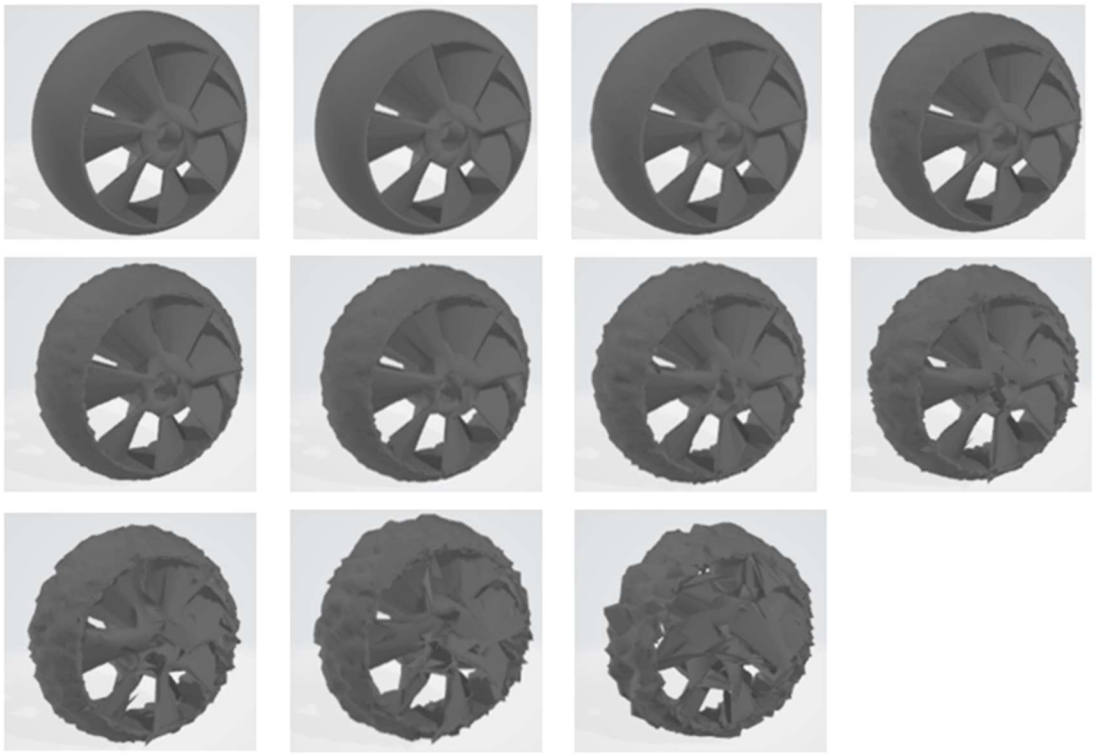
[16] “What is a PLY file?”. *Fileformat,* 2012.
Web: https://docs.fileformat.com/3d/ply/ [Última consulta: 30/9/2021]

[17] “Rendering 3D objects with epicycles”. *StackOverflow,* 2019.
Web: https://stackoverflow.com/questions/57117182/drawing-rendering-3d-objects-with-epicycles-and-fourier-transformations-animati/57127276 [Última consulta: 30/9/2021]

7. ANEXO

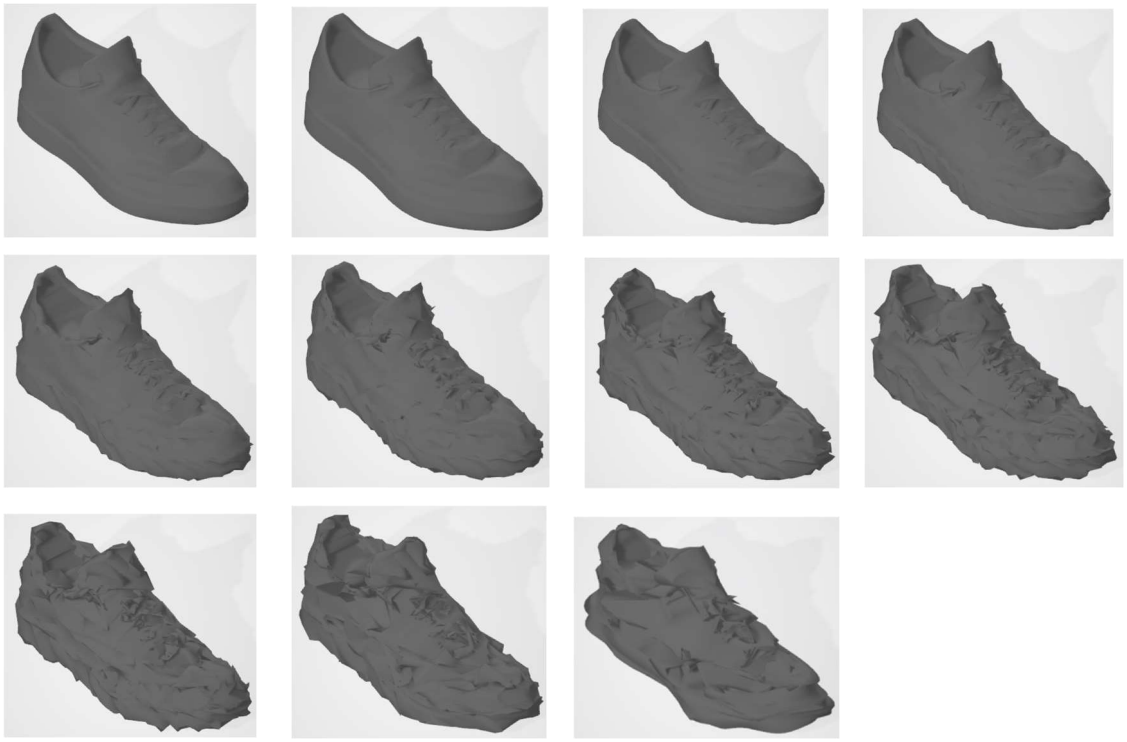
7.1. Compresión: Turbine.ply

Archivo original	# de vértices:	Tamaño	# de caras	Modelo
Turbine.ply	3309	220 KB	6618	0
% de X_n usados	# de X_n usados	Compresión	Error	Modelo
100	3309	264 KB	100.00%	1
90	2978	248 KB	99.96%	2
80	2647	231 KB	99.69%	3
70	2316	215 KB	98.66%	4
60	1985	198 KB	96.06%	5
50	1654	182 KB	89.83%	6
40	1323	166 KB	77.77%	7
30	992	150 KB	58.92%	8
20	661	135 KB	37.94%	9
10	330	119 KB	17.50%	10
0	0	104 KB	0.00%	….



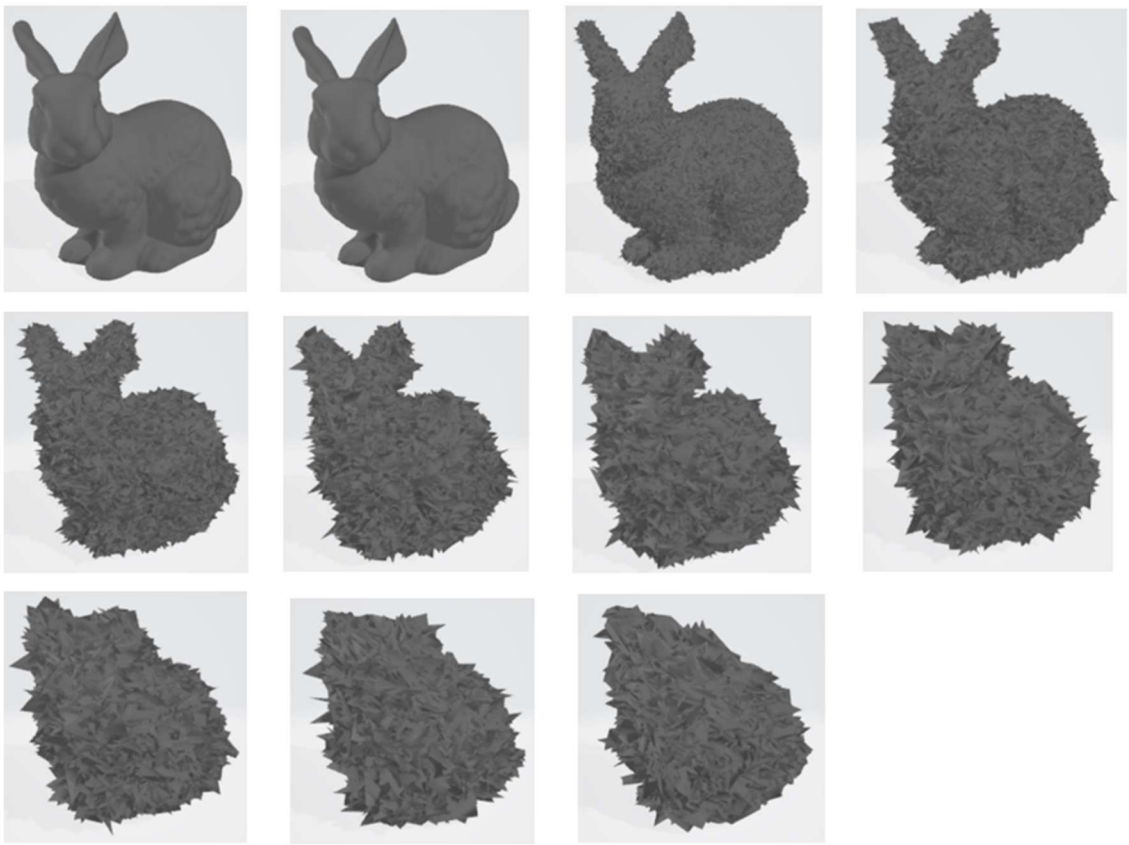
7.2. Compresión: Shoe.ply

Archivo original	# de vértices:	Tamaño	# de caras	Modelo
Shoe.ply	1841	121 KB	3634	0
% de X_n usados	**# de X_n usados**	**Compresión**	**Error**	**Modelo**
100	1841	140 KB	100%	1
90	1656	131 KB	99.71%	2
80	1472	122 KB	97.46%	3
70	1288	114 KB	95.34%	4
60	1104	105 KB	80.54%	5
50	920	96 KB	75.07%	6
40	736	88 KB	52.61%	7
30	552	79 KB	38.22%	8
20	368	71 KB	23.16%	9
10	184	63 KB	13.35%	10
0	0	55 KB	0.00%	11



7.3. Compresión: Bunny.ply

Archivo original	# de vértices:	Tamaño	# de caras	Modelo
Bunny.ply	35947	2576 KB	69451	0
% de X_n usados	# de X_n usados	Compresión	Error	Modelo
100	35947	3142 KB	100%	1
90	32352	2956 KB	97.46%	2
80	28757	2771 KB	90.16%	3
70	25162	2586 KB	75.06%	4
60	21568	2400 KB	57.61%	5
50	17973	2215 KB	42.12%	6
40	14378	2030 KB	30.01%	7
30	10784	1845 KB	15.12%	8
20	7189	1660 KB	7.38%	9
10	3594	1476 KB	3.87%	10
0	0	1295 KB	0%	11



7.4. Compresión: Teapot.ply

Archivo original	# de vértices:	Tamaño	# de caras	Modelo
Teapot.ply	1177	77 KB	2256	0
% de X_n usados	**# de X_n usados**	**Compresión**	**Error**	**Modelo**
100	3309	86 KB	100%	1
90	2978	80 KB	99.97%	2
80	2647	75 KB	99.73%	3
70	2316	70 KB	98.83%	4
60	1985	64 KB	96.24%	5
50	1654	59 KB	90.53%	6
40	1323	53 KB	80.78%	7
30	992	48 KB	66.50%	8
20	661	42 KB	46.91%	9
10	330	37 KB	25.34%	10
0	0	32 KB	0.00%	11

7.5. Programa 1: DFT

```
#include <iostream>
#include <fstream>
#include <string>
#include <bits/stdc++.h>
using namespace std;

#define real first
#define img second
#define amp first.first
#define phase first.second

int elementVertex;
int elementFace;


pair<double, double> Complex(double re, double im){
    pair<double, double> ComplexNumber;
    ComplexNumber = {re, im};
    return ComplexNumber;
}

pair<double, double> ComplexAdd(pair<double, double>num1, pair<double,
double>num2){
    pair<double, double> result;
    result.real = num1.real + num2.real;
    result.img = num1.img + num2.img;
    return result;
}

pair<double, double> ComplexProduct(pair<double, double>num1, pair<double,
double>num2){
    pair<double, double> result;
    result.real = num1.real * num2.real - num1.img * num2.img;
    result.img = num1.real * num2.img + num1.img * num2.real;
    return result;
}


vector<pair<pair<double, double>, int>>dft(vector<pair<double, double>>f){
    vector<pair<pair<double, double>, int>>X(elementVertex);
    pair<double, double>sum;
```

```
42
43    for(int k = 0; k < elementVertex; ++k){
44       sum = Complex(0.0, 0.0);
45       for(int n = 0; n < elementVertex; ++n){
46          double w = (M_PI * 2 * k * n)/elementVertex;
47          pair<double, double> phi = Complex(cos(w), -sin(w));
48          sum = ComplexAdd(sum, ComplexProduct(f[n], phi));
49       }
50
51       sum.real /= elementVertex;
52       sum.img /= elementVertex;
53
54
55       X[k].second = k;
56       X[k].amp = sqrt(sum.real * sum.real + sum.img * sum.img);;
57       X[k].phase = atan2(sum.img, sum.real);;
58    }
59
60    return X;
61 }
62
63 int main() {
64    string name;
65    cin >> name;
66    ifstream file(name);
67    string word;
68
69
70    while(file >> word) {
71       if(word == "vertex"){
72          file >> elementVertex;
73          cout << elementVertex << endl;
74       }
75       if(word == "face"){
76          file >> elementFace;
77          cout << elementFace << endl;
78       }
79       if(word == "end_header") break;
80    }
81
82
83    vector<double>x(elementVertex);
84    vector<double>y(elementVertex);
```

```
85    vector<double>z(elementVertex);
86
87    for(int i = 0; i < elementVertex; ++i){
88       file >> x[i] >> y[i] >> z[i];
89    }
90
91    vector<vector<int>>face(elementFace);
92    int size, inp;
93    for(int i = 0; i < elementFace; ++i){
94       file >> size;
95       face[i].push_back(size);
96       for(int r = 0; r < size; ++r){
97          file >> inp;
98          face[i].push_back(inp);
99       }
100   }
101
102
103   vector<pair<double, double>>complexSignalXY(elementVertex);
104
105   for(int i = 0; i < elementVertex; ++i){
106      complexSignalXY[i].real = x[i];
107      complexSignalXY[i].img = y[i];
108   }
109
110   vector<pair<pair<double, double>, int>>dftXY;
111   dftXY = dft(complexSignalXY);
112
113
114   vector<pair<double, double>>complexSignalYZ(elementVertex);
115   for(int i = 0; i < elementVertex; ++i){
116      complexSignalYZ[i].real = y[i];
117      complexSignalYZ[i].img = z[i];
118   }
119
120   vector<pair<pair<double, double>, int>>dftYZ;
121   dftYZ = dft(complexSignalYZ);
122
123
124 for(int num = 1; num <= 10; ++num){
125   int elementCoeficient = elementVertex*num/10;
126   string numS = to_string(num);
127   ofstream newFile("dft_"+numS+'_'+name);
```

```
128     newFile <<
129     "ply" << endl <<
130     "format ascii 1.0" << endl <<
131     "comment made by Pol de los Santos" << endl <<
132     "comment VCGLIB generated" << endl <<
133     "element vertex " << elementVertex  << endl <<
134     "element coeficient " << elementCoeficient << endl <<
135     "property float frequency" << endl <<
136     "property float aplitud" << endl <<
137     "property float phase" << endl <<
138     "element face " << elementFace << endl <<
139     "property list uchar int vertex_index" << endl <<
140     "end_header" << endl;
141
142
143     sort(begin(dftXY), end(dftXY));
144     reverse(begin(dftXY), end(dftXY));
145     sort(begin(dftYZ), end(dftYZ));
146     reverse(begin(dftYZ), end(dftYZ));
147
148
149     for(int i = 0; i < elementCoeficient ; ++i){
150         newFile << dftXY[i].second << ' ' << dftXY[i].amp << ' ' << dftXY[i].phase << endl;
151     }
152     for(int i = 0; i < elementCoeficient ; ++i){
153         newFile << dftYZ[i].second << ' ' << dftYZ[i].amp << ' ' << dftYZ[i].phase << endl;
154     }
155
156     for(int i = 0; i < elementFace; ++i){
157         newFile << face[i][0];
158         for(int t = 1; t < face[i].size(); ++t){
159             newFile << ' ' << face[i][t];
160         }
161         newFile  << endl;
162     }
163
164     newFile.close();
165 }
166     return 0;
167
168 }
```

7.6. Programa 2: IDFT

```
1  #include <iostream>
2  #include <fstream>
3  #include <string>
4  #include <bits/stdc++.h>
5  using namespace std;
6
7  #define real first
8  #define img second
9  #define real first
10 #define img second
11 #define amp first.first
12 #define phase first.second
13 int elementVertex;
14 int elementFace;
15 int elementCoeficients;
16
17
18 vector<pair<double, double>> idft(vector<pair<pair<double, double>, int>>plano_2D){
19    vector<pair<double, double>>f;
20    pair<double, double>sum;
21    double time = 0.0;
22
23    for(int n = 0; n < elementVertex; ++n){
24       sum = {0, 0};
25       for(int k = 0; k < elementCoeficients; ++k){
26          double freq = plano_2D[k].second;
27          double amplitud = plano_2D[k].amp;
28          double pha = plano_2D[k].phase;
29
30          double x = amplitud * cos(freq*time+pha);
31          double y = amplitud * sin(freq*time+pha);
32          sum.real+=x;
33          sum.img+=y;
34
35       }
36       double dt = (2*M_PI)/elementVertex ;
37       time += dt;
38       f.push_back(sum);
39
40    }
41
```

```
42    return f;
43  }
44
45  int main() {
46    string name;
47    cin >> name;
48    ifstream file(name);
49    string word;
50
51
52    while(file >> word) {
53      if(word == "vertex"){
54        file >> elementVertex;
55        cout << elementVertex << 'y' << endl;
56      }
57      if(word == "coeficient"){
58        file >> elementCoeficients;
59        cout << elementCoeficients << endl;
60      }
61      if(word == "face"){
62        file >> elementFace;
63        cout << elementFace << endl;
64      }
65      if(word == "end_header") break;
66    }
67
68
69    vector<pair<pair<double, double>, int>>XY(elementCoeficients);
70    vector<pair<pair<double, double>, int>>YZ(elementCoeficients);
71
72    for(int i = 0; i < elementCoeficients; ++i){
73      file >> XY[i].second >> XY[i].amp >> XY[i].phase;
74    }
75    for(int i = 0; i < elementCoeficients; ++i){
76      file >> YZ[i].second >> YZ[i].amp >> YZ[i].phase;
77    }
78
79    vector<vector<int>>face(elementFace);
80    int size, inp;
81    for(int i = 0; i < elementFace; ++i){
82      file >> size;
83      face[i].push_back(size);
84      for(int r = 0; r < size; ++r){
```

```
85          file >> inp;
86          face[i].push_back(inp);
87       }
88    }
89
90
91    vector<double>x(elementVertex);
92    vector<double>y1(elementVertex);
93    vector<double>y2(elementVertex);
94    vector<double>z(elementVertex);
95
96    vector<pair<double, double>>XY_pair(elementVertex);
97    XY_pair = idft(XY);
98    for(int i = 0; i < elementVertex; ++i){
99       x[i] = XY_pair[i].real;
100      y1[i] = XY_pair[i].img;
101   }
102
103
104
105   vector<pair<double, double>>YZ_pair(elementVertex);
106   YZ_pair = idft(YZ);
107   for(int i = 0; i < elementVertex; ++i){
108      y2[i] = YZ_pair[i].real;
109      z[i] = YZ_pair[i].img;
110   }
111
112
113   string num;
114   num.push_back(name[4]);
115
116   ofstream newFile("idft"+num+".txt");
117   newFile <<
118   "ply" << endl <<
119   "format ascii 1.0" << endl <<
120   "comment made by Pol de los Santos" << endl <<
121   "comment VCGLIB generated" << endl <<
122   "element vertex " << elementVertex  << endl <<
123   "property float x" << endl <<
124   "property float y" << endl <<
125   "property float z" << endl <<
126   "element face " << elementFace << endl <<
127   "property list uchar int vertex_index" << endl <<
```

```
128   "end_header" << endl;
129
130
131
132   for(int i = 0; i < elementVertex ; ++i){
133      newFile << x[i] << ' ' << (y1[i] + y2[i])/2 << ' ' << z[i] << endl;
134   }
135
136   for(int i = 0; i < elementFace; ++i){
137      newFile << face[i][0];
138      for(int t = 1; t < face[i].size(); ++t){
139         newFile << ' ' << face[i][t];
140      }
141      newFile  << endl;
142   }
143
144   newFile.close();
145   return 0;
146 }
147
148
```