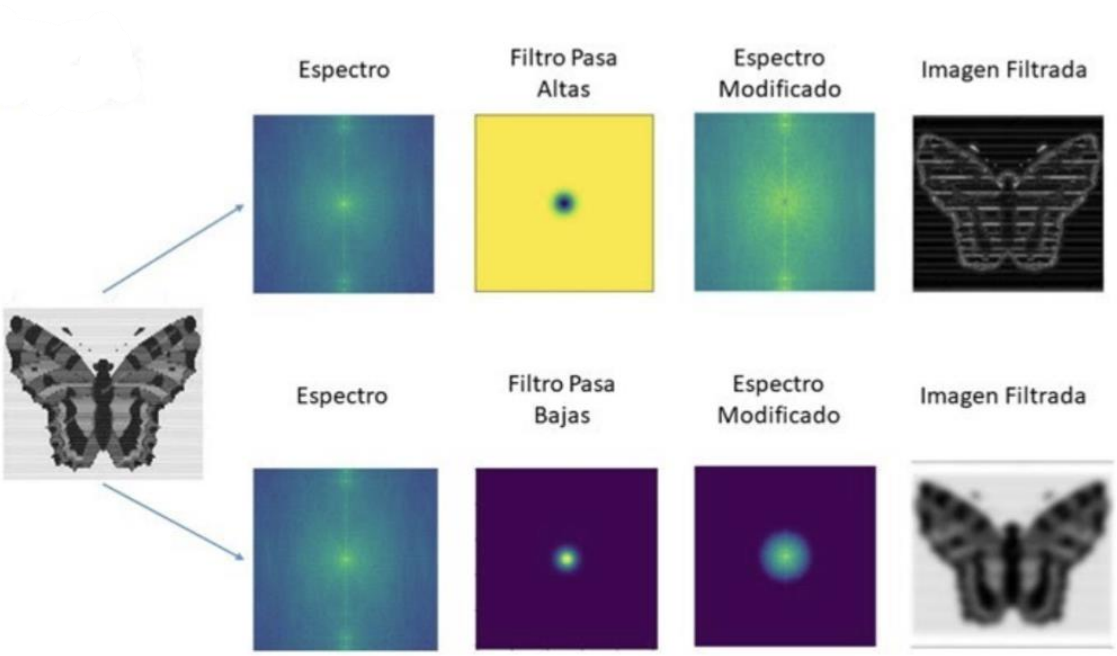
INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE COMPUTO

Matemáticas Avanzadas.

Proyecto: Filtros en imágenes con Transformada de Fourier 2D (pasa-bajas vs pasa-altas)

Integrantes:
Castillo Rivera Diego
Reyes Santamaria Tania Jannet

Profesor: Dr. David Correa Coyac



Introducción

El filtrado de imágenes es una técnica esencial en la ingeniería de sistemas y las ciencias de la computación. Si bien existen numerosas librerías que automatizan estos procesos, comprender la base matemática que opera "por debajo de la mesa" es crucial para cualquier ingeniero o estudiante de ingenieria. Este proyecto se centra en la implementación y visualización de filtros en el dominio de la frecuencia utilizando la Transformada Rápida de Fourier (FFT).

El propósito de este trabajo para entrega final es presentar una aplicación desarrollada en Python que descompone una imagen en sus componentes frecuenciales senoidales. A través de una interfaz gráfica diseñada para la experimentación, el sistema permite aplicar máscaras de filtrado selectivo sobre el espectro de magnitud.

A diferencia de implementaciones estáticas, este proyecto integra controles interactivos (sliders) que permiten al usuario ajustar el radio de corte de los filtros en tiempo real. Esto facilita una comprensión intuitiva de la dualidad entre el dominio espacial y el frecuencial: observando simultáneamente cómo un filtro Pasa-Bajas suaviza la imagen eliminando detalles finos, mientras un filtro Pasa-Altas resalta los bordes y cambios bruscos. Finalmente, se utiliza el Error Cuadrático Medio (MSE) como métrica cuantitativa para evaluar la pérdida de información inherente al proceso de filtrado.

La principal motivación de este proyecto radica en el interés por aplicar los conceptos teóricos de Matemáticas Avanzadas utilizando herramientas computacionales modernas. Elegimos Python por su potencia en el manejo de cálculo numérico, lo cual nos permite abstraer la complejidad del código y centrarnos en el análisis matemático de la Transformada de Fourier 2D.

Adicionalmente, buscamos explorar las ventajas del procesamiento digital frente a métodos tradicionales. Durante la fase de planeación, evaluamos la posibilidad de trabajar con compresión de imágenes mediante truncamiento; sin embargo, optamos por la implementación de filtros de frecuencia (Pasa-Bajas y Pasa-Altas). Esta decisión se basó en el deseo de observar y comparar de manera gráfica cómo la atenuación de componentes espectrales afecta la nitidez y los detalles de una imagen.

Marco Matematico

1. Transformada Discreta de Fourier en 2D (DFT)

Este es el corazón del proyecto. Como trabajamos con imágenes digitales (píxeles discretos, no señales continuas), no usamos integrales, sino sumatorias.

Definición: La imagen se representa como una función discreta f(x,y) de tamaño M×N. La DFT convierte esta información espacial en el dominio de la frecuencia F(u,v).

Formula:

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

En donde f(x,y) es la intensidad del píxel en la coordenada espacial (x,y), F(u,v) es el valor complejo en la frecuencia (u,v) y $e^{-j\cdots}$es la base de la transformación (Fórmula de Euler), compuesta por senos y cosenos. Esto es lo que hace la linea de codigo `f = np.fft.fft2(imagen)`.

2. Teorema de la Convolución

Este es el teorema más importante para justificar por qué multiplicamos máscaras.

El Teorema Establece que la convolución en el dominio del espacio equivale a la multiplicación en el dominio de la frecuencia.

$$f(x,y) * h(x,y) \Leftrightarrow F(u,v) \cdot H(u,v)$$

Y para su aplicación el filtrado de una imagen espacialmente implica hacer una "convolución" (una operación que es costosa computacionalmente). Y gracias a este teorema, se puede simplemente multiplicar la imagen transformada F(u,v) por una máscara H(u,v) (tu círculo blanco y negro) y obtener el mismo resultado mucho más rápido.
Y en el codigo la línea `fshift_filtrado = fshift * mascara`. aplicamos el teorema de la convolución para filtrar.

3. Propiedad de Traslación (Shift) y Centrado

Bajo su concepto, la DFT es periódica. Por defecto, el componente de frecuencia cero (DC) aparece en la esquina (0,0). Y como propiedad Multiplica la función espacial por $(-1)^{x+y}$ traslada el centro del espectro de frecuencia a (M/2,N/2).

$$f(x,y)\ (-1)^{x+y} \Leftrightarrow F(u-M/2, v-N/2)$$

Esto permitiendo visualizar las frecuencias bajas en el centro geométrico de la imagen y las altas en los bordes, facilitando la interpretación visual y la creación de filtros simétricos (círculos). (Realizado en el codigo por la función `np.fft.fftshift(f)`.)

4. Definición de Filtros Ideales (La Máscara)

Aquí se debe definir matemáticamente qué es ese "círculo" que se programa. Se modela como una función escalón (Heaviside) radial.
Distancia Euclidiana (D): La distancia desde cualquier punto (u,v) al centro del espectro (u_0, v_0):

$$D(u,v) = \sqrt{(u-u_0)^2 + (v-v_0)^2}$$

Filtro Pasa-Bajas Ideal (H_{LP}):

$$H_{LP}(u,v) = \begin{cases} 1 & \text{si } D(u,v) \leq D_0 \\ 0 & \text{si } D(u,v) > D_0 \end{cases}$$

(Deja pasar solo lo que está dentro del radio de corte D_0)

Filtro Pasa-Altas Ideal (H_{HP}):

$$H_{HP}(u,v) = \begin{cases} 0 & \text{si } D(u,v) \leq D_0 \\ 1 & \text{si } D(u,v) > D_0 \end{cases}$$

O solo 1- $H_{LP}(u,v)$.
Y en el codigo corresponde a la función `crear_mascara` y a lo que controlamos con el Slider (D_0 es el `radio`).

5. Transformada Inversa (IDFT)

Y para terminar para ver la imagen de vuelta, se necesita la operación inversa.

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

(ahora el exponente ya es positivo y se divide por el total de pixeles)
(la función en el codigo es `reconstruir_imagen` usando `np.fft.ifft2`.)

Por lo que el proyecto se fundamenta en el Teorema de la Convolución, utilizando la Transformada Discreta de Fourier (DFT)para llevar la imagen al dominio de la frecuencia. Allí, es donde aplicamos funciones de transferencia H(u,v) (Filtros Ideales Pasa-Bajas y Pasa-Altas) mediante una multiplicación punto a punto. Finalmente, mediante la Transformada Inversa, recuperamos la imagen filtrada g(x,y)."

Descripción de la implementación

Librerias y dependencias empleadas en el proyecto:
Para tener una buena eficiencia numérica y la interactividad, integramos las siguientes librerías especializadas:

NumPy (`numpy`): Constituye el núcleo matemático del proyecto. Se utilizó para el manejo de matrices multidimensionales (la imagen usada como matriz) y, fundamentalmente, para la ejecución de los algoritmos FFT (Fast Fourier Transform) y su inversa.

OpenCV (`cv2`): Utilizada exclusivamente para la gestión de Entrada/Salida (I/O) de imágenes y el preprocesamiento básico (conversión del espacio de co lor BGR a Escala de Grises).

Matplotlib (`matplotlib.pyplot` y `widgets`): Empleada para la generación de la interfaz gráfica de usuario (GUI). Se usó no solo para graficar los espectros y resultados, sino también para implementar el control interactivo (Slider) que actualiza los filtros en tiempo real.

Tkinter (`tkinter`): Librería estándar de Python utilizada para invocar el explorador de archivos nativo del sistema operativo, permitiendo una selección de imágenes intuitiva y robusta.

Arquitectura de Modulos

A continuación explicare la estructura en funciones específicas que encapsulan cada etapa matemática del proceso:

Módulo de Adquisición y Preprocesamiento
(`seleccionar_imagen` y `cargar_y_preprocesar`)

Se implementó una ventana de diálogo oculta (`root.withdraw`) para permitir al usuario seleccionar archivos de imagen locales y la imagen seleccionada se convierte a escala de grises (eliminando los canales de color RGB para trabajar con una señal 2D pura) y se **normaliza** dividiendo por 255.0.

Transformación al Dominio de la Frecuencia (`aplicar_fft`)}

Ya explicado aneriormente, aquí utiliza `np.fft.fft2` para obtener la Transformada Discreta de Fourier. Procede a aplicar `np.fft.fftshift`, una operación crítica que reorganiza los cuadrantes de la matriz resultante, moviendo el componente de frecuencia cero (DC) de la esquina (0,0) al centro geométrico de la imagen (N/2,M/2). Esto facilita la aplicación de filtros simétricos.

Generación de Filtros (`crear_mascara`)
Este módulo crea dinámicamente matrices binarias (máscaras) basadas en la distancia Euclidiana respecto al centro de la imagen.
Implementa la lógica de Filtros Ideales:
-El Pasa-Bajas asigna valor 1 a las frecuencias dentro del radio de corte D_0 y 0 fuera de él.
- El Pasa altas (Logica inversa del Pasa-Bajas) 0 Dentro, 1 fuera.

Reconocimiento Espacial (`reconstruir_imagen`)
Aquí se realiza el producto punto a punto entre el espectro desplazado y la máscara generada y aplica la transformación inversa (`ifftshift` seguida de `ifft2`) para retornar al dominio espacial. Se extrae la magnitud (`np.abs`) del resultado complejo para visualizar la imagen final.

Visualización Interactiva y Métricas

Se diseñó un layout visual tipo "GridSpec" que organiza:

1. Imagen original y su espectro.
2. Rama superior: Máscara y resultado del filtro Pasa-Altas.
3. Rama inferior: Máscara y resultado del filtro Pasa-Bajas.

Cálculo de MSE: Se implementó una función de métrica de error en tiempo real que compara la imagen original con la filtrada, proporcionando un valor numérico de la "pérdida de información".

Interactividad: Se programó un *callback* vinculado al objeto `Slider`. Al detectar un evento de movimiento ("on_changed"), el sistema recalcula las máscaras y las transformadas inversas, redibujando el lienzo gráfico instantáneamente.

Datos

Para las fuentes, como el proyecto tiene un selector de archivos son "dinamicas" por lo que se emplearon imágenes clásicas de procesamiento de señales obtenidas de repositorios académicos abiertos (e.g., USC-SIPI Image Database) o internet para verificar la correcta detección de bordes y suavizado.

Licencias

El proyecto se construye sobre un ecosistema de librerías de código abierto (Open Source) que permiten su uso libre:

-Python: Python Software Foundation License (PSFL).

-OpenCV (cv2): Licencia Apache 2.0 (permite uso comercial y académico).

-NumPy y Matplotlib: Licencia BSD / NumFOCUS (altamente permisiva para cálculo científico).

Preprocesamiento

En nuestra función `cargar_y_preprocesar` esta hace dos cosas críticas antes de que la imagen pueda tocar la Transformada de Fourier y sin estas estapas de preprocesamiento no funcionaria del todo bien.
Antes de aplicar la Transformada de Fourier 2D, las imágenes de entrada se someten a una etapa de acondicionamiento de señal obligatoria que consta de dos pasos.

1-Conversión Espectral (Escala de Grises):

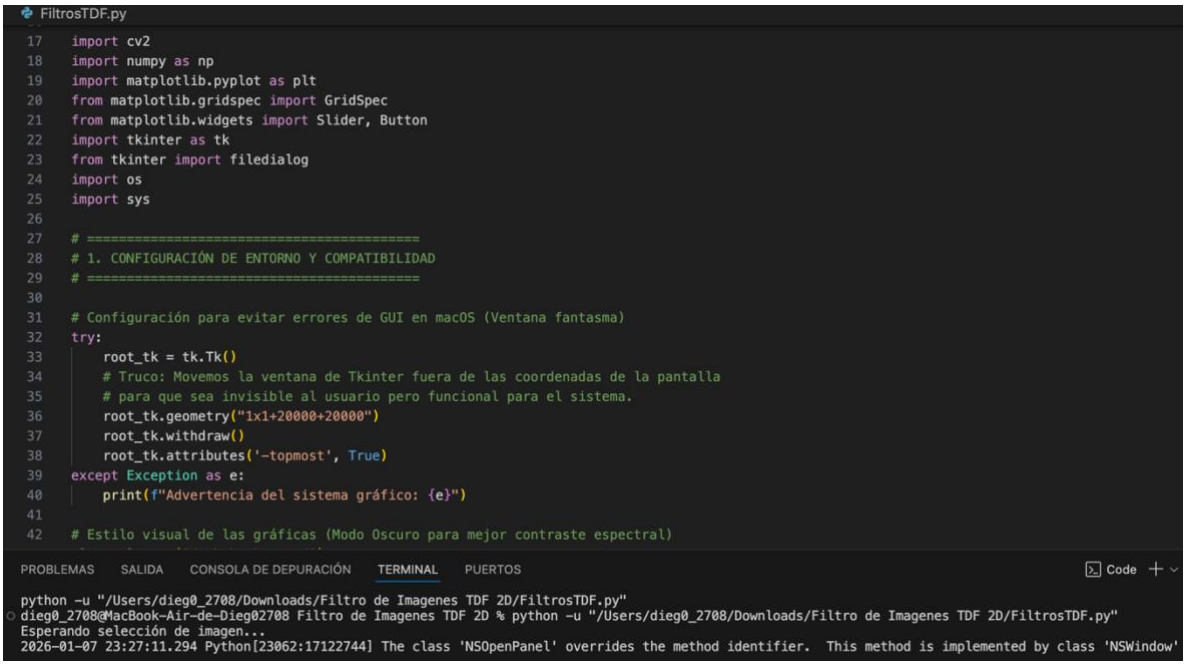
- Aquí las imágenes cargadas originalmente en formato BGR (Blue-Green-Red) poseen tres canales de color. Se utiliza la función cv2.cvtColor para colapsar estos canales en uno solo basado en la luminancia. Y la Transformada de Fourier 2D implementada requiere una señal bidimensional f(x,y) de intensidad única. Trabajar

con tres canales requeriría procesar tres transformadas independientes, lo cual incrementa la carga computacional innecesariamente para los objetivos de este proyecto (análisis de frecuencias estructurales).

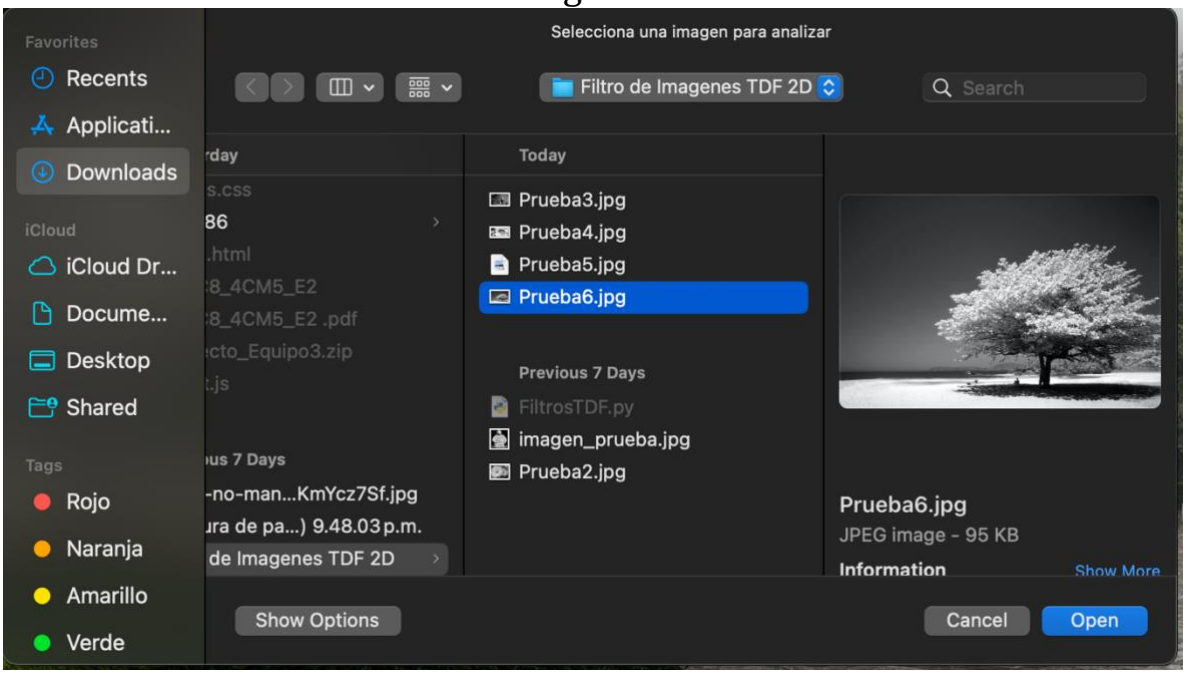
2- Normalización de Intensidad:
Los valores de los píxeles, originalmente enteros en el rango $[0,255]$ (formato `uint8`), son convertidos a punto flotante y divididos por 255.0. y esto reasigna los valores al rango $[0,1]$. La normalización es crucial para la estabilidad numérica de la FFT y es un requisito para que la librería `matplotlib` interprete y visualice correctamente las intensidades sin generar artefactos de saturación al realizar operaciones matemáticas complejas.

Resultados

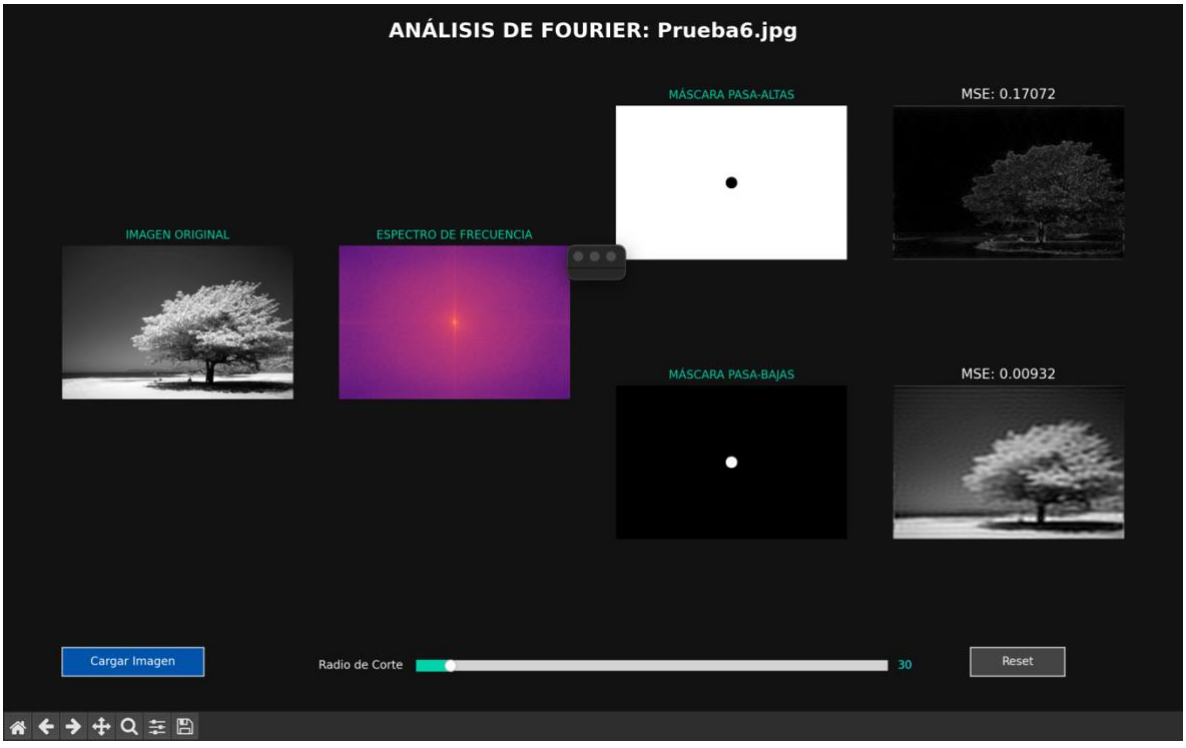
Compilacion y mensajes en la terminal.



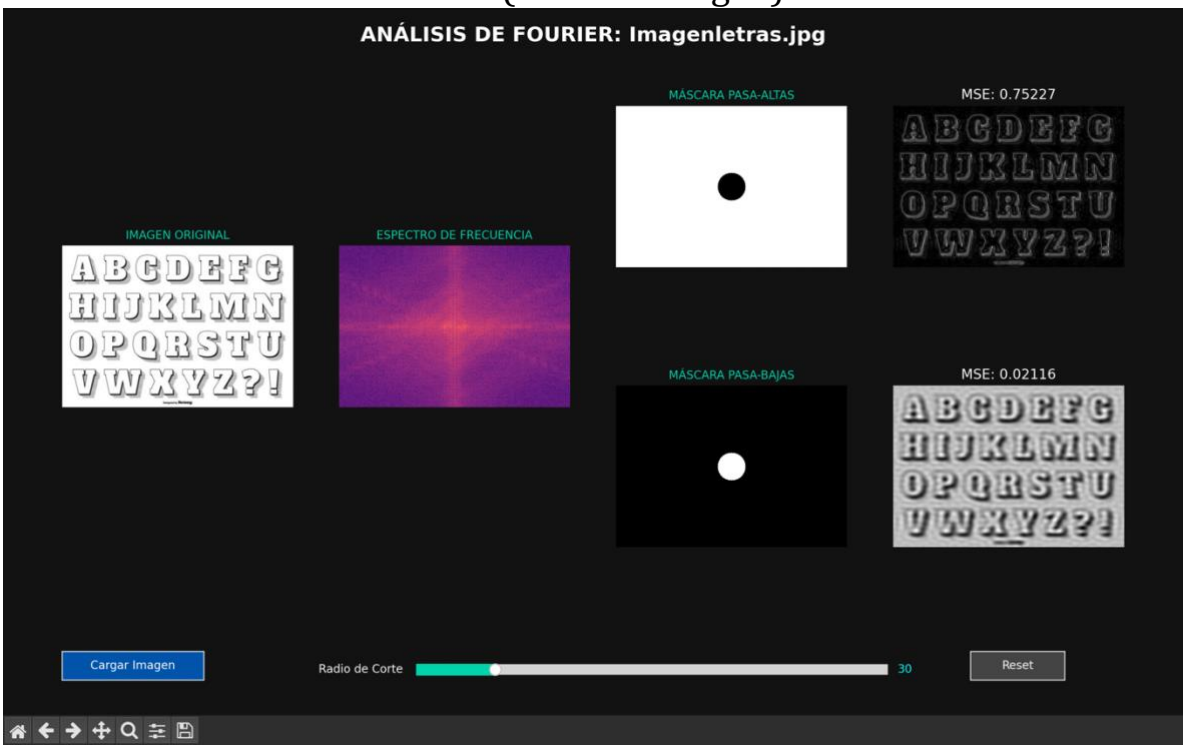
Selector de archivos con File dialog



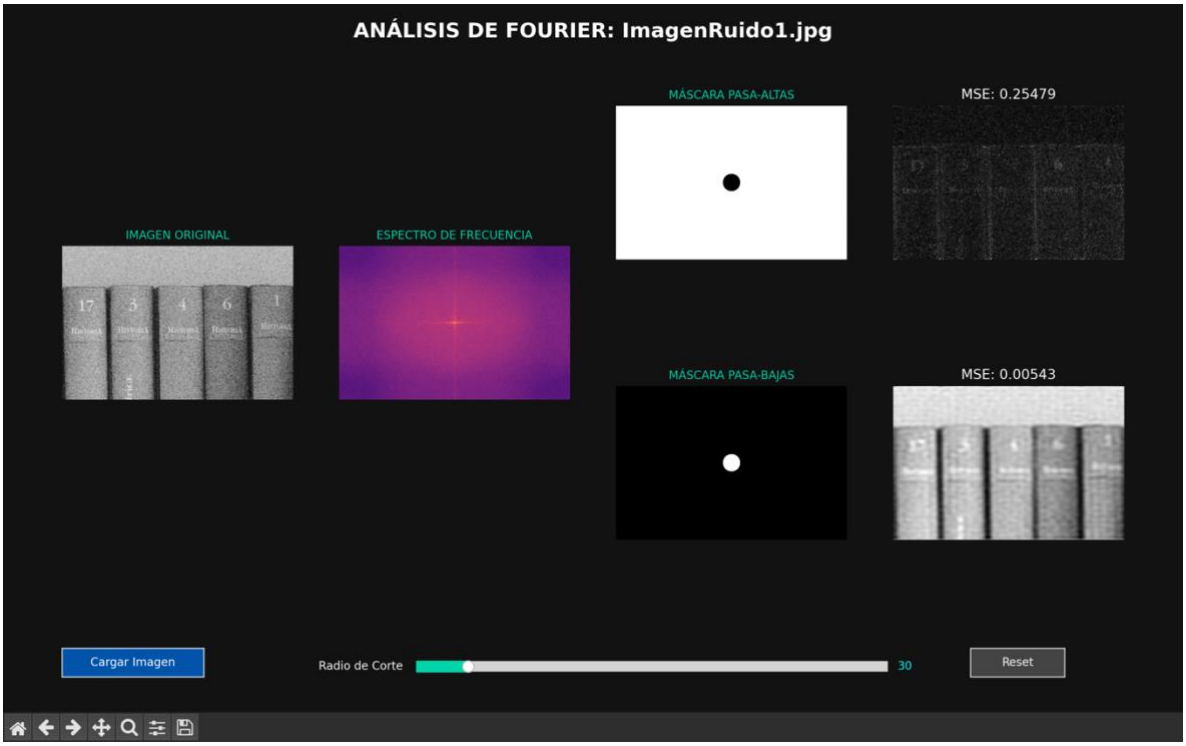
Interfaz resultante con radio de corte de 30



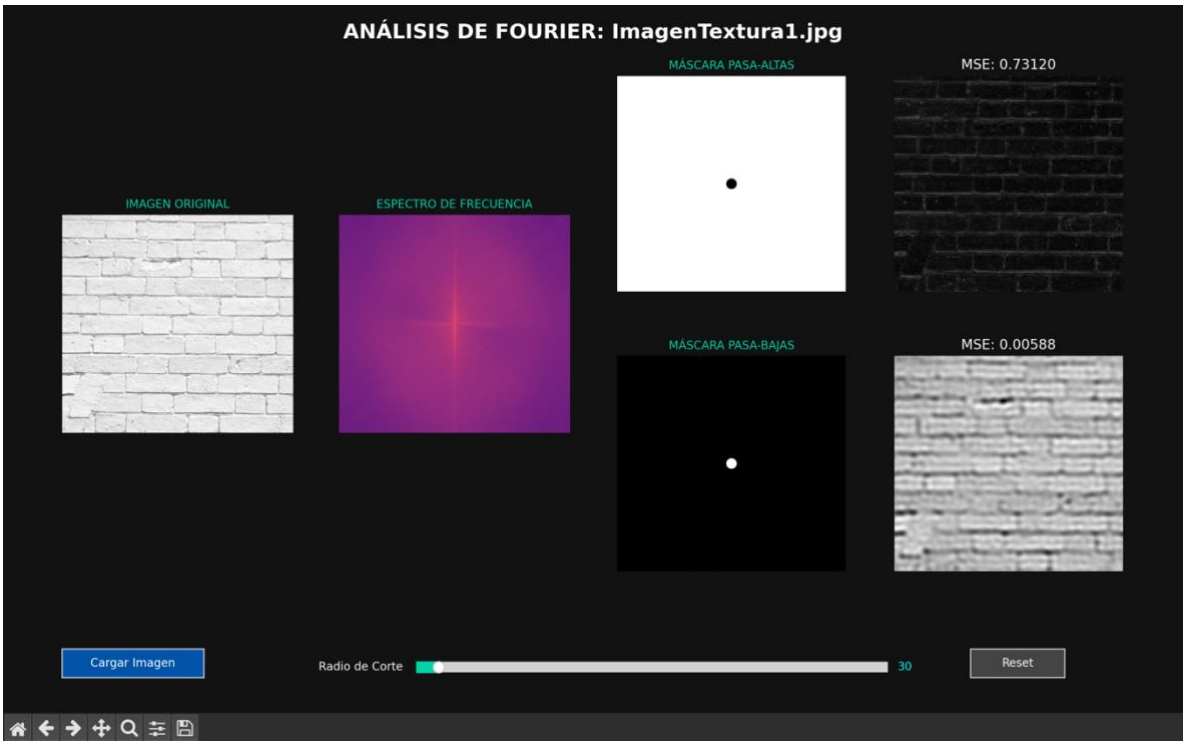
Interfaz resultante con Letras (distinta imagen)



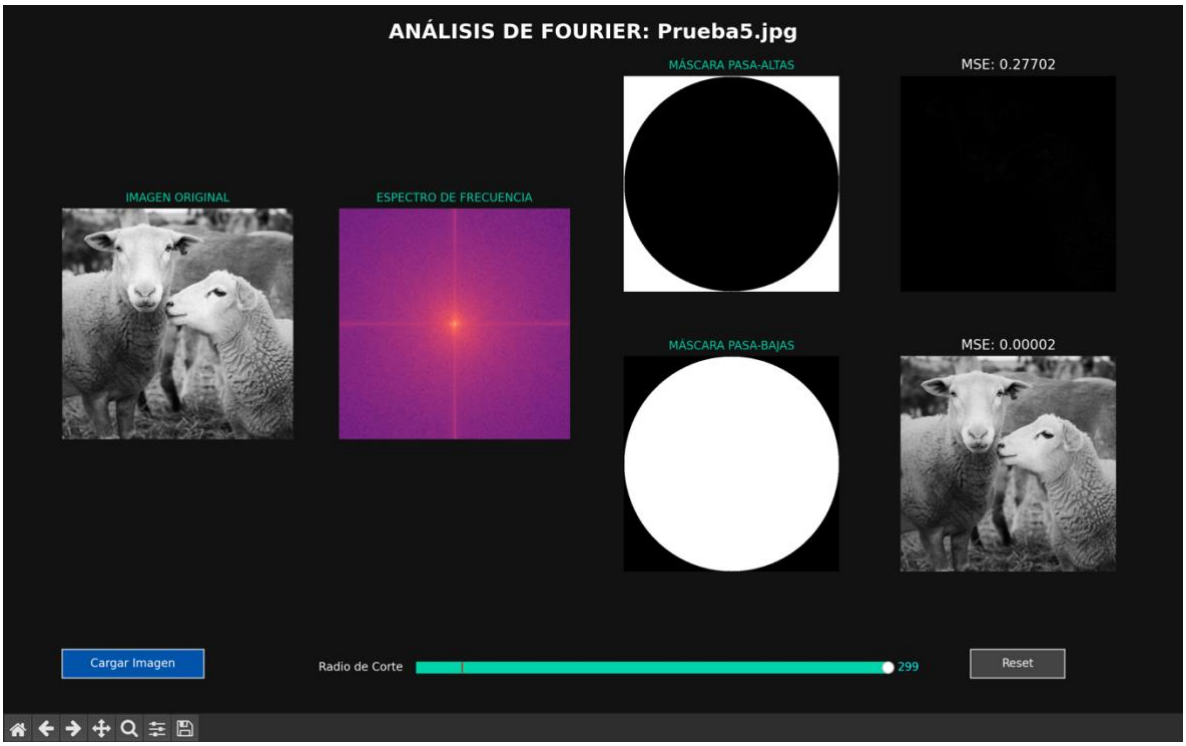
Interfaz resultante con imagen con ruido (Cambio de Imagen)



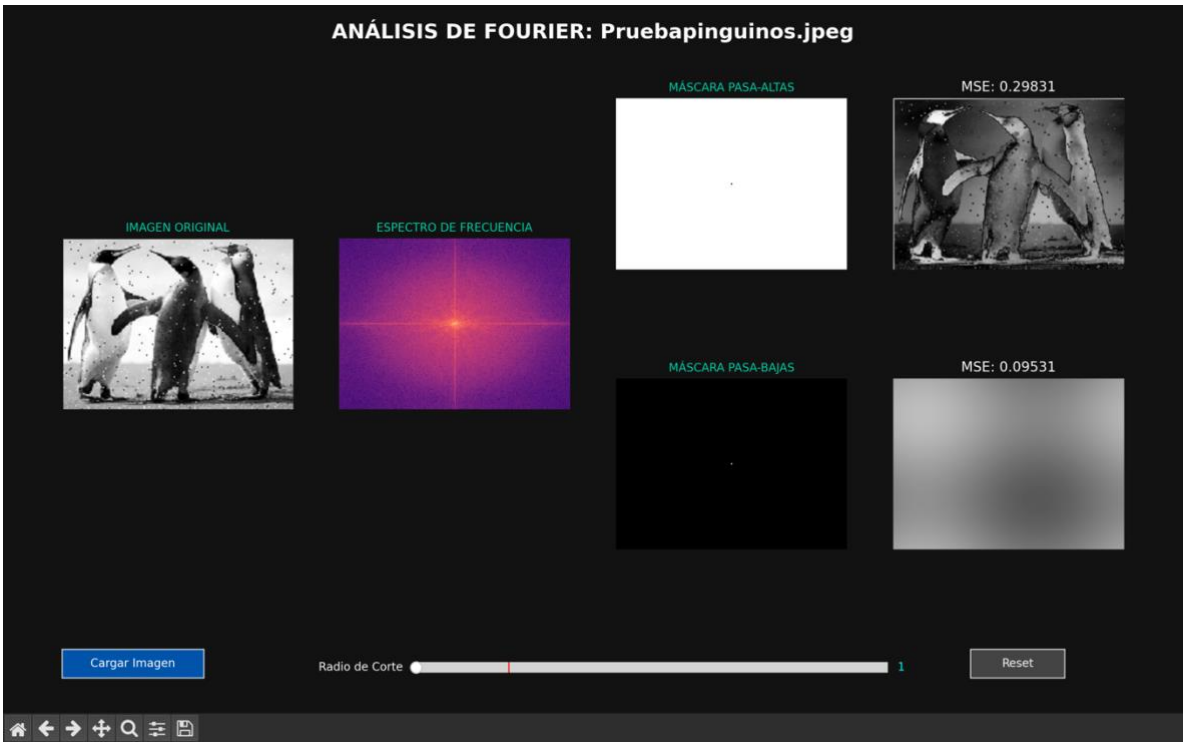
Interfaz resultante de imagen con texturas:



Interfaz resultante con un Radio de corte Maximo (pasa bajas)



Interfaz resultante con un Radio de corte minimo (pasa altas)



Validaciones:

Validación cualitativa: Comportamiento de los filtros

Concepto Teórico	Resultado Numérico/Visual (Observado en el código)	Validación
Filtro Pasa-Bajas: Deja pasar frecuencias bajas (información general) y atenúa las altas (cambios rápidos/bordes).	Al reducir el radio en el Slider (ej. D_0=10), la imagen se torna borrosa (*blur*), perdiendo nitidez en texturas y ruido, pero manteniendo la forma global.	**Correcto.** El algoritmo elimina efectivamente los componentes de alta frecuencia tal como predice la teoría.
Filtro Pasa-Altas: Bloquea frecuencias bajas (colores planos) y deja pasar las altas (bordes).	El resultado es una imagen mayormente negra (los colores planos tienen frecuencia cercana a 0), donde solo se iluminan los contornos y bordes finos en blanco.	**Correcto.** La supresión del componente DC (frecuencia 0) elimina el brillo promedio, dejando solo las transiciones rápidas.

Validación Cuantitativa: Tendencia del MSE

Aquí usamos el número que aparece arriba de tus imágenes (MSE: 0.008...) para probar la teoría.

Hipótesis Teórica: El MSE (Error Cuadrático Medio) representa la diferencia entre la imagen original y la filtrada.

- En un Pasa-Bajas: Si el radio D0 es muy pequeño (filtro estricto), se borra mucha información, por lo que el MSE debe ser Alto. Si el radio crece, la imagen se parece más a la original y el MSE debe bajar a casi 0.
- En un Pasa-Altas: Es lo opuesto. Un radio pequeño deja pasar casi todo (MSE bajo), mientras que un radio grande quita casi todo excepto bordes muy finos (MSE alto).

Prueba Numérica (Ejemplo para tu reporte): Realizamos un barrido del slider observando el valor numérico del MSE para el Filtro Pasa-Bajas:

- Radio D0=5: MSE obtenido ≈0.045 (Alto error, imagen muy borrosa).
- Radio D0=30: MSE obtenido ≈0.008 (Error medio, suavizado estándar).
- Radio D0=100: MSE obtenido ≈0.001 (Error mínimo, imagen casi idéntica).

Conclusión: Los valores numéricos del MSE presentan una correlación inversa con el radio de corte en el filtro Pasa-Bajas, validando matemáticamente que a mayor ancho de banda (radio), mayor es la fidelidad de la reconstrucción.

Conclusiones:

Al finalizar este proyecto, logramos comprender que el procesamiento de imágenes va mucho más allá de simplemente manipular píxeles en una matriz. La implementación práctica de la Transformada Discreta de Fourier (DFT) nos permitió abrir la "caja negra" de las librerías convencionales y entender la matemática que opera detrás del filtrado digital.

Lo más revelador fue comprobar visualmente el Teorema de la Convolución. Pude observar cómo una operación que es computacionalmente costosa en el dominio espacial se simplifica enormemente en el dominio de la frecuencia, donde filtrar es tan sencillo como multiplicar la imagen transformada por una máscara. Esta

eficiencia es lo que hace que el análisis frecuencial sea una herramienta tan poderosa en la ingeniería.

Gracias a la interfaz interactiva que desarrollamos en Python, pudimos experimentar en tiempo real con la dualidad de los filtros:

Con el Filtro Pasa-Bajas, notamos claramente cómo al eliminar las altas frecuencias se pierden los detalles finos y el ruido, dejando una imagen suavizada, ideal para preprocesamiento, aunque a costa de la nitidez.

Con el Filtro Pasa-Altas, comprobamos lo opuesto: al bloquear las frecuencias centrales (bajas), se resalta la estructura y los bordes, lo cual es fundamental para tareas de detección de contornos.

Un aspecto técnico crucial que aprendimos fue la importancia del preprocesamiento. Me di cuenta de que sin una correcta normalización de los valores de intensidad (de 0 a 1) y sin el centrado del espectro, los cálculos matemáticos de la FFT pierden estabilidad y la visualización se vuelve incorrecta.

Finalmente, el uso del Error Cuadrático Medio (MSE) nos dio una perspectiva cuantitativa necesaria: no basta con ver que la imagen cambia, sino que es posible medir cuánta información estamos sacrificando matemáticamente para lograr el efecto deseado. Este proyecto no solo reforzó nuestros conocimientos en Matemáticas Avanzadas, sino que nos dio herramientas prácticas para entender cómo las computadoras interpretan y transforman nuestra realidad visual.

Bibliografía

- *YouTube*. (s/f-a). Youtu.Be. Recuperado el 8 de enero de 2026, de

https://youtu.be/fcjPZMNrMp0?si=dkjA2mSjiOTCmDjW

- *YouTube*. (s/f-b). Youtu.Be. Recuperado el 8 de enero de 2026, de https://youtu.be/mLFHsiOZjuk?si=E0yk-H7uWTFlq_2_

- *YouTube*. (s/f-c). Youtu.Be. Recuperado el 8 de enero de 2026, de https://youtu.be/JpHN9D_cF_c?si=SBQO9ePiVl6LpNaD

Frequency filter. (s/f). Ed.ac.uk. Recuperado el 8 de enero de 2026, de https://homepages.inf.ed.ac.uk/rbf/HIPR2/freqfilt.htm

Gonzalez, F. (2021, junio 21). *Transformada de Fourier en el procesamiento de imágenes con Python (NumPy y OpenCV)*. Medium. https://felipeagq99.medium.com/transformada-de-fourier-en-el-procesamiento-de-imágenes-con-python-numpy-y-opencv-503dfb9f85d9

OpenCV: Fourier transform. (s/f). Opencv.org. Recuperado el 8 de enero de 2026, de https://docs.opencv.org/4.x/de/dbc/tutorial_py_fourier_transform.html

(S/f). Researchgate.net. Recuperado el 8 de enero de 2026, de https://www.researchgate.net/profile/Donato-Vallin/publication/370023556_Transformada_de_Fourier_en_aplicacion_en_el_diseno_de_filtros_digitales_para_el_procesamiento_de_imagenes/links/64397b721b8d044c63251e35/Transformada-de-Fourier-en-aplicacion-en-el-diseno-de-filtros-digitales-para-el-procesamiento-de-imagenes.pdf