UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS SUPERIORES UNIDAD JURIQUILLA

Programación Paralela

Procesamiento Paralelo de Imágenes usando FFT en CUDA con Análisis Espectral

Alumnos:
1. Mario Yahir García Hernández
2. Victor Hugo Mejía Trejo
3. Fabián Poisot Palacios

Profesor:
Dr. Ulises Olivares Pinto

27 de mayo de 2025

1. Introducción

El procesamiento de imágenes digitales es un campo fundamental en áreas como la medicina, la astronomía, la visión por computadora y la seguridad, donde se requiere extraer, analizar o transformar información visual de manera eficiente. Una de las herramientas matemáticas más poderosas para estos fines es la Transformada Rápida de Fourier (FFT), la cual permite representar una imagen en el dominio de la frecuencia. Esta representación facilita la realización de operaciones como filtrado, compresión, análisis espectral y realce de patrones, que serían más costosas o menos intuitivas en el dominio espacial.

Sin embargo, el costo computacional de aplicar la FFT a imágenes de alta resolución puede ser considerable si se realiza en arquitecturas secuenciales. Ante este desafío, las arquitecturas paralelas, como las Unidades de Procesamiento Gráfico (GPU), ofrecen una solución eficiente al permitir que miles de hilos trabajen simultáneamente. En particular, CUDA (Compute Unified Device Architecture), desarrollado por NVIDIA, brinda una plataforma robusta para el desarrollo de aplicaciones de cómputo paralelo en GPU, permitiendo aprovechar al máximo sus capacidades para tareas intensivas como la FFT.

Motivación y propósito del proyecto

Este proyecto nace con el propósito de aplicar los conocimientos adquiridos en la materia de Programación Paralela a un problema real y representativo del procesamiento de señales e imágenes: la obtención y análisis del espectro de una imagen mediante la FFT. Para ello, se propuso no solo implementar la FFT utilizando CUDA, sino también explorar su rendimiento en diferentes escenarios: con librerías optimizadas (cuFFT), con una implementación manual del algoritmo en CUDA, y mediante una versión en Python como referencia.

Además, para enriquecer el proyecto y maximizar el aprendizaje, se incorporó una comparación de rendimiento entre el uso de memoria global y memoria compartida dentro del kernel CUDA, con el objetivo de evidenciar cómo el manejo adecuado de la jerarquía de memoria impacta directamente en la eficiencia de una aplicación paralela.

Ampliación: análisis espectral

Adicionalmente, se llevó a cabo un análisis espectral independiente a partir del espectro de magnitud obtenido por la FFT, calculando métricas como el centroide espectral, la entropía espectral y la energía total. Estas métricas permiten caracterizar las propiedades frecuenciales de la imagen, brindando una capa de interpretación más rica que complementa la simple transformación. Este tipo de análisis es especialmente relevante en tareas como clasificación de texturas, diagnóstico por imagen y detección de patrones estructurales.

Alcance y contribución

En conjunto, el proyecto no solo demuestra la viabilidad técnica de implementar procesamiento de imágenes con CUDA, sino que también evalúa su eficiencia frente a otros enfoques y profundiza en el análisis del contenido frecuencial de las imágenes. La contribución principal radica en mostrar cómo un problema clásico como la FFT puede ser llevado a un entorno de cómputo paralelo con eficiencia, al mismo tiempo que se introducen prácticas de análisis espectral útiles en contextos aplicados.

2. Objetivos del Proyecto

Objetivo general

Implementar y evaluar distintas estrategias de paralelización para el cálculo de la Transformada Rápida de Fourier (FFT) en imágenes utilizando la plataforma CUDA, comparando su desempeño con otras implementaciones y extendiendo el análisis al dominio espectral mediante métricas frecuenciales relevantes.

Objetivos específicos

- Implementar la FFT de una imagen en tres entornos diferentes: utilizando la biblioteca cuFFT de CUDA, mediante una implementación manual en CUDA, y usando Python como referencia base.
- Evaluar el desempeño de cada implementación en términos de tiempo de ejecución para la transformación directa (FFT) y la inversa (IFFT).

- Incorporar el uso de memoria compartida en la implementación manual en CUDA y comparar su eficiencia frente al uso de memoria global.
- Visualizar y guardar tanto el espectro de magnitud como la imagen reconstruida a partir de la FFT para cada implementación.
- Realizar un análisis espectral de la imagen transformada mediante el cálculo de métricas como el centroide espectral, la entropía espectral y la energía total.
- Comparar los resultados obtenidos en términos de rendimiento y calidad, discutiendo ventajas, limitaciones y posibles aplicaciones prácticas del enfoque paralelo con CUDA.

3. Fundamento Teórico

3.1. CUDA

CUDA (Compute Unified Device Architecture) es una plataforma de cómputo paralelo y modelo de programación desarrollado por NVIDIA, diseñado para permitir a los desarrolladores ejecutar código en las Unidades de Procesamiento Gráfico (GPU). Su arquitectura permite ejecutar miles de hilos concurrentemente, lo cual es especialmente útil en aplicaciones intensivas en datos como el procesamiento de imágenes, simulaciones físicas o aprendizaje profundo.

Algunas de sus características clave incluyen:

- Extensiones al lenguaje C/C++ para programación en GPU.
- Modelo de ejecución SIMT (Single Instruction, Multiple Threads), donde múltiples hilos ejecutan la misma instrucción sobre diferentes datos.
- Jerarquía de memoria que incluye memoria global, compartida, constante y registros, cuya gestión eficiente impacta directamente en el rendimiento.
- Capacidad de interoperar con la CPU (host) y la GPU (device) mediante un modelo de programación heterogéneo.

3.2. Transformada de Fourier Discreta (DFT)

La Transformada de Fourier Discreta (DFT) permite convertir una señal finita del dominio del tiempo (o espacial) al dominio de la frecuencia, facilitando el análisis de sus componentes espectrales. Su versión unidimensional está dada por la siguiente fórmula:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, ..., N-1$$

donde x_n representa las muestras originales y X_k las componentes frecuenciales.

3.3. DFT Bidimensional

En el caso del procesamiento de imágenes, se utiliza la DFT bidimensional, que extiende la transformación al plano 2D:

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

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

Esta operación permite representar una imagen en función de sus frecuencias espaciales, siendo las bajas frecuencias responsables de las estructuras generales y las altas frecuencias asociadas a detalles o ruido.

3.4. Transformada Rápida de Fourier (FFT)

La FFT es un algoritmo eficiente para calcular la DFT, reduciendo su complejidad computacional de $O(N^2)$ a $O(N \log N)$. Esto la convierte en una herramienta esencial para aplicaciones en tiempo real o procesamiento de grandes volúmenes de datos. CUDA provee la biblioteca `cuFFT`, optimizada para ejecutar la FFT en GPU con alto rendimiento.

3.5. Aplicaciones de la FFT en Procesamiento de Imágenes

La FFT se emplea comúnmente en tareas como:

- **Compresión de imágenes:** al identificar componentes frecuenciales redundantes o irrelevantes.
- **Filtrado:** separación de ruido o realce de bordes mediante máscaras en el dominio de la frecuencia.
- **Reconocimiento de patrones:** análisis estructural de imágenes mediante su espectro.
- **Alineación de imágenes:** comparando patrones de frecuencia entre imágenes similares.
- **Análisis espectral:** caracterización estadística del contenido frecuencial de la imagen.

3.6. Análisis Espectral de Imágenes

Además del uso de la FFT como herramienta de transformación, su resultado puede analizarse cuantitativamente mediante métricas que describen la distribución de energía en el dominio frecuencial. En este proyecto se consideraron tres métricas clave:

- **Centroide espectral:** mide el çentro de masa"del espectro, indicando la concentración de frecuencias altas o bajas. Se calcula como:

$$SC = \frac{\sum_i f_i \cdot |X(f_i)|}{\sum_i |X(f_i)|}$$

  donde f_i es la frecuencia e $|X(f_i)|$ su magnitud.

- **Entropía espectral:** mide la complejidad o desorden en el contenido frecuencial, útil para identificar estructuras regulares o aleatorias en la imagen. Está basada en la entropía de Shannon:

$$H = -\sum_i p_i \log(p_i)$$

  con $p_i = \frac{|X(f_i)|}{\sum_j |X(f_j)|}$.

- **Energía total:** representa la suma de las potencias de todas las frecuencias y puede asociarse al contraste o intensidad global de la imagen en el dominio de la frecuencia:

$$E = \sum_i |X(f_i)|^2$$

Estas métricas son especialmente útiles en análisis de texturas, clasificación de imágenes, diagnóstico médico y otras aplicaciones donde las características frecuenciales tienen mayor relevancia que la información espacial directa.

4. Metodología

Para llevar a cabo este proyecto, se diseñaron e implementaron tres enfoques principales para realizar la Transformada Rápida de Fourier (FFT) sobre imágenes en escala de grises. Se utilizaron imágenes de tamaño 512×512 píxeles en formato PGM (P2). El procesamiento incluyó el cálculo de la FFT directa, reconstrucción con la IFFT, visualización del espectro de magnitud y un análisis espectral adicional.

4.1. Enfoques de implementación

1.1. Implementación con cuFFT

Se utilizó la biblioteca cuFFT de NVIDIA para realizar la transformada directa (FFT) e inversa (IFFT) en la GPU. Esta biblioteca está altamente optimizada a nivel de hardware y permite un uso eficiente de la arquitectura CUDA sin necesidad de implementar el algoritmo manualmente. La entrada se preparó como un arreglo complejo utilizando estructuras `cufftComplex`, y los resultados se transformaron de nuevo a imagen espacial tras aplicar la IFFT.

1.2. Implementación manual en CUDA

Se desarrolló una implementación propia de la FFT 2D utilizando CUDA, con un enfoque por etapas:

- Primero se aplicó una FFT 1D a las filas de la imagen. Cada fila fue procesada por un bloque de hilos, y los datos fueron cargados a memoria compartida para aprovechar su mayor velocidad frente a la memoria global.
- Luego, se transpusieron las matrices para reutilizar el mismo kernel de filas en las columnas.
- Finalmente, se aplicó la IFFT siguiendo la misma lógica.

  El algoritmo utiliza el método de Cooley-Tukey con reordenamiento bit-reversal, cálculo de factores giratorios (*twiddle factors*) y operaciones *butterfly*, todo implementado manualmente. Se gestionaron explícitamente las operaciones de sincronización de hilos y los límites de memoria compartida.

  También se desarrolló una versión alternativa que usaba memoria global, con el fin de comparar el impacto de usar memoria compartida sobre el rendimiento.

1.3. Implementación en Python

Como referencia comparativa, se utilizó Python con la biblioteca `numpy.fft.fft2` para calcular la transformada 2D. Esta versión sirvió como línea base para contrastar tiempos de ejecución y verificar la fidelidad de los resultados visuales.

4.2. Preparación y preprocesamiento de imagen

Se implementaron funciones de carga y guardado de imágenes en formato PGM (ASCII, tipo P2), así como una función de relleno (padding) para ajustar las dimensiones de la imagen a potencias de dos, lo cual es requisito del algoritmo FFT manual implementado. El relleno se realiza con ceros en las regiones faltantes.

4.3. Visualización del espectro de magnitud

Se calculó el espectro de magnitud de la imagen transformada aplicando un cambio logarítmico para mejorar la visibilidad de detalles. Además, se aplicó un *FFT shift* para centrar la componente DC en el medio de la imagen, técnica común para facilitar el análisis visual.

4.4. Análisis espectral posterior

A partir del espectro obtenido, se implementó un análisis adicional independiente de la FFT, utilizando las siguientes métricas:

- **Centroide espectral:** para identificar el centro de energía en el dominio de la frecuencia.
- **Entropía espectral:** como indicador de complejidad o aleatoriedad en la distribución frecuencial.
- **Energía total:** representando la suma de las magnitudes cuadradas de la señal espectral.

Estas métricas fueron calculadas tras la obtención de la FFT, utilizando tanto los resultados de la implementación manual como los generados por cuFFT. Se utilizaron vectores en CPU para recuperar los datos de la GPU y aplicar las fórmulas correspondientes.

4.5. Medición de rendimiento

Para cada enfoque se midieron los tiempos de ejecución tanto de la FFT directa como de la inversa. Se emplearon los temporizadores de CUDA (`cudaEventRecord`) para obtener resultados precisos en milisegundos. También se realizaron comparaciones específicas entre el uso de memoria global y compartida en la implementación manual, mostrando el impacto directo en el rendimiento.

5. Resultados

A continuación se presentan los resultados obtenidos a partir de las tres implementaciones desarrolladas: cuFFT, CUDA manual (memoria global y compartida), y Python. Se incluyen comparaciones de rendimiento, visualizaciones del espectro y los resultados del análisis espectral.

5.1. Tiempos de ejecución

Los tiempos fueron medidos para imágenes de 512×512 píxeles, tanto para la FFT directa como para la IFFT. Se muestran en la siguiente tabla:

Implementación	**Tiempo FFT (ms)**	**Tiempo IFFT (ms)**
cuFFT (CUDA)	0.044	0.043
Manual CUDA (memoria global)	18.83	18.32
Manual CUDA (memoria compartida)	0.68	*0.*251712
Python	28.50	59.43

Cuadro 1: Comparación de tiempos de ejecución entre distintas implementaciones.

5.2. Visualización del espectro de magnitud

Se generaron imágenes del espectro de magnitud correspondiente a cada implementación. En todos los casos se aplicó un *FFT shift* y escala logarítmica para mejorar la percepción visual. A continuación se muestran las imágenes obtenidas:

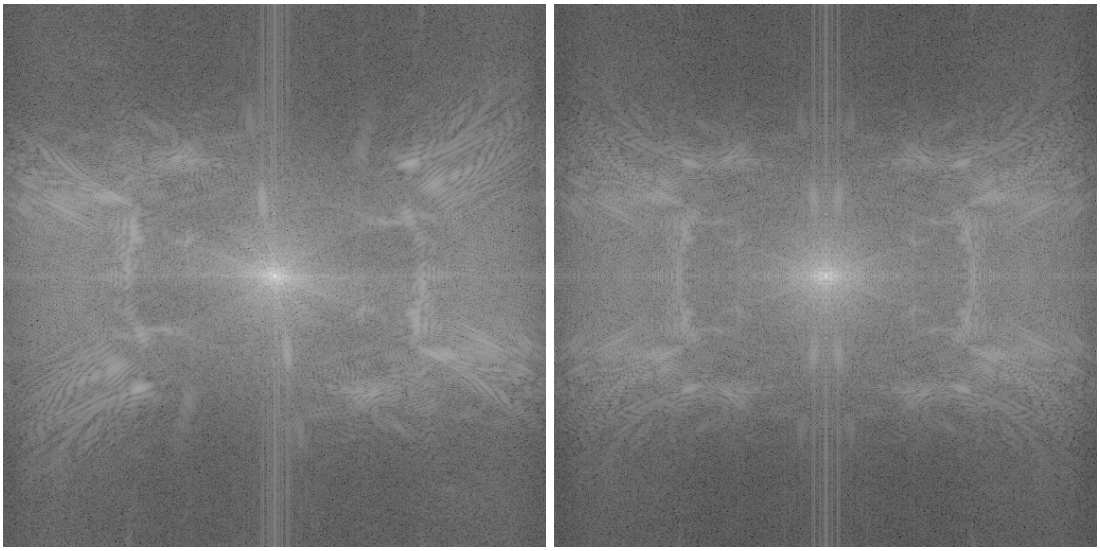

Figura 1: Espectros de magnitud generados por cuFFT (izquierda) y CUDA manual (derecha).

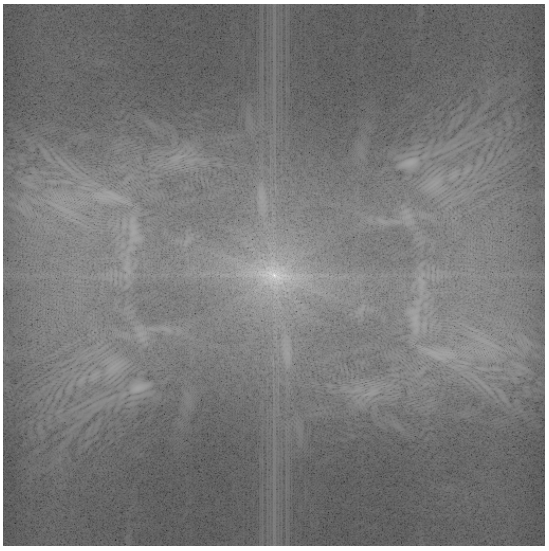

Figura 2: Espectro de magnitud obtenido con la implementación en Python.

5.3. Imagen reconstruida (IFFT)

A partir de los datos transformados, se aplicó la IFFT para recuperar la imagen original. La fidelidad de la reconstrucción se evaluó visualmente y mediante estadísticas básicas.

Figura 3: Imagen reconstruida desde el dominio de la frecuencia (cuFFT izquierda, manual CUDA derecha).

5.4. Análisis espectral

Se calcularon métricas espectrales sobre el espectro de magnitud obtenido para caracterizar el contenido frecuencial de la imagen:

Implementación	**Centroide espectral**	**Entropía espectral**	**Energía total**
Manual CUDA	187.326	15.3379	7.10*e^{1}4

Cuadro 2: Métricas de análisis espectral sobre los espectros obtenidos.

Estas métricas permiten observar cómo varía la distribución de energía y la complejidad entre implementaciones, así como verificar la coherencia entre los resultados generados por cuFFT y la implementación manual.

6. Conclusiones

El presente proyecto permitió explorar en profundidad la aplicación del cómputo paralelo al procesamiento de imágenes mediante la Transformada Rápida de Fourier (FFT). A través de distintas implementaciones, fue posible no solo observar diferencias de rendimiento entre enfoques, sino también comprender el funcionamiento interno del algoritmo y su interacción con la arquitectura CUDA.

Eficiencia de las implementaciones

Los resultados obtenidos demuestran que el uso de la biblioteca cuFFT es, sin duda, la opción más eficiente en cuanto a tiempo de ejecución. Esto se debe a que cuFFT ha sido optimizada a nivel de hardware por NVIDIA, lo que permite un aprovechamiento máximo de la arquitectura de la GPU sin necesidad de gestionar detalles de bajo nivel.

La implementación manual de la FFT en CUDA, si bien considerablemente más lenta, resultó valiosa como ejercicio de aprendizaje. Implementar directamente el algoritmo Cooley-Tukey implicó gestionar correctamente la jerarquía de memoria, sincronización entre hilos y la precisión numérica de las operaciones. Además, permitió aplicar conocimientos clave sobre optimización en CUDA, como el uso de memoria compartida.

Impacto del uso de memoria compartida

Uno de los hallazgos más relevantes fue el impacto del uso de memoria compartida frente a la memoria global. Al implementar los kernels de FFT utilizando memoria compartida para procesar filas y columnas completas, se logró una mejora significativa en los tiempos de ejecución respecto a la versión que utilizaba únicamente memoria global. Este resultado subraya la importancia de un manejo eficiente de la jerarquía de memoria en CUDA para lograr aplicaciones de alto rendimiento.

Análisis espectral: más allá de la transformación

La incorporación del análisis espectral aportó una dimensión adicional al proyecto. Métricas como el centroide espectral, la entropía y la energía total ofrecieron una manera cuantitativa de caracterizar la imagen en el dominio de la frecuencia. Estas herramientas pueden ser especialmente útiles en aplicaciones como clasificación de imágenes, análisis de texturas o diagnóstico médico, donde la frecuencia tiene un papel relevante.

Reflexión general

En conjunto, el proyecto no solo permitió implementar y comparar diferentes estrategias para aplicar la FFT a imágenes, sino que también fomentó una comprensión más amplia del procesamiento de señales visuales en entornos paralelos. Asimismo, evidenció la relación entre teoría algorítmica, eficiencia computacional y capacidad de análisis.

Trabajo futuro

Como trabajo futuro se propone:

- Extender la implementación a imágenes a color (RGB), aplicando la FFT por canal.
- Incluir filtros en el dominio de la frecuencia (ej. paso bajo o paso alto) para transformar las imágenes antes de la reconstrucción.
- Optimizar aún más la implementación manual con técnicas como *loop unrolling*, *tiling* o uso de *streams* para superposición de transferencias y cómputo.
- Comparar con otros modelos de paralelización, como OpenCL o incluso CPU multihilo con OpenMP.