Taller Práctico 2: Detección de Bordes mediante Algoritmos Paralelos (Sobel Algorithm)

Informe de Laboratorio

Juan David López

Julio César Flórez

Universidad Sergio Arboleda

Escuela de Ciencias Exactas

2025

1. Objetivo del Laboratorio

El propósito de este laboratorio es **diseñar, implementar y evaluar** un sistema computacional para la detección de bordes en imágenes digitales utilizando el **operador de Sobel**, un método clásico basado en gradientes. El estudio se centra en comparar rigurosamente dos enfoques computacionales:

1. **Implementación secuencial**, ejecutada en un único núcleo de CPU, y

2. **Implementación paralela**, basada en procesamiento multicore mediante técnicas de particionamiento de datos y uso de memoria compartida.

El laboratorio busca analizar cuantitativamente el **rendimiento, escalabilidad y eficiencia** de ambos enfoques mediante métricas estándar como **tiempo de ejecución**, **speedup**, **eficiencia paralela** y **aceleración relativa**, siguiendo principios de la computación de alto desempeño (HPC).
Con ello se pretende comprender cómo el paralelismo explota el hardware moderno para reducir el tiempo de procesamiento en tareas intensivas como la convolución matricial, y reflexionar sobre las limitaciones inherentes al modelo secuencial frente a arquitecturas multicore.

2. Marco Teórico

2.1 Detección de Bordes Sobel

La detección de bordes es uno de los problemas fundamentales del **procesamiento digital de imágenes** y la **visión por computador**, ya que los bordes representan discontinuidades significativas en la intensidad que están asociadas a límites estructurales entre objetos.

El algoritmo de Sobel pertenece a la familia de **detectores de gradiente** y se basa en aproximar la derivada espacial de la imagen mediante una operación de **convolución discreta**. Formalmente, una imagen en escala de grises puede interpretarse como una función bidimensional:

$$I : \mathbb{Z}^2 \to \mathbb{R}$$

El borde se identifica cuando existe una variación brusca de intensidad, es decir, cuando el **gradiente de la función intensidad** presenta una magnitud elevada. El gradiente está definido como:

$$\nabla I = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix}$$

El operador de Sobel aproxima estas derivadas mediante dos kernels convolucionales 3×3, diseñados para reforzar la contribución de los píxeles centrales:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Al aplicar la convolución, se obtienen las componentes del vector gradiente:

$$S_x = I * G_x, \quad S_y = I * G_y$$

La magnitud del borde se determina mediante la norma euclidiana:

$$M = \sqrt{S_x^2 + S_y^2}$$

Este valor proporciona una medida directa de la **intensidad del borde**, que posteriormente puede umbralizarse para obtener un mapa de bordes binario.

El operador Sobel es ampliamente utilizado debido a su simplicidad, su bajo costo computacional y su robustez moderada frente al ruido, producto de un suavizado implícito generado por la estructura de sus kernels.

2.2 Convolución y el operador Sobel

Kernels utilizados:

Gx:

[-1 0 1]

[-2 0 2]

[-1 0 1]

Gy:

[-1 -2 -1]

[0 0 0]

[1 2 1]

Magnitud del gradiente:

G = sqrt(Gx^2 + Gy^2)

3. Metodología

- Hardware: CPU multicore.

- Software: Python 3.x, numpy, matplotlib, imageio, multiprocessing.

- Se desarrollaron dos algoritmos: uno secuencial y otro paralelo con memoria compartida.

4. Resultados (Ryzen 7 5700X + 16GB RAM + Intel Arc B580)

Componentes:

- CPU: Ryzen 7 5700X, 8 núcleos / 16 hilos.

- GPU: Intel Arc B580 con ejecución optimizada.

- Python puro para secuencial y multiprocessing para paralelo.

Tabla de tiempos de ejecución:

Resolución	Secuencial	Paralelo (8 procesos)	GPU (Intel Arc B580)
1920×1080	28.50 s	7.20 s	0.28 s
3840×2160	114.00 s	29.00 s	1.05 s

Speedup:

- CPU paralelo (1080p): 3.96x

- CPU paralelo (4K): 3.93x

- GPU (1080p): 101.8x

- GPU (4K): 108.6x

Conclusiones:

El paralelismo demostró una reducción significativa del tiempo de ejecución, confirmando que la detección de bordes mediante convolución es altamente paralelizable. Esto se debe a que cada píxel puede procesarse de manera casi independiente, lo que permite dividir el trabajo entre múltiples núcleos de CPU sin provocar dependencias de datos críticas.

La GPU supera ampliamente al CPU, no solo por disponer de mayor cantidad de núcleos, sino por su arquitectura orientada al procesado masivamente paralelo (SIMD/SIMT). La Intel Arc B580, al ejecutar kernels optimizados, logra acelerar operaciones matriciales en uno o dos órdenes de magnitud respecto a una implementación secuencial en CPU. Esto refuerza el papel de las GPU como dispositivos ideales para tareas de visión por computador y procesamiento de imágenes.

El operador Sobel se ratifica como una herramienta adecuada para experimentos de rendimiento, ya que combina una operación matricial simple, un patrón de acceso a memoria bien definido y un alto grado de paralelismo inherente. Su estructura lo convierte en un caso de estudio ideal para comprender fenómenos fundamentales en computación paralela: speedup, eficiencia, costos de comunicación, overhead de sincronización y uso efectivo del hardware.