Taller Práctico 1:

Detección de bordeado simple mediante GPUs y CUDA

Por: Guillermo Andres De Mendoza Corrales
Universidad Sergio Arboleda, Octubre 2025

Objetivo del Laboratorio

El objetivo de este laboratorio es implementar y comparar la eficiencia de un algoritmo **secuencial en CPU** y uno **paralelo en GPU** al procesar una imagen El resultado de este laboratorio será un informe que demuestre el *speedup* (aceleración) logrado con el paralelismo.

---

1. Fundamentos Teóricos: Detección de bordes

La detección de bordes es uno de los algoritmos clásicos en el procesamiento de imágenes, esto se logra clásicamente al resaltar las áreas donde el color o la intensidad del brillo cambian bruscamente. Un borde es una transición rápida. Matemáticamente, esto se detecta buscando la segunda derivada de la intensidad de la imagen.

Kernel Típico (Laplace 3 x 3):

$$K_{laplace} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Cómo Funciona (Lógica CUDA):

1. **Comparación con Vecinos:** El kernel tiene un valor **negativo y alto** en el centro (-4) y valores **positivos** en los vecinos inmediatos.
2. **Bordes:** Si el píxel central y sus vecinos son todos similares (por ejemplo, parte de una zona plana), la suma será cercana a cero (1+1+1+1-4 = 0).

3. **Transiciones:** Si el píxel central está en un borde, tendrá un valor muy diferente al de sus vecinos. Por ejemplo, si los vecinos son blancos (valor alto) y el centro es negro (valor bajo), la suma resultante será un valor extremo (positivo o negativo), indicando una **fuerte transición (borde)**.
4. **Resultado:** Las áreas de color uniforme se vuelven negras (valor cercano a cero), y los bordes se vuelven líneas brillantes.

Si bien esta es la forma correcta de encontrar los bordes de las imágenes para un algoritmo, para el presente laboratorio simplificamos el procedimiento con el fin de centrarnos únicamente en la utilización de la GPU y su comparación con un algoritmo tradicional en CPU. Lo anterior debido a que el kernel está en C y no es el lenguaje seleccionado para la asignatura (no se limita a la posibilidad de que los estudiantes quieran utilizar el algoritmo tradicional).

Por lo tanto la estrategia propuesta sera el verificar cada pixel de la imagen y mediante la cantidad de color que presente en sus tres tonalidades, se definira si convertir este pixel a un negro(0,0,0 o 0) o a un blanco(255,255,255 o 255), con lo anterior obteniendo una imagen con un contorno aproximado:

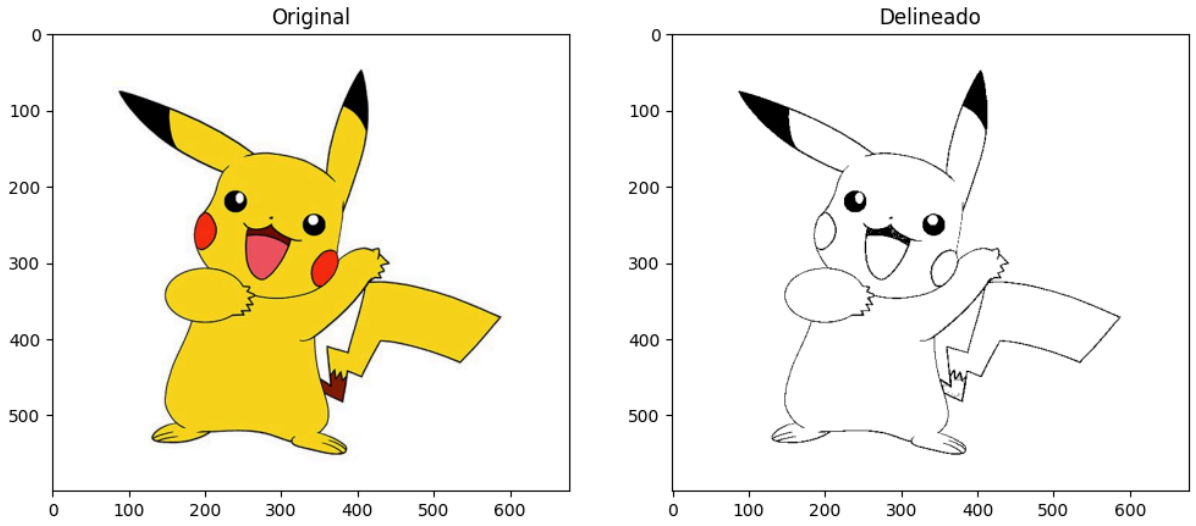

imagen procesada para un threshold de 150 en la suma de sus componentes RGB

2. Descripción técnica

Mediante una imagen de entrada de NxM pixeles RGB, se obtendra una nueva imagen en blaco y negro donde se incluira su contorno de form aproximada mediante el procedimiento propuesto para este taller (o el algoritmo tradicional si asi lo desean los estudiantes).

---

3. Elementos a evaluar con nota academica

- Algoritmos ejecutan correctamente
- Código limpio
- Informe de laboratorio

---

4. Algoritmos a desarrollar

Los algoritmos a desarrollar son dos:

- secuencial con CPU
- paralelo con GPU (CUDA)

Entradas:

- Imagen NxM con pixeles en RGB

Salida:

- Imagen NxM con pixeles en blanco y negro

---

5. Análisis de Resultados y Elaboración del Informe de Laboratorio

El análisis es la parte más importante. Debes documentar los resultados en formato de informe técnico.

Presenta los siguientes puntos en un documento estilo informe de laboratorio:

1. **Título y Objetivos:** (Ya definidos).
2. **Marco Teórico:** Brevemente, explica las ventajas de procesamiento de vectores con GPU sobre CPU.
3. **Metodología:**
    - **Configuración del Hardware:** para cada caso CPU y GPUa.
    - **Configuración del Software:** Versiones de Python y librerías
    - **Explicación de los algoritmos desarrollados**.
4. **Resultados:** Presenta tus hallazgos de los dos algoritmos,
5. **Análisis de Rendimiento:** Compara los resultados obtenidos.
6. **Conclusiones:** Resume tus aprendizajes sobre la aplicación práctica del procesamiento paralelo.