Taller Práctico 2:

Detección de bordeado mediante GPUs y CUDA (SOBEL ALGORITHM)

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 Detección de Bordes Sobel

La detección de bordes es una técnica fundamental en el procesamiento digital de imágenes que busca identificar los puntos de **cambio brusco de intensidad** en una imagen. Estos cambios son esenciales, ya que definen los límites de los objetos.

1.1. El Concepto de Borde como un Gradiente

En matemáticas, el cambio más rápido en una función se describe mediante el **gradiente**. En una imagen (que es una función bidimensional de intensidad I(x, y)), un borde se corresponde con un **alto valor del gradiente**.

El operador Sobel estima la **derivada** de la intensidad de la imagen en dos direcciones principales:

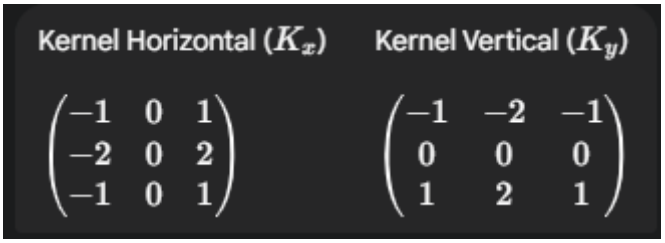
- **Derivada Horizontal (G_x):** Mide los cambios de intensidad en la dirección vertical, detectando **bordes verticales**.
- **Derivada Vertical (G_y):** Mide los cambios de intensidad en la dirección horizontal, detectando **bordes horizontales**.

1.2. La Convolución y el Operador Sobel

Los algoritmos de detección de bordes utilizan la operación de **convolución** (filtro) sobre la imagen. La convolución es la aplicación de una pequeña matriz llamada **núcleo** o **kernel** a cada píxel y sus vecinos.

A. Los Núcleos Sobel 3x3

El operador Sobel utiliza dos kernels 3×3 predefinidos:



Los valores 2 y -2 en el centro de los kernels le dan **más peso** a los píxeles que están justo al lado del píxel central, haciendo el filtro más sensible al cambio inmediato.

B. Cálculo de las Derivadas en el Kernel

El código implementa la convolución para un píxel (i, j) de la siguiente manera:

1. **Cálculo de G_x (Bordes Verticales):** El valor de G_x para el píxel (i, j) se obtiene multiplicando cada píxel vecino en la ventana 3 x 3 de la imagen en escalas de grises por el coeficiente correspondiente en el kernel K_x y sumando los resultados.
2. **Cálculo de G_y (Bordes Horizontales):** Se realiza el mismo proceso, pero utilizando el kernel K_y.

¿Qué Detecta G_x?

Este cálculo de G_x está diseñado para detectar **bordes verticales** (es decir, cambios de intensidad que ocurren en la dirección horizontal):

- **Ponderación Positiva (Columna Derecha):** Los píxeles a la derecha (j+1) tienen un peso positivo (+1 o +2).
- **Ponderación Negativa (Columna Izquierda):** Los píxeles a la izquierda (j-1) tienen un peso negativo (-1 o -2).
- **Ponderación Cero (Columna Central):** Los píxeles en el centro (j) tienen peso cero.

Si la intensidad de la imagen cambia bruscamente de **oscura (valores bajos) a clara (valores altos)** al moverse de izquierda a derecha, el resultado de G_x será un **valor positivo grande**. Si cambia de **clara a oscura**, el resultado será un **valor negativo grande**. Este valor grande indica la presencia de un borde vertical fuerte.

¿Qué Detecta G_y?

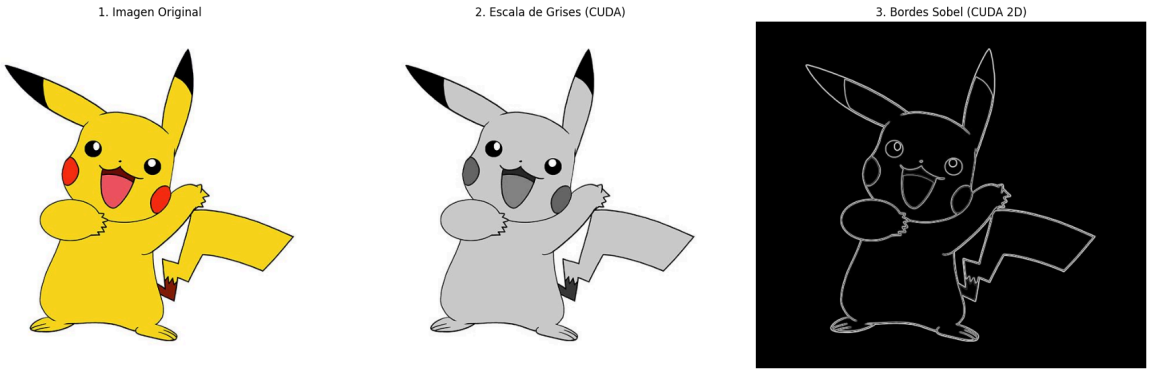
En este caso es lo mismo que G_x, solo que de forma horizontal

1.3. La Magnitud del Gradiente (El Borde Final)

Los resultados G_x y G_y representan las componentes del gradiente en dos direcciones. Para obtener el valor **final del borde (magnitud)**, que indica la fuerza del cambio de intensidad, se combinan utilizando la fórmula de la norma Euclidiana:

$$\text{Magnitude}(G) = \sqrt{G_x^2 + G_y^2}$$

El valor resultante, asignado a `la imagen resultado`, representa la **fuerza del borde** en ese píxel: un valor alto indica un borde fuerte (como la línea exterior de un objeto), y un valor bajo indica una región suave o uniforme.



2. Procedimiento

Con el fin de simplificar la explicación del marco teórico, se recomienda a los estudiantes seguir los siguientes pasos para llegar a una solución del taller:

Flujo general

- Cargar una imagen
- Pasar la imagen original a escala de grises
- Pasar la imagen de escala de grises a bordes

Lógica del kernel de detección de bordes, para cada pixel:

- Solo procesar los píxeles desde 1 < i < (n-1), tanto para filas o columnas. Esto con el fin de prevenir no sobrepasar las dimensiones de la imagen

- Obtener la matriz del pixel y sus alrededores (3x3)

$$P_i = \begin{pmatrix} I_{i-1,j-1} & I_{i-1,j} & I_{i-1,j+1} \\ I_{i,j-1} & I_{i,j} & I_{i,j+1} \\ I_{i+1,j-1} & I_{i+1,j} & I_{i+1,j+1} \end{pmatrix}$$

- Gx: realizar una suma de productos de nuestra matriz del pixel i por Kx (recomiendo hacer la operación manual, ya que como es un kernel, no tendremos las operaciones de numpy a las cuales puedan estar acostumbrados), donde obtendremos un valor Gx

$$G_x = \sum_{m=-1}^{1} \sum_{n=-1}^{1} P_{i+m,j+n} \cdot K_{x,m,n}$$

$$\begin{aligned} G_x &= I_{i-1,j-1}(-1) + I_{i-1,j}(0) + I_{i-1,j+1}(1) \\ &+ I_{i,j-1}(-2) + I_{i,j}(0) + I_{i,j+1}(2) \\ &+ I_{i+1,j-1}(-1) + I_{i+1,j}(0) + I_{i+1,j+1}(1) \end{aligned}$$

- Gy: multiplicar nuestra matriz del pixel i por Ky (recomiendo hacer la operación manual, ya que como es un kernel, no tendremos las operaciones de numpy a las cuales puedan estar acostumbrados)

$$G_y = \sum_{m=-1}^{1} \sum_{n=-1}^{1} I(i+m, j+n) \cdot K_y(m,n)$$

$$G_y = I_{i-1,j-1}(-1) + I_{i-1,j}(-2) + I_{i-1,j+1}(-1) +$$
$$I_{i,j-1}(0) + I_{i,j}(0) + I_{i,j+1}(0) +$$
$$I_{i+1,j-1}(1) + I_{i+1,j}(2) + I_{i+1,j+1}(1)$$

- Para calcular el gradiente de aproximación sqrt(Gx^2 + Gy^2), utilice el método SQRTF de cuda.libdevice

```
G = cuda.libdevice.sqrtf(Gx**2 + Gy**2)
```

- El valor G será el que debemos incorporar en nuestra imagen de salida

```
img_edges[i, j] = G
```

---

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:

- paralelo con GPU (CUDA) para detección de bordes con algoritmo SOBEL

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 elalgoritmo de SOBEL y su ejecucion en GPU.
3. **Metodología:**
    - **Configuración del Hardware**
    - **Configuración del Software:** Versiones de Python y librerías
    - **Explicación del algoritmo desarrollado**.
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.