ALGORITMO PARA APLICAR FILTRO DE SOBEL A IMÁGENES DE MUESTRA UTILIZANDO PROCESAMIENTO A TRAVÉS DE HOST (CPU) Y DEVICE (GPU)

EDISSON ALEXANDER RUIZ ROJAS

EST. INGENIERÍA DE SISTEMAS Y COMPUTACION

JOHN HAIBER OSORIO RIOS

ASIGNATURA: ARQUITECTURA DE COMPUTADORES

UNIVERSIDAD TECNOLOGICA DE PEREIRA

FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN

PEREIRA RISARALDA

AÑO 2015

ALGORITMO PARA APLICAR FILTRO DE SOBEL A IMÁGENES DE MUESTRA UTILIZANDO PROCESAMIENTO A TRAVÉS DE HOST (CPU) Y DEVICE (GPU)

Por medio de un algoritmo aplicado en CUDA se desea realizar filtro de sobel a imágenes de muestra y proceder a comparar tiempos de procesamiento ejecutados en el Host del computador (CPU) y el Device (GPU), la aceleración de algoritmos es algo muy útil cuando se trata de vectores o matrices de muchos elementos que usualmente suelen tomar mucho más tiempo de ejecución en la CPU, para la aceleración se utiliza CUDA y la plataforma http://judge.utp.edu.co:3000/compiler/textEditor por medio de la cual podemos correr y realizar pruebas a algoritmos que procesan muchos elementos.

PROCEDIMIENTO

1. Inicialmente se procede a cargar la imagen desde un archivo.

Figura1. Imagen de muestra 1 original

Se tienen seis imágenes de muestra de diferentes tamaños las cuales se van a cargar por medio de un servidor utilizando la ruta “./inputs/img*.jpg”.

2. Posteriormente se procederá a llevar la imagen a escala de grises.

Figura 2. Imagen de muestra 1 en escala de grises

Todas las imágenes de muestra en escala de grises se guardaran con el fin de verificar que el algoritmo está procesando las mismas a la escala correcta antes de aplicar el filtro de sobel.

3. Continuamos aplicando filtro de sobel por medio del algoritmo ejecutado tanto en la CPU como en la GPU.

Figura 3. Imagen de muestra 1 filtro de sobel Algoritmo Secuencial CPU

Figura 4. Imagen de muestra 1 filtro de sobel Algoritmo Paralelo GPU

4. Se toman los tiempos de ejecución de cada algoritmo con cada una de las imágenes de muestra. Los resultados obtenidos fueron los siguientes:

IMAGEN 1	SECUENCIAL	PARALELO
1	0,006201	0,000668
2	0,005947	0,000822
3	0,006079	0,000532
4	0,006479	0,000538
5	0,006254	0,000822
PROMEDIO	**0,006192**	**0,000676**

Tabla 1. Imagen 1 tamaño 580 X 580 pixeles

IMAGEN 2	SECUENCIAL	PARALELO
1	0,009493	0,000837
2	0,009897	0,000941
3	0,009316	0,000934
4	0,009536	0,000925
5	0,009420	0,000934
PROMEDIO	**0,009532**	**0,000914**

Tabla 2. Imagen 2 tamaño 638 X 640 pixeles

IMAGEN 3	SECUENCIAL	PARALELO
1	0,016185	0,001320
2	0,016554	0,001314
3	0,016327	0,001332
4	0,016333	0,001338
5	0,016018	0,001319
PROMEDIO	**0,016283**	**0,001325**

Tabla 3. Imagen 3 tamaño 1366 X 768 pixeles

IMAGEN 4	SECUENCIAL	PARALELO
1	0,055374	0,004226
2	0,054315	0,004481
3	0,048612	0,004206
4	0,068525	0,004208
5	0,049729	0,004011
PROMEDIO	**0,055311**	**0,004226**

Tabla 4. Imagen 4 tamaño 2560 X 1600 pixeles

IMAGEN 5	SECUENCIAL	PARALELO
1	0,267605	0,018875
2	0,267717	0,018872
3	0,264011	0,018742
4	0,266567	0,018944
5	0,261126	0,018411
PROMEDIO	**0,265405**	**0,018769**

Tabla 5. Imagen 5 tamaño 5226 X 4222 pixeles

IMAGEN 6	SECUENCIAL	PARALELO
1	0,189591	0,014315
2	0,191358	0,016824
3	0,194240	0,016814
4	0,185044	0,015755
5	0,188565	0,015964
PROMEDIO	**0,189760**	**0,015934**

Tabla 6. Imagen 6 tamaño 4928 X 3264 pixeles

5. Posteriormente se procede a realizar una gráfica de los datos obtenidos anteriormente para observar el comportamiento de ambos algoritmos (secuencial y paralelo) con respecto al tamaño de las imágenes de muestra que fueron procesadas en las pruebas. Las gráficas obtenidas fueron las siguientes:

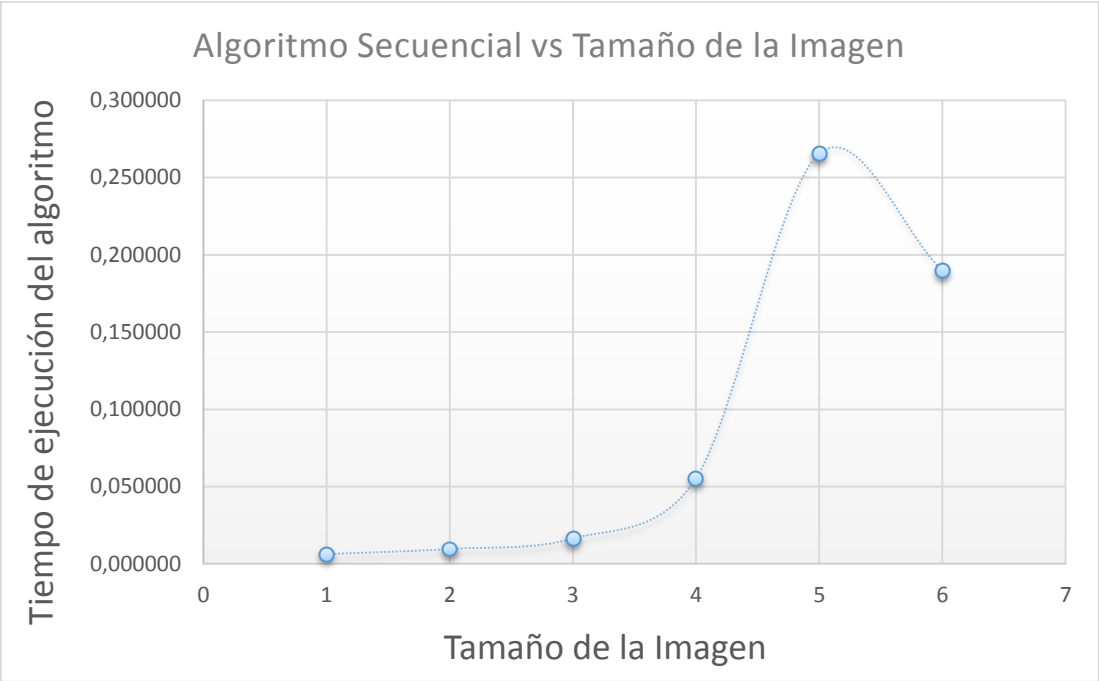

Figura 5. Tiempo empleado por el algoritmo secuencial para procesar la imagen.

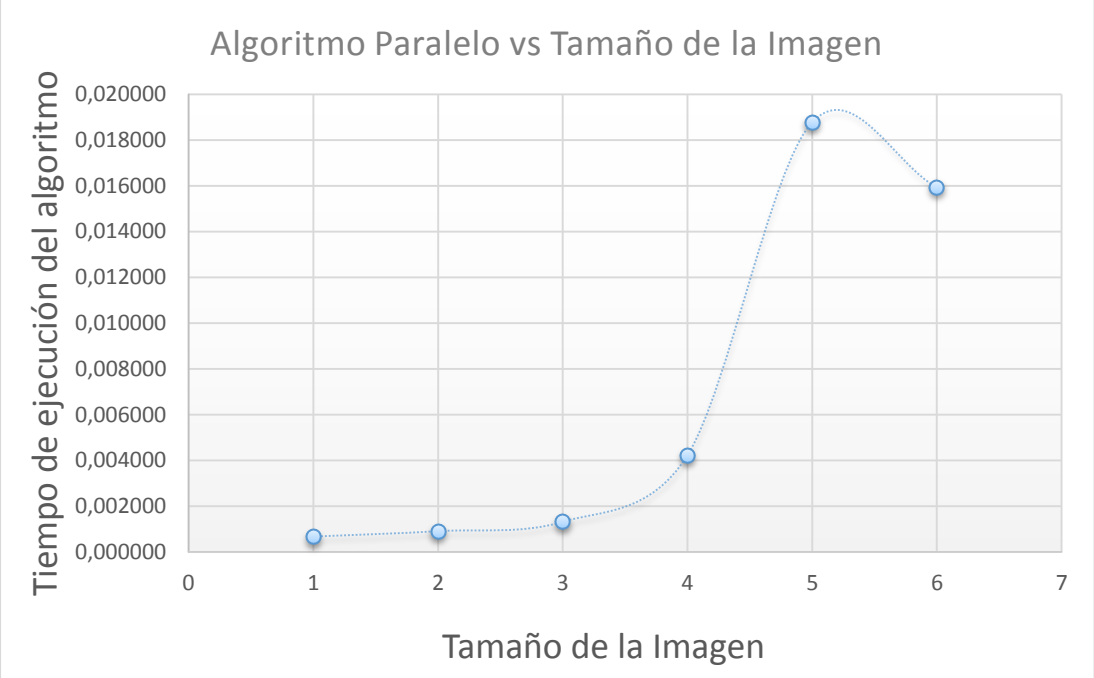

Figura 6. Tiempo empleado por el algoritmo secuencial para procesar la imagen.

6. Gráficas del comportamiento de ambos algoritmos con respecto al tiempo y aceleración obtenida en el procesamiento de imágenes,

IMAGEN	SECUENCIAL	PARALELO	ACELERACION (X)
1	0,006192	0,000676	9,15
2	0,009532	0,000914	10,43
3	0,016283	0,001325	12,29
4	0,055311	0,004226	13,09
5	0,265405	0,018769	14,14
6	0,189760	0,015934	11,91

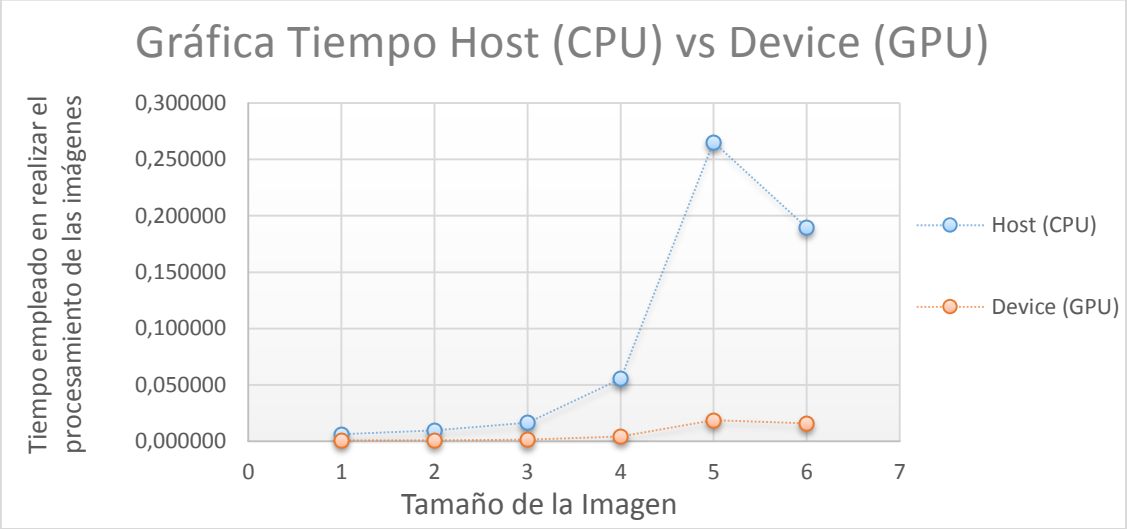

Figura 7. Comportamiento de ambos algoritmos para procesar las imágenes según el tamaño.

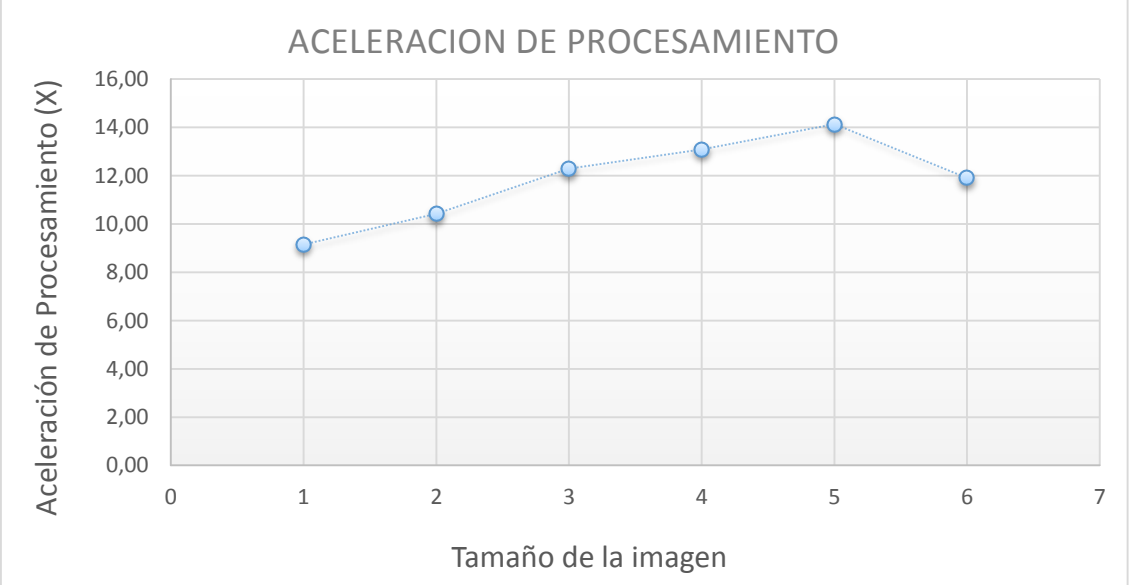

Figura 8. Aceleración obtenida al comparar el tiempo de ambos algoritmos.

CONCLUSIONES

1. Al observar la figura 5 se puede deducir que el tiempo de ejecución del algoritmo secuencial para procesar las imágenes aumenta proporcionalmente al tamaño de la imagen; en la figura se observa que al final existe una curvatura hacia abajo debido a que la imagen 6 es un poco más pequeña que las anteriores lo cual genera la caída del tiempo de ejecución.

2. La aceleración de la GPU para procesar imágenes de grandes dimensiones aumenta proporcionalmente al tamaño de la imagen y al final decrece por el tamaño de la imagen 6 que es más pequeña, lo anterior nos confirma que en efecto el algoritmo está tomando los valores correctos y su variación depende del tamaño de la imagen.

3. Al observar la figura 7 en donde se detalla el comportamiento de ambos algoritmos se puede notar que el algoritmo paralelo (procesado por la GPU) tiene un tiempo de ejecución mucho menor que el algoritmo secuencial (procesado por la CPU) con respecto al tamaño de las imágenes y llegado inclusive a tener una aceleración de 14x en la imagen más grande de la muestra.

4. La figura 8 de la aceleración nos muestra que esta aumenta conforme aumenta el tamaño de la matriz y elementos a calcular, lo anterior se debe a que el Device cuenta con muchos procesadores que pueden realizar las operaciones simultáneamente mientras que en la Host todos los algoritmos se realizan secuencialmente.

5. Es de anotar que el procedimiento aplicado se realizó para todas las imágenes de muestra y observando las figuras 3 y 4 se evidencia que no existe diferencia notoria entre el filtro de sobel aplicado a ambas imágenes con diferentes algoritmos; el resultado es igual la diferencia radica en el tiempo de ejecución que para el caso particular de esta muestra la aceleración corresponde aproximadamente a 9x.

6. En general se puede concluir que el uso de la GPU como procesador de algoritmos es una herramienta muy útil y que nos ayuda a reducir el tiempo de procesamiento de algoritmos que contengan muchos elementos.