SEGUNDO PARCIAL HPC (HIGH PERFORMANCE COMPUTING)

Ingeniería De Sistemas y Computación

Luis Fernando Salazar Escamilla

1088309888

Universidad Tecnológica de Pereira

Abril 2015

USO DEL FILTRO DE SOBEL PARA RESALTAR BORDES DE UNA IMAGEN USANDO DIFERENTES ALGORITMOS DE COMPUTACIÓN PARALELA

1. DETECCIÓN DE BORDES.

Los bordes de una imagen digital se pueden definir como transiciones entre dos regiones de niveles de gris significativamente distintos. Suministran una valiosa información sobre las fronteras de los objetos y puede ser utilizada para segmentar la imagen, reconocer objetos, etc. Y esto es frecuentemente usado en la visión artificial.

La mayoría de las técnicas para detectar bordes emplean operadores locales basados en distintas aproximaciones discretas de la primera y segunda derivada de los niveles de grises de la imagen.

2. OPERADOR SOBEL

El operador Sobel es utilizado en procesamiento de imágenes, especialmente en algoritmos de detección de bordes. Técnicamente es un operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad de una imagen. Para cada punto de la imagen a procesar, el resultado del operador Sobel es tanto el vector gradiente correspondiente como la norma de éste vector.

3. PROCEDIMIENTO

Para el desarrollo del algoritmo se utilizó el operador Sobel con el fin de detectar los bordes de una imagen. Primero que todo, mediante el uso de la librería OPENCV se pasa la imagen a formato de grises para poder procesarla, después de leer la imagen de entrada. Ya en formato de grises, se implementaron 3 algoritmos de computación paralela que ejecutara el filtro de Sobel, usando memoria global,

memoria constante y memoria compartida. Y para ejecutar el algoritmo secuencial se utiliza una función de la librería que permite aplicar el operador Sobel a la imagen.

4. DATOS

Se realizó un promedio de los datos para resultantes para poder tener mayor exactitud a la hora del análisis. En la ejecución de los algoritmos se tomaron los tiempos de ejecución de cada uno y después de esto se pasó a calcular la aceleración que tienen los algoritmos en paralelo con respecto al algoritmo secuencial usando la función de la librería OPENCV. A continuación se presenta la tabla de los promedios de los datos y las aceleraciones respectivas:

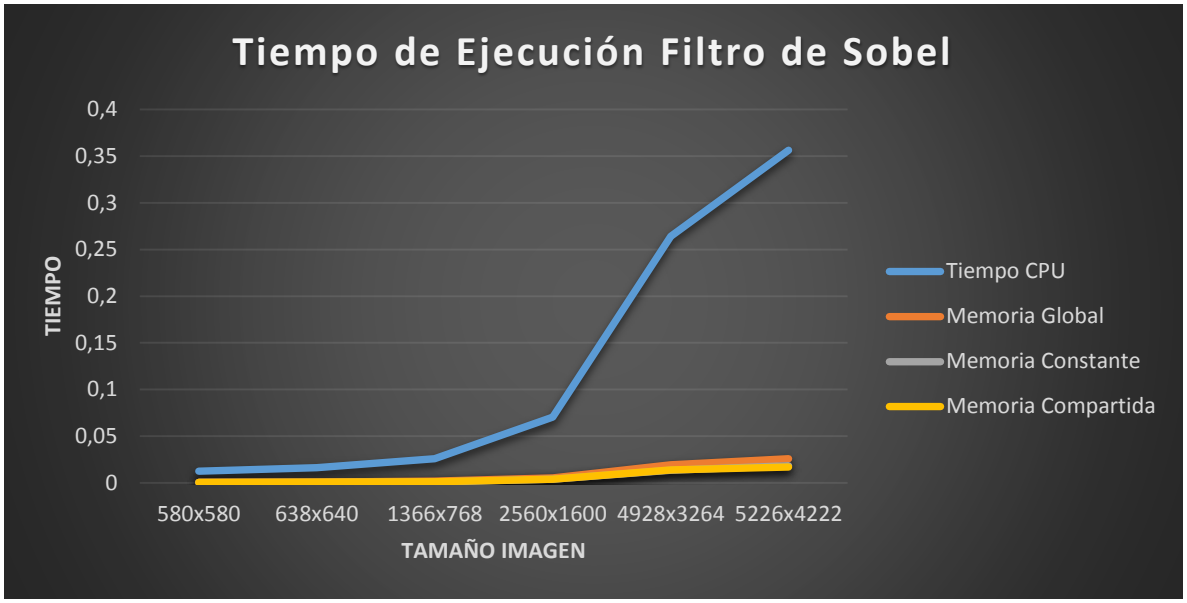
Tabla 1

Tamaño Imagen	TIEMPO CPU	TIEMPOS MEMORIA			ACELERACION MEMORIA		
		Global	Constante	Compartida	Global	Constante	Compartida
580x580	0,0123186	0,0005835	0,0003434	0,0004627	21,11156812	35,872452	26,62329803
638x640	0,0161104	0,0006987	0,0004323	0,0006718	23,05767855	37,2667129	23,98094671
1366x768	0,0259365	0,0017068	0,0009433	0,0011456	15,19598078	27,4954945	22,64010126
2560x1600	0,0703243	0,0053765	0,0039005	0,0040311	13,07994048	18,0295603	17,44543673
4928x3264	0,2640497	0,0192321	0,0135939	0,0138439	13,72963431	19,4241314	19,07336083
5226x4222	0,3565325	0,0259854	0,0184396	0,01698901	13,72049305	19,3351537	20,98606687

Elaboración Propia

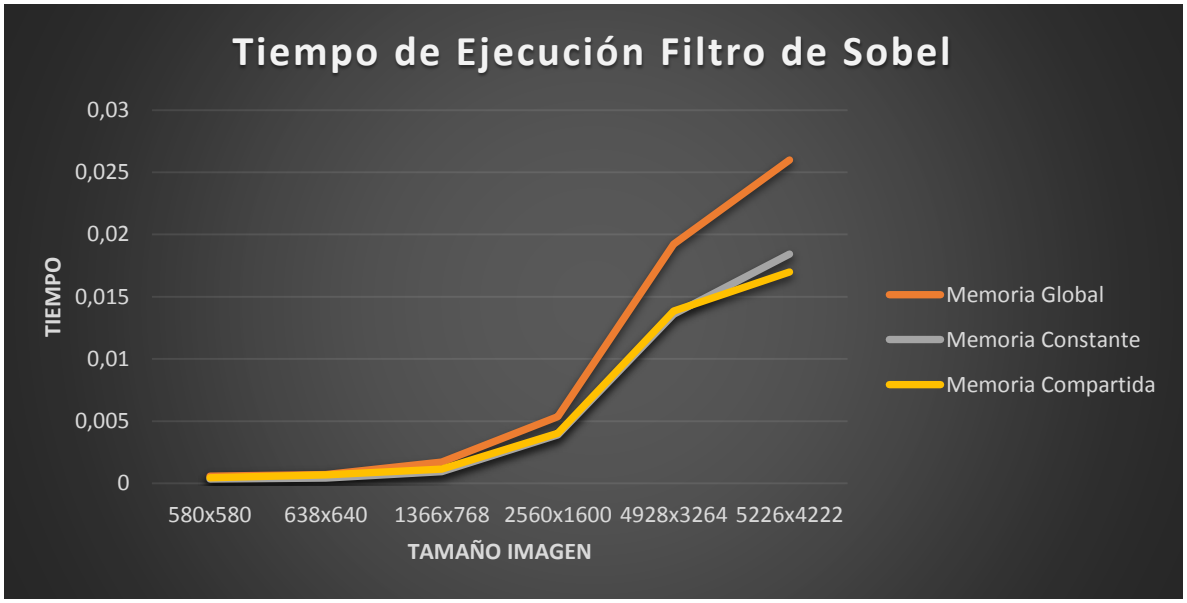
A partir de estos datos se realizaron las siguientes tablas para que ya se pueda interpretar el desempeño de los algoritmos con el tamaño de las imágenes y con cada uno de los algoritmos desarrollados.

Ilustración 1



Como se puede observar en la imagen ilustración anterior, el tiempo de ejecución del operador Sobel para la detección de bordes en una imagen es mucho más demorado en algoritmos secuenciales. Los algoritmos que usan programación paralela tienen un tiempo de ejecución mucho menor.

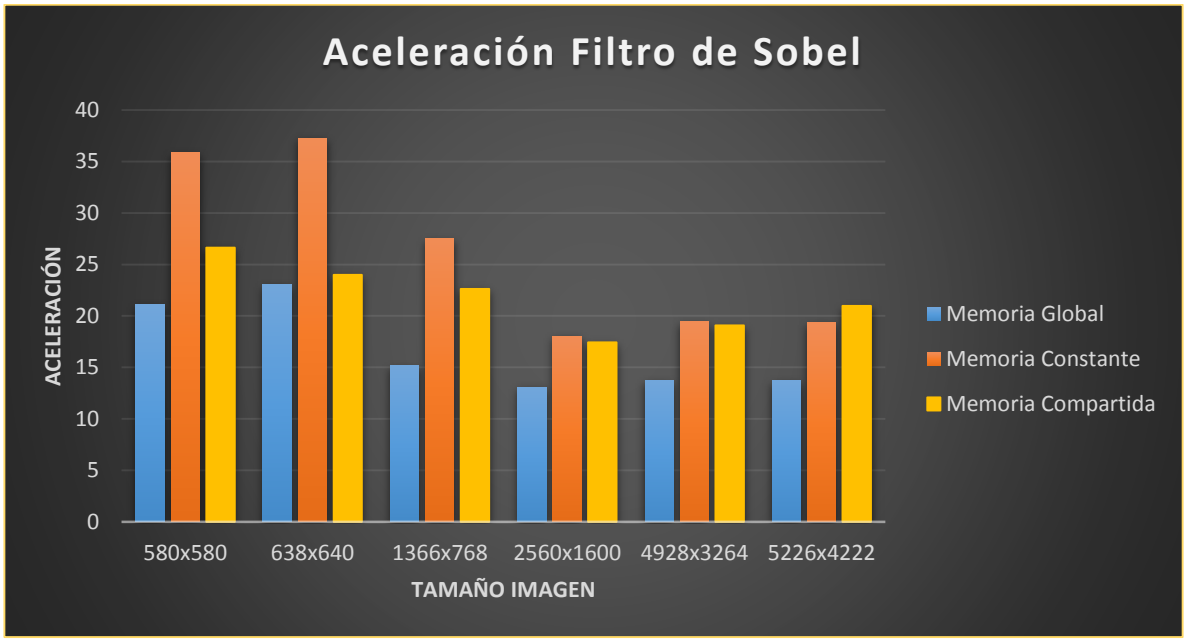
Ilustración 2



Para poder distinguir cuál de estos es más óptimo se presenta la anterior ilustración, ahí se puede observar que de los algoritmos en paralelo el algoritmo usando memora global es el menos optimo y que el desempeño de los algoritmos con memoria constante y memoria compartida es muy reñido hasta la imagen más grande donde el tiempo del algoritmo con memoria constante empieza a ser mayor.

A continuación se presenta una gráfica donde se pueda examinar las aceleraciones que tienen los algoritmos en paralelo con respecto al algoritmo secuencial.

Ilustración 3



En la ilustración 3 se puede analizar que las aceleraciones que presentan los algoritmos en paralelo con respecto al algoritmo secuencial son significativas y que con imágenes de dimensiones 1366x768 el algoritmo con memoria constante evidentemente tiene un mejor desempeño que los otros 2, pero cuando las imágenes se tornan más grandes el algoritmo con memoria compartida empieza a aumentar y el algoritmo con memoria constante disminuye significativamente.

5. Conclusiones

- Utilizar computación paralela para la detección de bordes de una imagen con ayuda de la librería OPENCV es mucho más eficaz que utilizar algoritmos secuenciales (Como se especificó esto con ayuda de la librería OPENCV no se analizó el proceso con otras librerías como para concluir que es el proceso más eficiente).

- Para imágenes de grandes dimensiones el algoritmo en paralelo utilizando memoria compartida es más eficiente que los demás.

- Para imágenes de dimensiones no muy grandes tanto el algoritmo con memoria constante y el algoritmo con memoria compartida son una buena opción para ejecución y tienen un nivel de eficiencia parecido.

- En cualquier caso el utilizar un algoritmo en paralelo con memoria global no es la mejor opción y los resultados de desempeño no son los mejores.

- Las aceleraciones que presentan los algoritmos en paralelo con respecto a un algoritmo en secuencial son significativas.