

Parcial 2

Convolución en 2D en imágenes

High Performance Speed

José Daniel Osorio Morales

Daniel Cardona Martinez



**UNIVERSIDAD TECNOLÓGICA DE PEREIRA
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
RISARALDA, PEREIRA
2015**

Introducción:

En este documento se muestran los resultados obtenidos del algoritmo de convolución en dos dimensiones a seis imágenes de diferentes resoluciones. A cada imagen se le aplica la convolución con un algoritmo secuencial y tres algoritmos paralelos con diferentes técnicas de uso de memoria: memoria global, constante y compartida. Se toman los tiempos de ejecución y la aceleración obtenida con los algoritmos paralelos respecto al algoritmo secuencial.

Para la versión secuencial del algoritmo se utilizó la librería de C, OpenCV que permite integrar toda la funcionalidad del filtro Sobel. Para las versiones en paralelo se utilizó la librería CUDA, para trabajar sobre GPU y crear algoritmos que trabajen en paralelo.

El procedimiento para aplicar el algoritmo se basó en la teoría expuesta en la página: https://es.wikipedia.org/wiki/Operador_Sobel.

Para la parte secuencial se usó las funciones `Sobel()`, `convertScaleAbs()`, `addWeighted()`.

Para la versión en paralelo, se crearon 2 Kernels, uno para realizar la convolución y otro Kernel para efectuar la ecuación descrita en el link de wikipedia.

Tabla de muestreo:

Presentaremos los datos resultantes de la toma de datos con cada una de las 6 imágenes, empezando por una tabla de tiempos promedio y otra de aceleración, que compara cada algoritmo secuencial con su parte en paralelo.

Tabla de tiempos promedio:

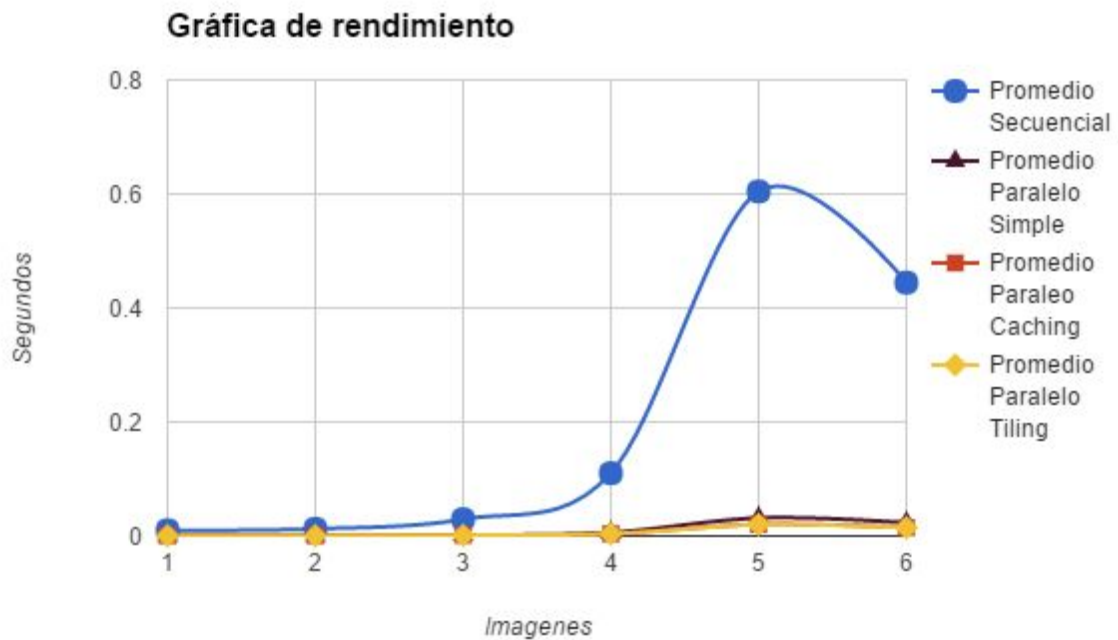
Imágenes	Promedio Secuencial	Promedio Paralelo Simple	Promedio Paralelo Caching	Promedio Paralelo Tiling
1	0.009966	0.000618450	0.0004352	0.000439
2	0.01254765	0.0007334	0.000525	0.0005164
3	0.0293942	0.0017273	0.00123125	0.00120625
4	0.1101705	0.0058539	0.0038918	0.0037757
5	0.60444915	0.0315709	0.02034055	0.0199492
6	0.44450095	0.0231443	0.0148941	0.0146058

Tabla de aceleraciones:

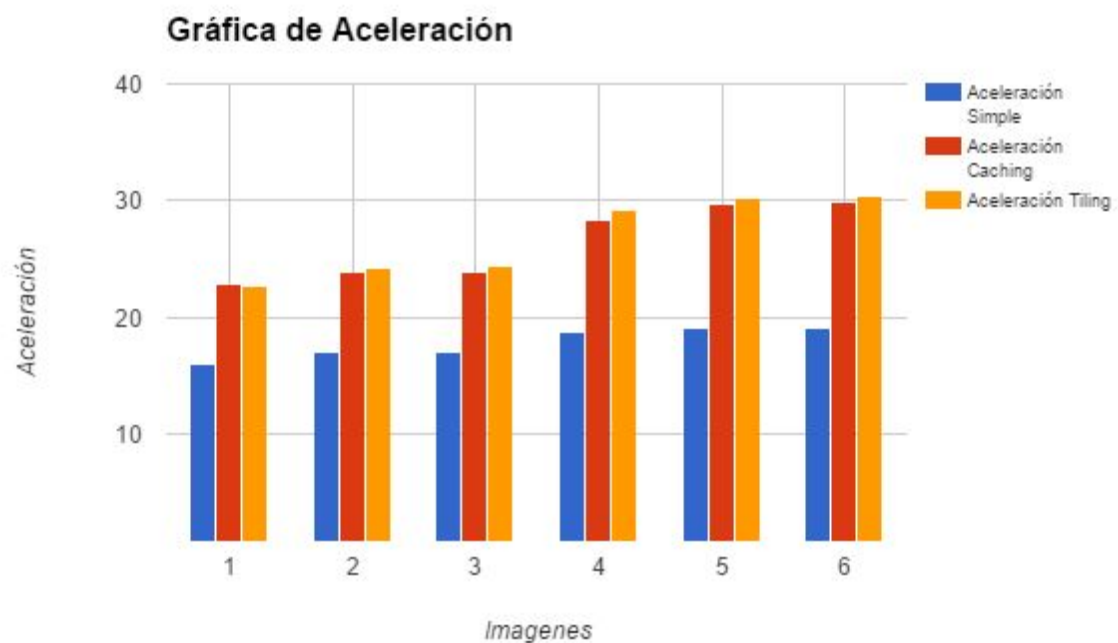
Aceleración	Aceleración Simple	Aceleración Caching	Aceleración Tiling
1	16.11520737	22.90085018	22.70261959
2	17.10887647	23.90028571	24.29831526
3	17.01742604	23.87346193	24.3682487
4	18.82001742	28.30836631	29.17882777
5	19.14576873	29.71646047	30.29941802
6	19.20563378	29.84409598	30.43318065

Gráficas resultantes:

Gráfica de tiempos promedios.



Gráfica de aceleración.

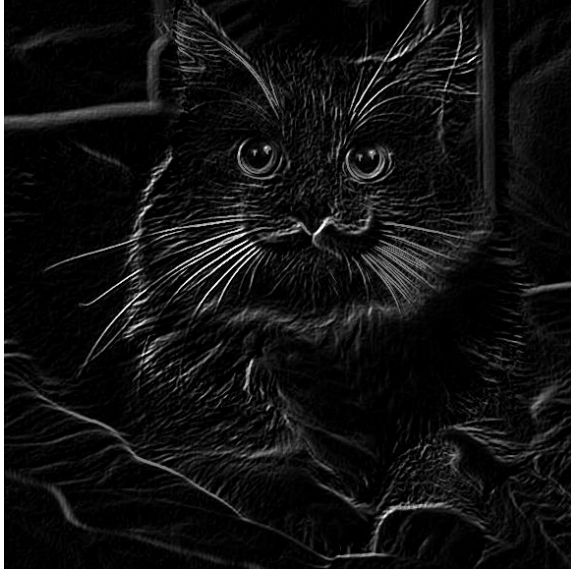


Resultados:

A continuación mostraremos los resultados de cada uno de los algoritmos, empezando por las imágenes resultantes con su respectivo tiempo en cada una de las versiones.

Imagen 1 (580 x 580)





Secuencial: 0.009966



Paralelo Simple: 0.00061845
Paralelo Caching: 0.0004352
Paralelo Tiling: 0.000439

Imagen 2 (638 x 640)





Secuencial: 0.012548



Paralelo Simple: 0.0007334
Paralelo Caching: 0.000525
Paralelo Tiling: 0.0005164

Imagen 3 (1366 x 768)

	
	
Secuencial: 0.029394	Paralelo Simple: 0.0017273 Paralelo Caching: 0.00123125 Paralelo Tiling: 0.00120625

Imagen 4 (2560 x 1600)

	
	
Secuencial: 0.110171	Paralelo Simple: 0.0058539 Paralelo Caching: 0.0038918 Paralelo Tiling: 0.0037757

Imagen 5 (5226 x 4222)






	
	
Secuencial: 0.604449	Paralelo Simple: 0.0315709 Paralelo Caching: 0.02034055 Paralelo Tiling: 0.0148941

Imagen 6 (4928 x 3264)

	
	
Secuencial:0.444501	Paralelo Simple: 0.0231443 Paralelo Caching: 0.0148941 Paralelo Tiling: 0.0146058

Conclusiones

1. En términos generales el algoritmo de convolución con memoria compartida presenta un mejor tiempo de ejecución en comparación con los otros algoritmos paralelos.
2. Al igual que en el tiempo de ejecución, el algoritmo de convolución con memoria compartida presenta mayor aceleración en comparación con los otros algoritmos paralelos.
3. El algoritmo de convolución simple presente un tiempo de ejecución más lento y una aceleración menor en comparación con el algoritmo con memoria constante y memoria compartida.
4. Al igual que otros trabajos, el algoritmo secuencial presenta un bajo rendimiento y poca aceleración en comparación con los algoritmos paralelos.
5. Al realizar algunas pruebas se puede concluir que el tipo de dato de la máscara afecta la forma en que el filtro se aplica en las imágenes.
6. Se recomienda usar el algoritmo de caching para las imágenes más pequeñas, ya que esta presenta un mejor rendimiento para este tipo de imágenes.
7. Al comparar las imágenes resultantes, se puede observar que las que se generaron con el algoritmo de convolución en paralelo presentan un mayor brillo en contraste con las generadas por el algoritmo en secuencial usado por OpenCV.