

PARCIAL 2: CONVOLUCIÓN EN DOS DIMENSIONES

**DANIEL FELIPE MARIN SANCHEZ
1089746672**

**UNIVERSIDAD TECNOLÓGICA DE PEREIRA
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
HIGH PERFORMANCE COMPUTING
PEREIRA 2015**

INTRODUCCIÓN

Este documento expone los resultados obtenidos en la aplicación de la 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.

La mascara usada para la convolución es una matriz de 3x3 [-1,0,1,-2,0,2,-1,0,1]. Igualmente para el procesamiento de las imágenes se esta usando la librería opencv, que permite cargar la imagen en escala de grises para posteriormente aplicarle la convolución.

Los algoritmos están implementados en CUDA y son ejecutados en la plataforma <http://judge.utp.edu.co:3000>, se pueden encontrar en <https://github.com/danielfms/hpc/tree/master/hpc/convolution2D> con los nombres convolution2DGlobal.cu, convolution2DConstant.cu y convolution2DShared.cu. De igual forma se encuentran las tablas y gráficas con todos los datos obtenidos en el mismo repositorio con el nombre de datos.ods .

RESULTADOS OBTENIDOS

Img1:580x580

Secuencial

Paralelo



Tiempo ejecución(seg):

Secuencial: 0,0030203

Paralelo Global: 0,000336 Aceleración: 8,989311

Paralelo Constant: 0,000254 Aceleración: 11,88061

Paralelo Shared: 0,0002493 Aceleración: 12,213438

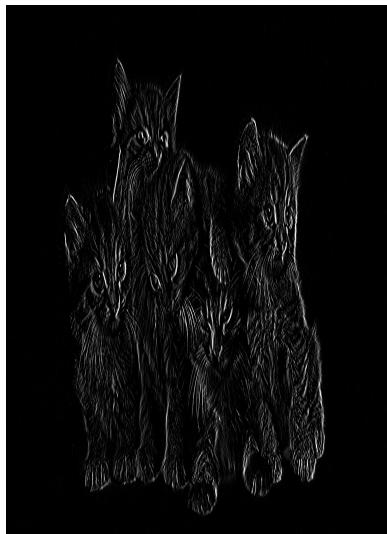
Img2:638x640



Tiempo ejecución(seg):

Secuencial: 0,0039651
Paralelo Global: 0,0003913 Aceleración: 10,141478
Paralelo Constant:0,0002913 Aceleración: 13,75795842
Paralelo Shared: 0,0002845 Aceleración: 13,781971

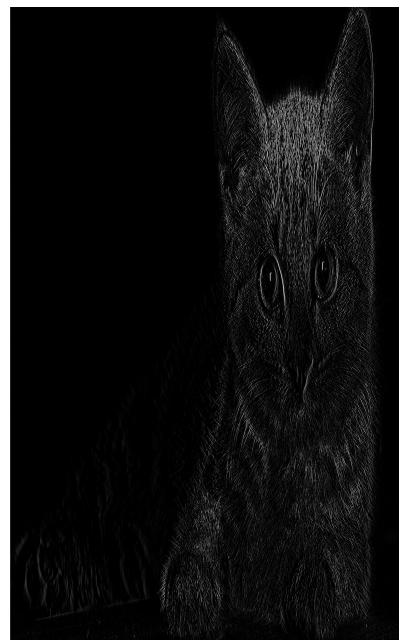
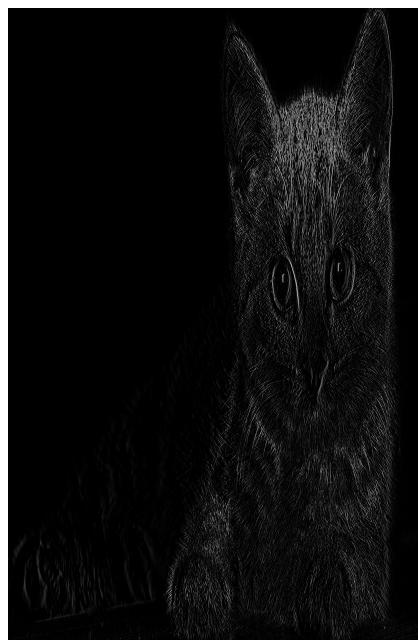
Img3:1336x768



Tiempo ejecución(seg):

Secuencial: 0,0089419
Paralelo Global: 0,000937 Aceleración: 9,54384
Paralelo Constant:0,0006783 Aceleración: 13,1528987
Paralelo Shared: 0,0006691 Aceleración: 13,327128

Img4:4928x3264



Tiempo ejecución(seg):

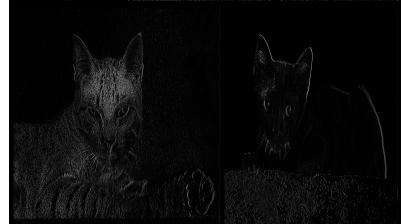
Secuencial: 0,0338858

Paralelo Global: 0,0031812 Aceleración: 10,651976

Paralelo Constant:0,0022484 Aceleración: 15,07672668

Paralelo Shared: 0,0021925 Aceleración: 15,494231

Img5:4928x3264



Tiempo ejecución(seg):

Secuencial: 0,1920223

Paralelo Global: 0,0168091 Aceleración: 11,424873

Paralelo Constant:0,0115179 Aceleración: 16,686933

Paralelo Shared: 0,0112296 Aceleración: 17,121888

Img6:5226x4222



Tiempo ejecución(seg):

Secuencial: 0,1413818

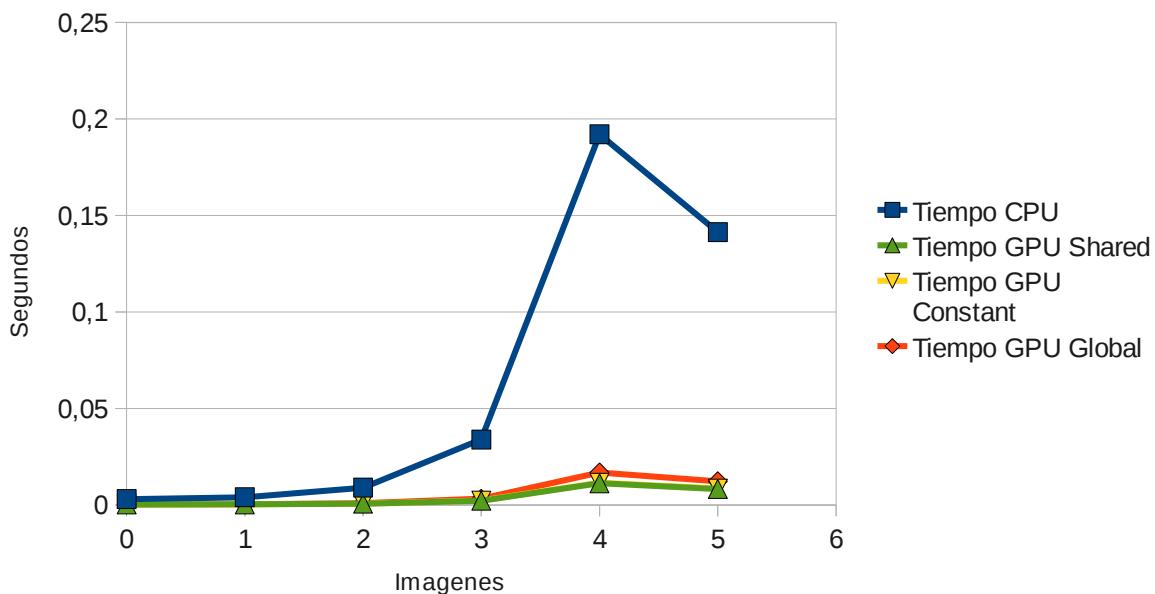
Paralelo Global: 0,0122281 Aceleración: 11,562228

Paralelo Constant:0,0084885 Aceleración: 16,61201675

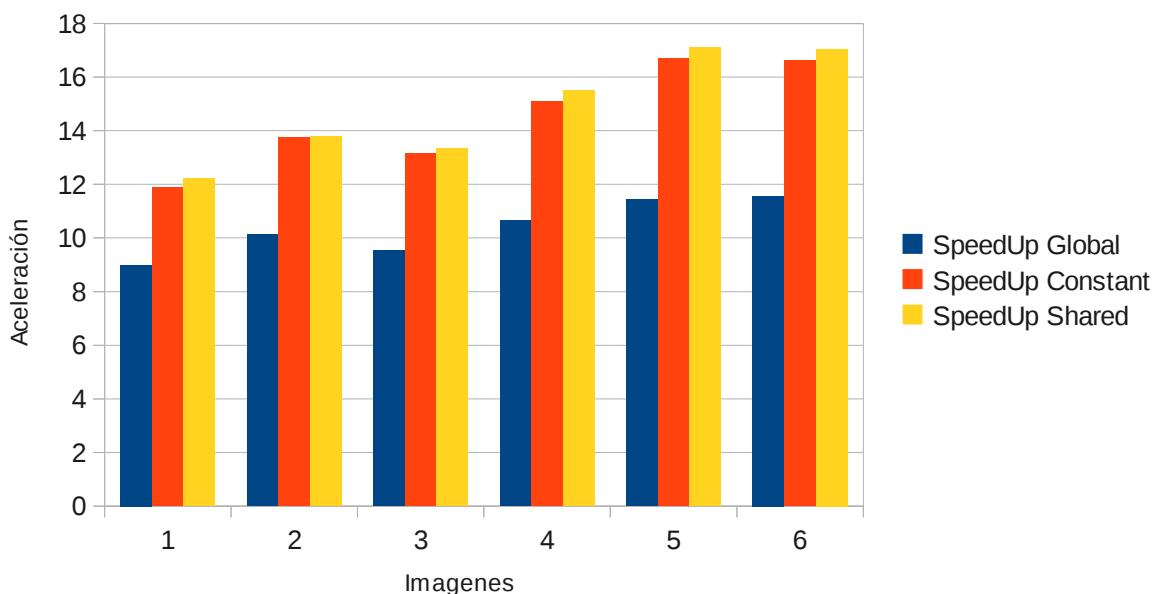
Paralelo Shared: 0,0082828 Aceleración: 17,050971

GRÁFICAS

Tiempos de ejecución



Aceleración Obtenida



CONCLUSIONES

- Los tiempos de ejecución siempre son mejores en memoria compartida con respecto a los otros tres algoritmos.
- En las gráficas se puede observar que se obtiene una mejor aceleración en el algoritmo con memoria compartida que en el algoritmo con memoria constante, sin embargo esta diferencia es mínima en los datos obtenidos.
- La implementación usando de memoria global no es una forma óptima de hacer paralelización.
- A pesar que la imagen 6 tiene mas resolución 5226x4222 que la imagen 5 4928x3264, el tiempo de ejecución de todos los algoritmos es mayor en la imagen 5 que en la imagen 6. Se puede observar un pico en las gráficas.
- En el algoritmo con memoria compartida se debe tener en cuenta los bordes de los tiles para que la mascara se aplique adecuadamente, de lo contrario la imagen resultado contendrá lineas blancas horizontales sobre la imagen esperada[1]. Esto se debe a que no se están procesando algunos datos necesarios que se encuentran en diferentes bloques.

[1] Imagen resultado sin tener el cuenta datos de otros bloques usando memoria compartida. En el repositorio se encuentra como shared.cu .

