Practica 2 -Implementación del efecto borroso de una ima-gen con GPU

Fernando Vargas, Omar Roa
Universidad Nacional Bogotá, Colombia
Email:fevargasmo@unal.edu.co, oeroaq@unal.edu.co

Figura 1. Escudo de la Universidad.

I. Antes de empezar

I-A. *Instalar CUDA 7.5*

Instalar CUDA 7.5 http://www.pradeepadiga.me/blog/2017/03/22/installing-cuda-toolkit-8-0-on-ubuntu-16-04/.
Este link explica para cuda 8.0, pero entonces descargamos cuda 7.5 de aca https://developer.nvidia.com/cuda-75-downloads-archive y simplemente cambiamos el archivo .deb

I-B. *OpenCV 3*

Para la instalación de OpenCV se puede seguir la guía
https://linuxhint.com/how-to-install-opencv-on-ubuntu/

I-C. *Como compilar el programa*

Comando de compilacion:

```
nvcc -I/usr/include -L/usr/local/lib
-g -o blur_CUDA blur_CUDA.cu
-lopencv_core -lopencv_imgproc
-lopencv_highgui -lopencv_imgcodecs
```

I-D. *Como ejecutar el programa*

Comando de ejecución:

```
./blur-effect imagenIn.jpg
imagenOut.jpg
 k t
```

-imagenIn: Seleccionan la imagen a tratar
-imagenOut: nombre de la imagen resultante, se crea.
-k: Cantidad de kernel
-t: Numero de hilos

II. Introducción

II-A. *Efecto borroso de una imagen en CUDA*

En este documento se vera la implementación del efecto borroso en una imagen utilizando CUDA. Para el tratado de la imagen, se usara la librería OpenCV. Se diseñó y realizó un programa con unos metodos para aplicar el efecto borroso en una imagen dada. Este programa trabaja con kernel en forma de çruz".

II-B. *PARALELIZACIÓN DEL ALGORITMO.*

El método de paralelización utilizado en este programa es Block Wise en 2 dimensiones.

Si por ejemplo se utilizan 4 hilos para aplicar el efecto borroso en una imagen, la imagen se divide como en la imagen anterior, es decir, a cada hilo le corresponde una sección horizontal, partida equitativamente entre el numero de hilos.

III. EXPERIMENTOS Y RESULTADOS.

III-A. *Dispositivo GPU*

Se utilizo la tarjeta nvidia GeForce 610m.

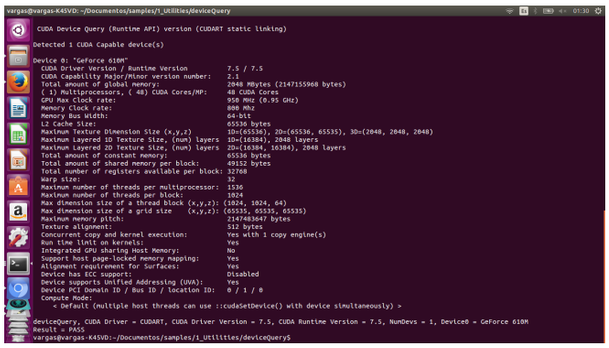

Figura 2. GeForce 610m.

III-B. *Gráficas Secuencial*

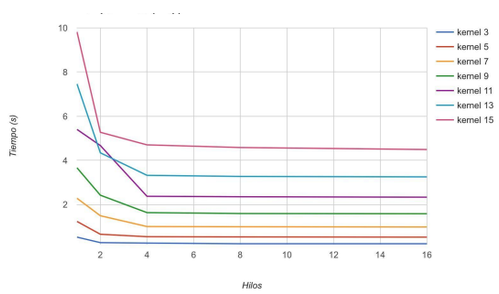

Figura 3. Tiempo secuencial en imagen de 720p.

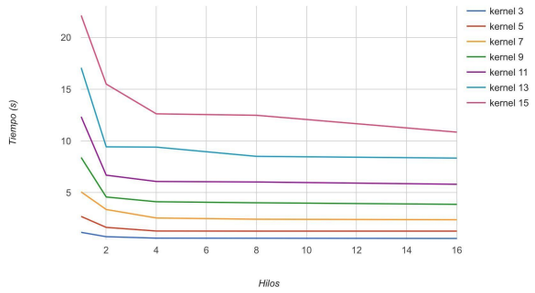

Figura 4. Tiempo secuencial en imagen de 1080p.

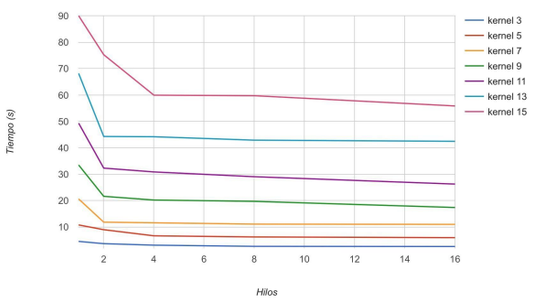

Figura 5. Tiempo secuencial en imagen de 4K.

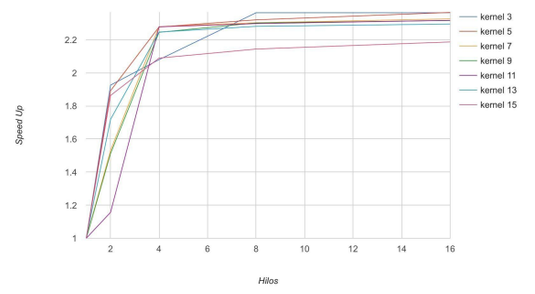

Figura 6. SpeedUp secuencial en imagen de 720p.

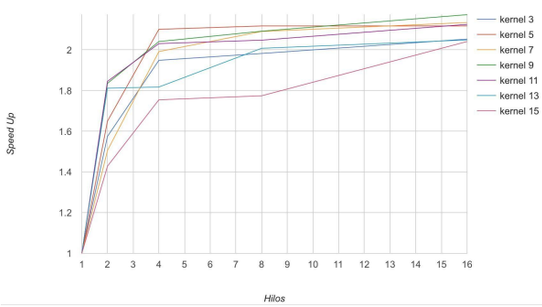

Figura 7. SpeedUp secuencial en imagen de 1080p.

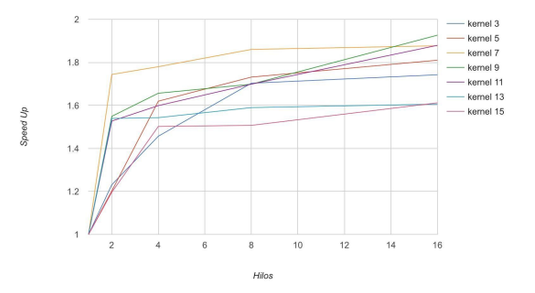

Figura 8. SpeedUp secuencial en imagen de 4K.

III-C. Gráficas con CUDA

Para esta prueba se realizaron las pruebas usando CUDA.

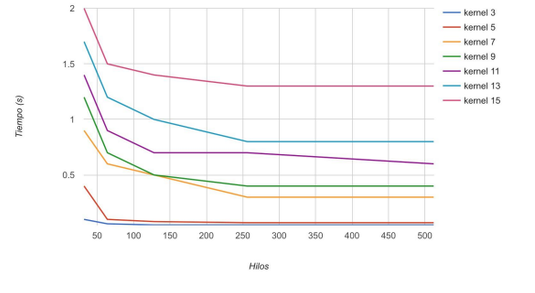

Figura 9. Tiempo con CUDA en imagen de 720p.

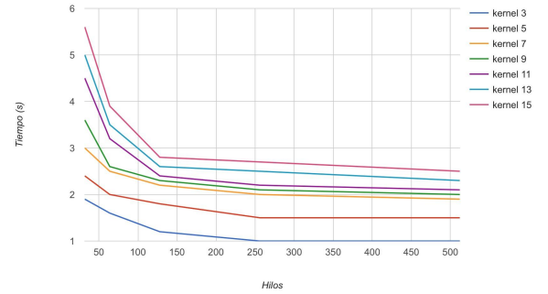

Figura 10. Tiempo con CUDA en imagen de 1080p.

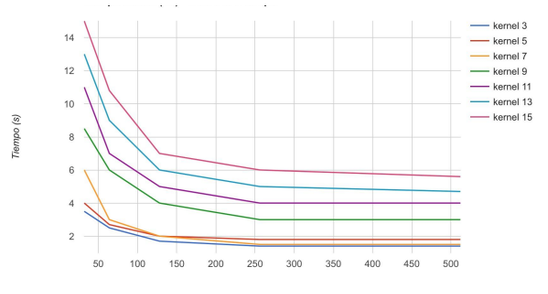

Figura 11. Tiempo con CUDA en imagen de 4K.

III-D. Graficas SpeedUp CUDA

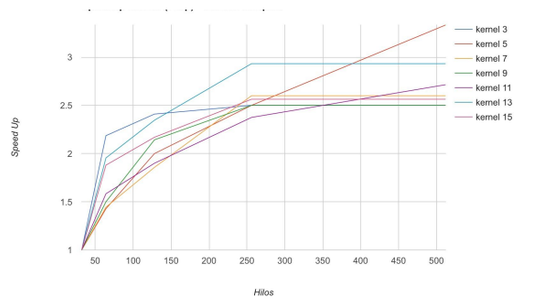

Figura 12. SpeedUp con CUDA en imagen de 720p.

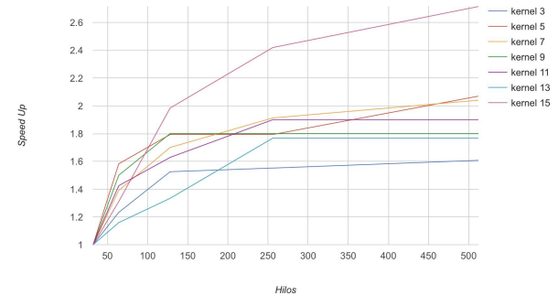

Figura 13. SpeedUp con CUDA en imagen de 1080p.

IV. Conclusiones

Con estos resultados podemos concluir el impacto en el tiempo de ejecución al paralelizar un algoritmo o programa. Hay que tener en cuenta que al usar mas

hilos mas comunicaciónn debe haber entre ellos y esto resulta en más tiempo.
Pero ya que el costo de crear hilos en la gpu con la ayuda de CUDA es muy bajo, es buena idea crear varios hilos para realizar la tarea.
En CUDA es necesario conocer el dispositivo gpu, ya que se debe configurar para utilizarlo al máximo.
La mejora del tiempo de un programa al paralelizarlo es limitado, en las gráficas podemos ver que la mejora ya no es notable al aumentar mucho la cantidad de hilos. Para verificar esto, el SpeedUp también nos da un resultado en el que su aumento, al ser mayor los hilos usados, no es notable.