Practica I Implementación del efecto borroso de una imagen

Heyner Martinez, Yesid Ochoa
Universidad Nacional Bogotá, Colombia
Email:hsmartineza@unal.edu.co, yaochoal@unal.edu.co

Figura 1: Escudo de la Universidad.

***Resumen*—En este documento se explica la implementación del algoritmo gausseano usado para el desenfoque de imágenes, que aplica un efecto de suavizado para mapas de bits, se encuentra en muchos software de edición gráfica. Se presenta el algoritmo en el lenguaje de programación C, para esto se corre el algoritmos de forma secuencial, a través de hilos POSIX (2,4,8 y 16 hilos) y a través de la librería Openmp, para así hacer un análisis del decremento del tiempo con los diferentes métodos.**

I. Introducción.

El algoritmo y tipo de filtrado de imágenes usado es a través de una matriz de convolución. Es el tratamiento de una matriz por otra que se llama kernel. El filtro de matriz de convolución usa la primera matriz que es la imagen que se va a tratar. La imagen es una colección bidimensional de píxeles en coordenada rectangular. El kernel usado depende del efecto deseado.
El filtro examina, uno a uno cada píxel de la imagen. Para cada uno de ellos que se llama píxel inicial, se multiplica el valor de este píxel y los valores de los 8 circundantes por el valor correspondiente del kernel. Entonces se añade el resultado, y el píxel inicial se regula en este valor resultante final. Se debe tener en cuenta el tamaño de la matriz kernel debe ser impar, ya que se requiere de una celda central que sera el píxel inicial, para este caso se uso un kernel de 15x15.
Debido a que la imagen se encuentra representada por una matriz de píxeles, se puede medir el tiempo en el que tarda una matriz ser procesada de forma completa, ejecutando el algoritmo de forma secuencial y con la paralelización del algoritmo ya sea con hilos POSIX o con la librería Openmp. Y de esta manera analizar las diferencias y comportamientos.

I-A. Librerías

Para el manejo de imágenes se empleo la librería OpenCV para C/C++, la cual nos permitió cargar una imagen representada como una matriz de enteros, en tres grupos que corresponden a los canales RGB, para poder trabar con ella y mostrar el resultado del filtro en una nueva imagen.

II. Paralelización del algoritmo.

El método utilizado para la paralelización del programa es Block-Wise en dos dimensiones por columnas.

Figura 2: Particionamiento de la imagen.

Si por ejemplo se utilizan 4 hilos para aplicar el efecto de filtrado de imagen, la imagen se divide en cuatro parte iguales como la imagen anterior, es decir que a cada hilo le corresponde una sección vertical para hacer el procesamiento de la imagen.

III. Experimentos y resultados.

Para realizar las pruebas se midió el tiempo de ejecución del programa para diferentes tamaños de imagen (720p, 10080p y 4k) y para diferente numero de hilos.

III-A. Secuencial

Tiempo (s)	Tamaño
1.967	720p
4.448	1080p
4.448	4k

Cuadro I: Ejecución forma secuencial.

III-B. Paralelización Imagen 720p

Ejecución para una imagen de 720p en POSIX.

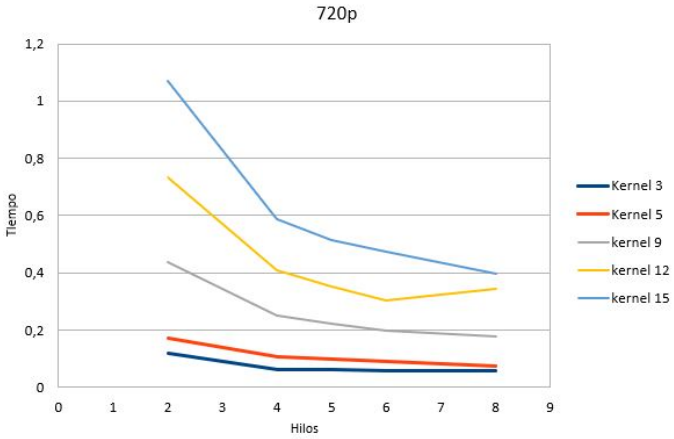

Figura 3: Gráfica imagen 720p POSIX.

Ejecución para una imagen de 720p en Openmp.

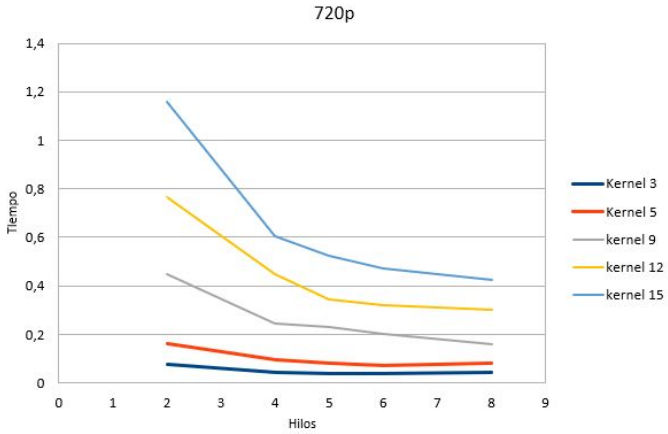

Figura 4: Gráfica imagen 720p Openmp.

III-C. Paralelización Imagen 1080p

Ejecución para una imagen de 1080p en POSIX.

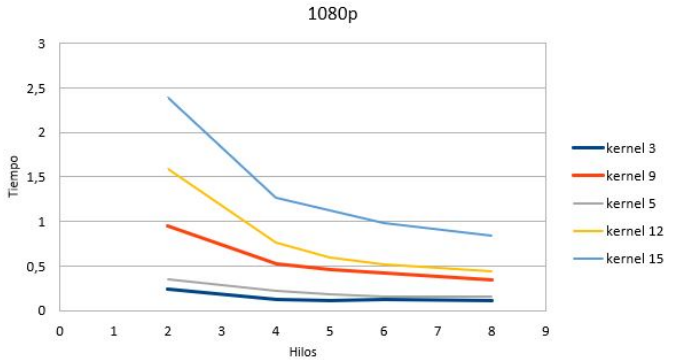

Figura 5: Gráfica imagen 1080p POSIX.

Ejecución para una imagen de 1080p en Openmp.

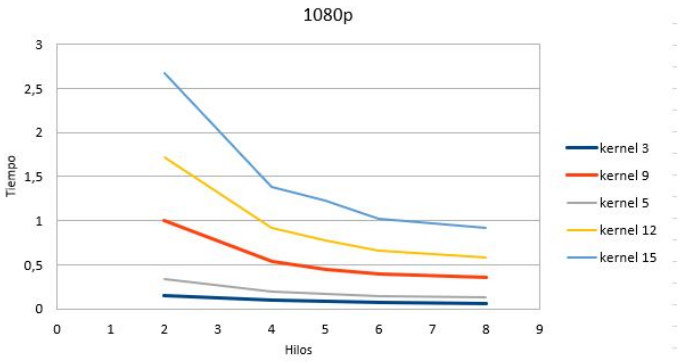

Figura 6: Gráfica imagen 1080p Openmp.

III-D. Paralelización Imagen 4k

Ejecución para una imagen de 4k en POSIX.

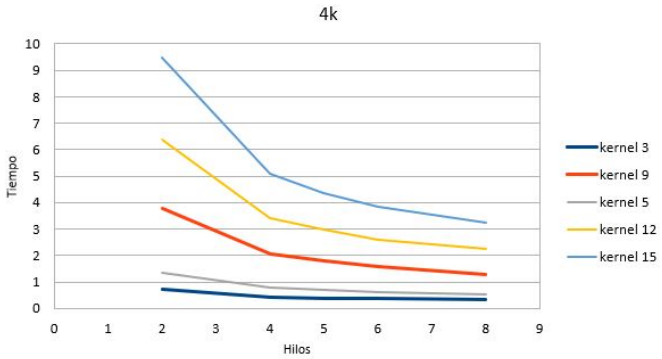

Figura 7: Gráfica imagen 4k POSIX.

Ejecución para una imagen de 4k en Openmp.

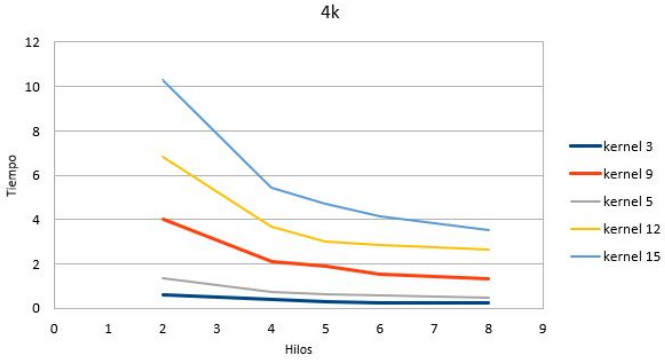

Figura 8: Gráfica imagen 4k Openmp.

IV. CUDA.

Para la ejecución del programa se uso la tarjeta gráfica Nvidia 970 Gtx y se uso diferente cantidad de hilos y distintos kernel, con sus respectivo tamaños de imagen.

IV-A. Paralelización Imagen 720p

Ejecución para una imagen de 720p en POSIX.

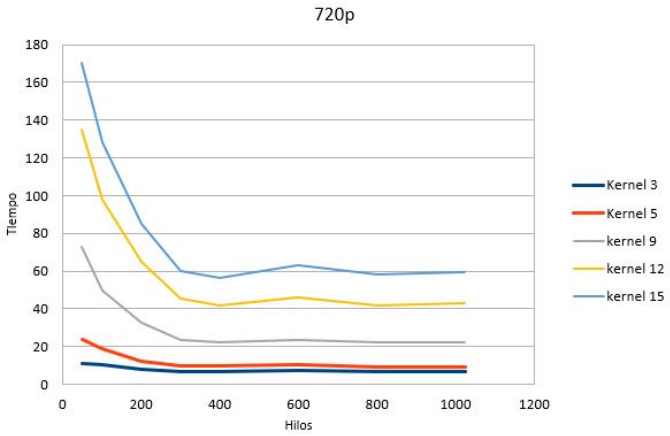

Figura 9: Gráfica imagen 720p CUDA.

IV-B. Paralelización Imagen 1080p

Ejecución para una imagen de 1080p.

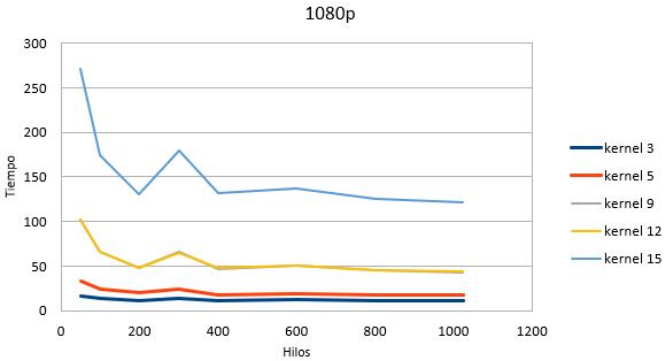

Figura 10: Gráfica imagen 1080p CUDA.

IV-C. Paralelización Imagen 4k

Ejecución para una imagen de 4k.

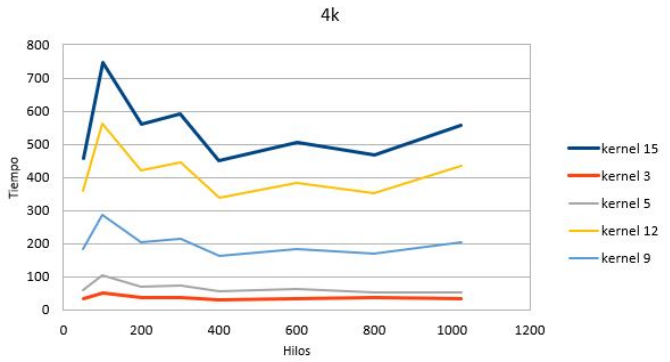

Figura 11: Gráfica imagen 4k CUDA.

CONCLUSIONES.

Con estos resultados se puede concluir el impacto en el tiempo de ejecución al paralelizar un programa, cuando se requiere realizar un gran número elevado de operaciones. Se debe tener en cuenta que no siempre cualquier programa es paralelizable y escoger la forma correcta en que se paraleliza, también cabe recalcar que entre más hilos mayor comunicación entre ellos y no siempre significa mayor rapidez, como se ve en los resultados que llegara el programa a un punto en que el tiempo de ejecución se estabilizara por mas hilos que se ejecuten. Para verificar esto también el SpeedUp nos da un resultado que al aumentar el número de hilos no se notara la diferencia.

De acuerdo a las gráficas, a partir de los 8 hilos el tiempo que tarda en ejecutar el programa no va disminuir significativamente.

REFERENCIAS.

- https://es.wikipedia.org/wiki/Desenfoque_gaussiano
- https://opencv.org/
- https://docs.gimp.org/es/plug-in-convmatrix.html