Implementación del Efecto Borroso de una Imagen con Método Secuencial y Multihilo POSIX

Hugo Alejandro Arias Zamora
e-mail: huaariasza@unal.edu.co
Diego Andrés Baquero Tibocha
e-mail: dabaquerot@unal.edu.co

***Resumen*—En este documento se presentan dos formas para la implementación del algoritmo que permite el efecto borroso de una imagen, conocido en inglés como *Gaussian blur* o *Gaussian smoothing*; de forma secuencial con uso de OpenCV y haciendo uso de paralelización a través de múltiples hilos. Por otro lado, se presentan las los tiempos de respuesta por cada uno de los métodos haciendo uso de punteros y memoria dinámica.**

***Índice de términos*—Parallel processing, Multiprocessing systems, Multithreading, Parallel algorithms.**

I. Introducción

ESTE documento detalla la implementación del algoritmo *Gaussian Blur* o efecto de difuminado de manera secuencial y usando múltiples hilos. Sin embargo, los tiempos de procesamiento que se tarda cada uno de los métodos con los diferentes tamaños de imágenes; son lo que realmente se va a analizar de forma detallada a lo largo de este documento. Adicionalmente, se muestra un análisis de cómo el kernel y la cantidad de hilos (en el método multihilo) pueden afectar a los tiempos de ejecución de la función. Finalmente, el término de kernel se explicará con más detalle en el documento.

II. Implementación del algoritmo gaussian blur

La implementación del algoritmo Gaussian Blur se llevó a cabo en el lenguaje de programación C++ con el uso de hilos de POSIX (Portable Operating System Interface en sistemas Unix) con el uso de la biblioteca *pthead* para la manipulación de hilos y la implementación de conceptos básicos de multihilos. Lo anterior, con el objetivo de lograr procesamiento multihilo en el procesador que ejecuta la aplicación. Adicionalmente, se hizo uso de OpenCV para la manipulación de imágenes.

A. Blur Effect

El algoritmo *Gaussian Blur*, también conocido como *Gaussian smoothing* es el resultado de aplicar la función gaussiana con el objetivo de realizar un desenfoque a la imagen objetivo. Por lo tanto, el efecto de *blur* es ampliamente utilizado en el software de gráficos, con el objetivo de reducir el ruido de la imagen y reducir los detalles, basada en una borrosidad suave [1].

Por lo tanto, el algoritmo se encarga de mezclar de manera ligera los colores de los píxeles que estén vecinos al pixel que está siendo procesado, generando que la imagen pierda algunos detalles mínimos y que luzca más suave, pero más borrosa respecto a la imagen original.

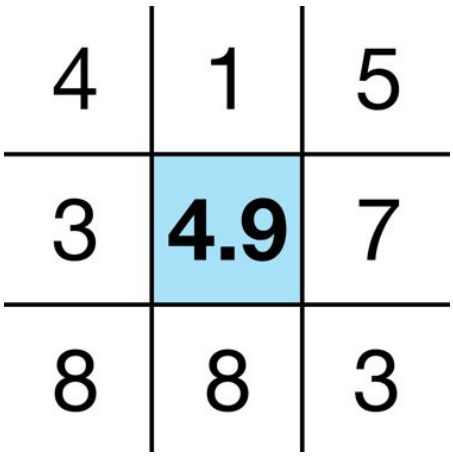

Fig. 1 Cálculo de Blur Effect. Imagen tomada de: https://koenig-media.raywenderlich.com/uploads/2014/09/blur-how2.png

En la anterior imagen se puede observar cómo sería el cálculo que realiza el algoritmo con un kernel igual a tres. Suponiendo que la casilla que se va a procesar, representada en color azul, cuenta con un valor inicial de 5, el nuevo valor se calcularía del promedio de sus vecinos. Por lo tanto, el cálculo sería el siguiente: (4 +1 + 5 + 3 + 7 + 8 + 8 + 3 + **5**) / 9 = 4,9 tomando sólo un decimal de precisión. Por otro lado, el kernel corresponde a la raíz cuadrada de la matriz de vecinos, por consiguiente, corresponde a la raíz cuadrada de nueve, para el ejemplo anterior.

B. *Secuencial*

Se utilizó esta técnica inicialmente con el objetivo de realizar la implementación de la función *blur-effect* a diferentes tamaños de imágenes. Con la ayuda de OpenCV se obtienen los valores de cada uno de los pixeles. De esta manera, a la función secuencial sólo es necesario pasarle como parámetro el tamaño del kernel y se encarga de realizar el cálculo para cada uno sus pixeles dependiendo del valor de sus pixeles vecinos y el tamaño del kernel.

C. *Hilos de POSIX*

Por un lado, POSIX permite la creación de aplicaciones confiables, y como su nombre lo indica, portables, de forma un poco más sencilla. Por otro lado, para la implementación multihilo de la aplicación se hizo uso de la biblioteca *pthread* para la manipulación de hilos. Por consiguiente, la manipulación de hilos permite lograr un procesamiento con más de un hilo.

Inicialmente, a la implementación multihilo es necesario pasarle como parámetro la imagen a la cual se le va a aplicar la función de suavizado, el kernel y la cantidad de hilos en los que se va a realizar la tarea. Posteriormente, con la herramienta OpenCV se divide la imagen original en la cantidad de hilos que se van a usar. Luego, cada uno de los hilos va a ejecutar la función de *blur-effect* con el valor del kernel definido por el usuario.

Finalmente, se unen las imágenes a las que se les realizó el proceso, obteniendo la imagen con el tamaño original, pero con el suavizado aplicado. En cuanto mayor sea el kernel, mayor será el suavizado y la imagen se verá más distorsionada.

D. *Método de Particionamiento y Balanceo de Carga*

Mediante el método secuencial no se realiza ningún particionamiento sobre la imagen.

Por otro lado, en el método multihilo implementado, se realiza la división de la imagen en la cantidad de hilos que se desean lanzar. Adicionalmente, el balanceo de carga, está sujeto a la cantidad de pixeles que va procesar cada uno de los hilos, es decir, si la imagen tiene una longitud que es divisible por la cantidad de hilos que se van a lanzar, con un módulo igual a cero, cada uno de los hilos va a tener la misma carga, de lo contrario, puede ocurrir que un hilo procese una imagen con menor tamaño, por lo tanto este hilo tardará menos tiempo, asumiendo que la carga adicional del procesador es nula.

E. *Experimento*

Inicialmente, se tomaron tiempos de respuesta, haciendo uso de la librería de C, llamada *ctime* para los diferentes tamaños de imágen, los cuales fueron 780p, 1080p y 4k, con diferentes tamaños de kernel, los cuales fueron 3, 5, 13 y 15; para cada uno de los procedimientos. Adicionalmente, se tomaron los tiempo antes y después de realizar los recortes de cada una de las imágenes.

A continuación, se muestra la imagen a la cual se le aplicó la función de Blur con la imagen de 720p:

Fig. 2 Efecto Blur para la imagen 720p

F. *Resultados*

Inicialmente para una imagen con resolución 1280x720 (720p) se realizó la medición y se obtuvieron los siguientes resultados:

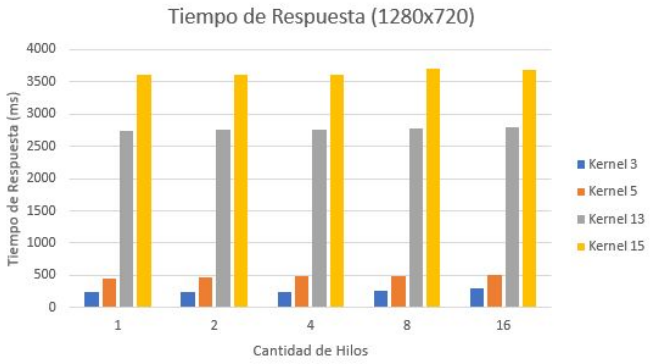

Fig. 3 Tiempos de respuesta imagen 720p

Como se observa en la anterior imagen los valores no cambian de forma dramática. En la siguiente imagen se muestra la tabla que contiene el promedio de tres muestras para cada esta imagen.

Hilos	Kernel	Tiempo (ms)	Speedup
1	3	234,808	1
1	5	444,235	1
1	13	2732,33	1
1	15	3606,78	1
2	3	241,812	1,03
2	5	475,708	1,071
2	13	2755,28	1,008
2	15	3615,32	1,002
4	3	244,774	1,042
4	5	480,707	1,082
4	13	2760,56	1,01
4	15	3620,31	1,004
8	3	261,972	1,116
8	5	488,007	1,099
8	13	2782,64	1,018
8	15	3703,05	1,027
16	3	295,467	1,258
16	5	501,271	1,128
16	13	2798,85	1,024
16	15	3692,75	1,024

Tabla. 1 Promedio de tiempos y Speedup para imagen de 720p

Adicionalmente, se hizo la gráfica para el Speedup, el cual se calculó con la fórmula $O(n)/O(1)$, para cada uno de los kernel e hilos para dicho tamaño de imagen.

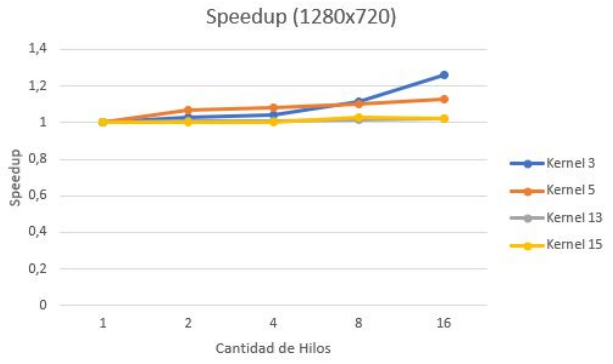

Fig. 4 Speedup imagen 720p

Por otro lado, se tomaron estas mismas mediciones posterior a la partición de la imagen, con el fin de detallar si esta tarea puede generar cambios en los datos obtenidos.

En la siguiente imagen se muestran los tiempos de respuesta y su Speedup sólo para la función Blur:

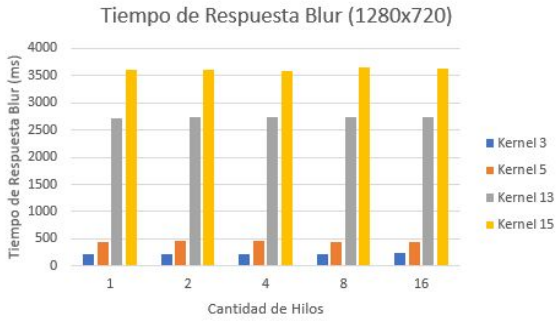

Fig. 5 Tiempos de respuesta de la función Blur en la imagen 720p

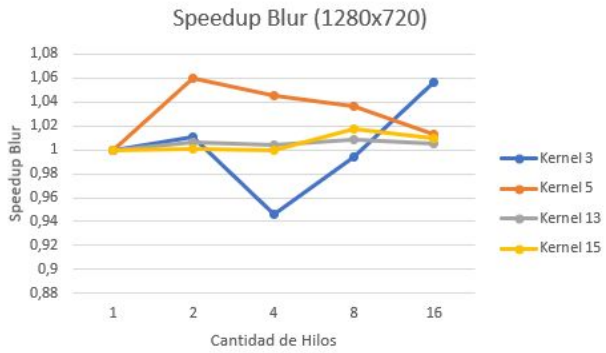

Fig. 4 Speedup de la función Blur en la imagen 720p

Bien, para la imagen de 1080p se realizó el mismo procedimiento. En las Fig. 6 y 7 podrá detallar el tiempo de respuesta para dicha imagen.

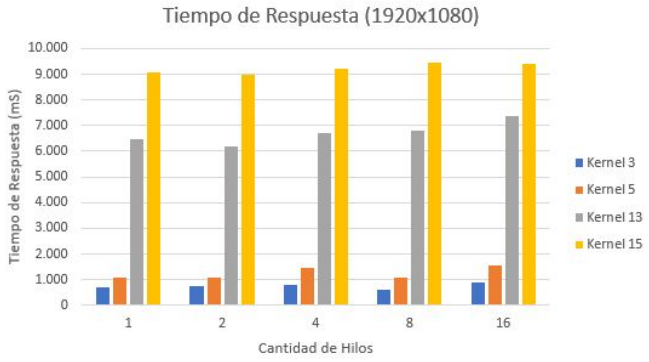

Fig. 6 Tiempos de respuesta imagen 1080p.

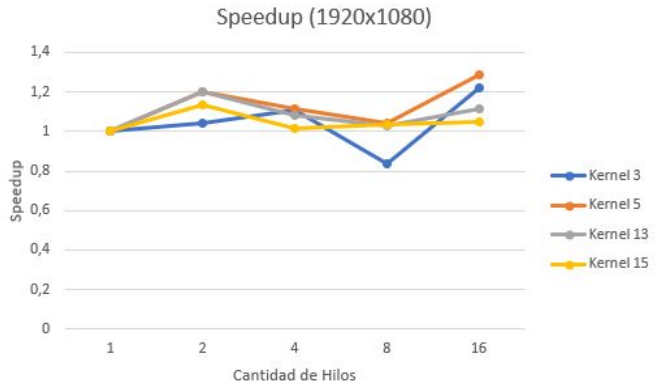

Fig. 7 Speedup imagen 1080p

Adicionalmente, también se tomaron los tiempos de ejecución sólo para la función de Blur, como se detalla en las las Fig. 8 y 9.

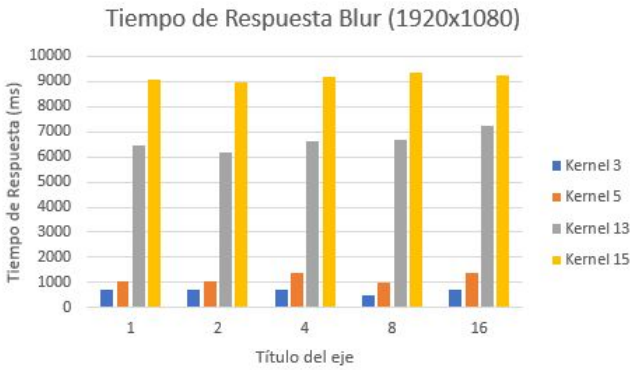

Fig. 8 Tiempos de respuesta de la función Blur en la imagen 1080p.

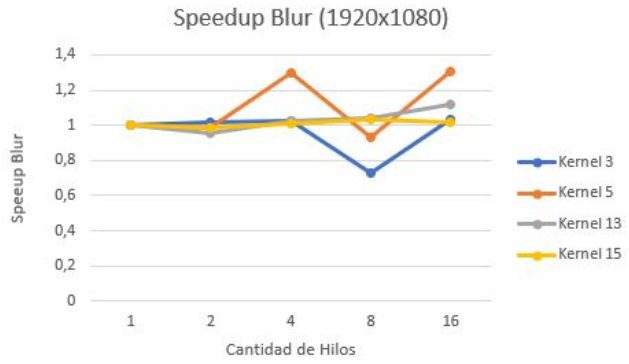

Fig. 9 Speedup de la función Blur en la imagen 1080p

Finalmente, se realiza el procedimiento para la imagen con resolución 4K. En el siguiente gráfico se puede observar el tiempo de respuesta para dicha imagen:

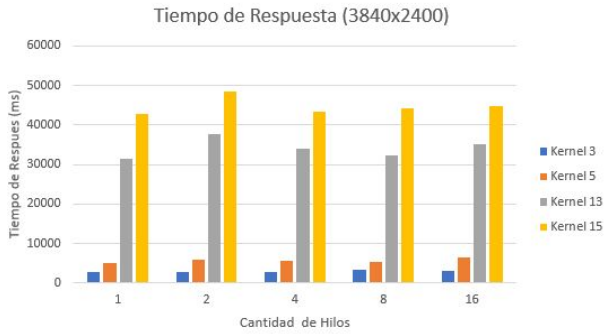

Fig. 10 Tiempos de respuesta imagen 4K.

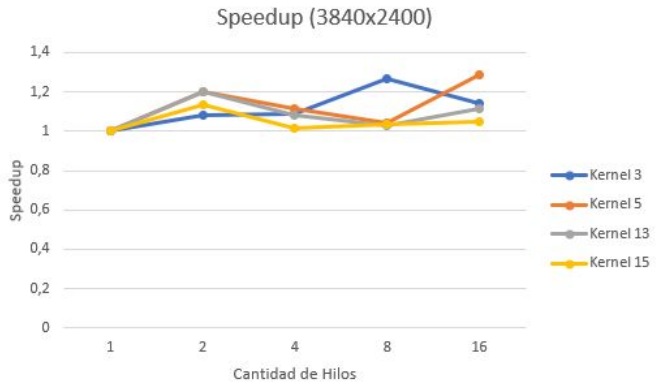

Fig. 11 Speedup imagen 4K.

Finalmente, similar a las anteriores se calculó el tiempo de respuesta y el Speedup sólo de la función de Blur.

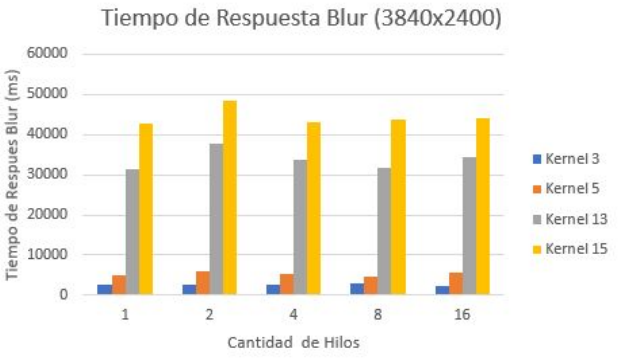

Fig. 12 Tiempos de respuesta de la función Blur en la imagen 4K.

Fig. 9 Speedup de la función Blur en la imagen 1080p

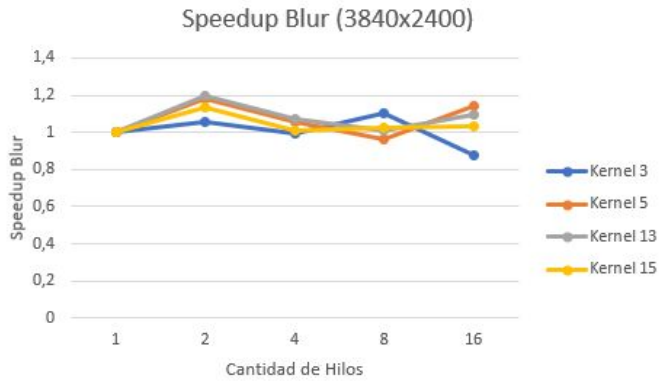

Fig. 13 Speedup de la función Blur en la imagen 4K

Las anteriores gráficas reflejan los resultados obtenidos para los tiempos de respuesta y el speedup promedio en tres iteraciones para cada uno de las imágenes. Por otro lado, se tomaron kernel de 3, 5, 13 y 15 para observar el comportamiento en los puntos extremos.

III. Conclusiones

El tiempo de particionado de la imágen para la implementación de hilos afecta el beneficio obtenido gracias a estos por lo tanto solo se puede observar un beneficio real de estos en casos en los que el kernel tenga un valor alto (mayor a 25).

En cada partición hay que tener en cuenta la implementación del algoritmo de blur en los bordes, ya que si simplemente se tiene en cuenta como un bloque normal de menor tamaño que el kernel la imagen se va a cargar hacia un lado haciendo que al integrar los bloques se observen líneas de unión afectando el resultado.

Se debe tener una estructura con el identificador de cada bloque al realizar la implementación con hilos ya que cada uno de estos va a entregar sus resultados en desorden, otra solución hubiera sido la implementación de un semáforo.

El algoritmo de blur promedia cada píxel con sus píxeles adyacentes de acuerdo a un tamaño de kernel definido, por lo tanto entre mayor sea el kernel más borrosa se va a ver la imágen pero a su vez se va a requerir un tiempo de procesamiento mayor.

Los tiempos que requiere el sistema para la presentación de la imágen en la interfaz de usuario afectan el análisis de rendimiento, por lo tanto no se tuvieron en cuenta, para la generación de los resultados.

IV. Referencias

[1] Wikipedia, “Gaussian blur”, [online] *Wikipedia*, 2017. Disponible en: https://en.wikipedia.org/wiki/Gaussian_blur.