Práctica 1 - Implementación del efecto borroso de una imagen

Lizzy Tengana Hurtado, Nicolás Restrepo Torres
Universidad Nacional Bogotá, Colombia

***Resumen*—El siguiente documento busca presentar la implementación hecha del algoritmo BoxBlur para desenfocar imágenes implementado de forma secuencial, con hilos POSIX y finalmente con la librería OpenMP. Esto con el fin de mostrar los efectos que tiene la paralelización en el procesamiento de las imágenes y poder comparar el tiempo de ejecución obtenido en el algoritmo paralelizado con diferente número de hilos y comprobar los beneficios que se tienen al usar la GPU.**

I. Librerías utilizadas

Para el manejo de imágenes se empleo la librería OpenCV para C/C++ la cual permitió cargar una imagen representada como una matriz de enteros (grupo de 3 enteros) correspondientes a los canales RGB y modificarla de acuerdo a los efectos que se plantearon en el taller.

II. Procedimiento

Box Blur consiste en un kernel de tamaño $n \times n$ que no es más que una matriz que circunda a uno de los píxeles en un momento dado. Este píxel se pone con en el centro con respecto a los demás pixeles que consisten en la matriz en uno de los cuatro puntos centrales o bien en el centro exacto del kernel. Esto ocurre en uno de dos casos en que el tamaño del kernel sea un número impar o un par, para un tamaño impar de kernel será siempre un único punto y en el caso de que sea par serán cuatro puntos de los cuales es indiferente escoger cualquiera de estos.

Se separan las componentes RGB que es el formato predeterminado que maneja la librería OpenCV en su valor de 0 a 255 para R, G Y B y se hace un procedimiento análogo para cada una de ellas. Se calcula un promedio simple con los pixeles que encierran al kernel descrito anteriormente y cada componente que contenga al kernel. Se verifica que el pixel es uno válido para determinar si entra en el promedio, esto es, que el tamaño del kernel en los bordes puede desbordarse e igual la imagen no va a perder ninguna fila ni ninguna columna porque se calcula el promedio con base a los pixeles que puedan ser encerrados por el kernel.

Finalmente se reemplaza el número de cada una de las componentes con el número promedio que haya sido retornado. Posteriormente se carga la información de la imagen un archivo mediante la estructura de datos Mat que genera OpenCV para notar el resultado de borrosidad. Esta librería no es más que una matriz especial de tres dimensiones con cada una de las componentes y capaz de representar una imagen en un formato usual como JPEG, PNG y demás, dados los valores para cada componente de la matriz.

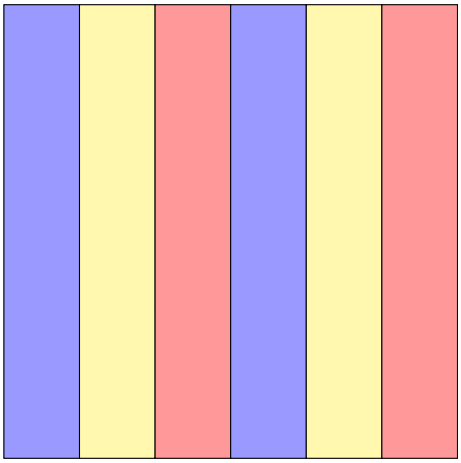

Figura 1: Particionamiento del problema (block-wise)

III. Balanceo de carga

El procesamiento de cualquier imagen bajo este esquema está dado por el número de pixeles precisamente porque el número de promedios a calcular. Se ejecuta entonces una columna de pixeles determinada para cada hilo, el método usado aquí sirve igualmente en caso de que se quisieran ejecutar filas para cada uno de hilos. El balanceo de carga entonces fue realizado por clases de equivalencia, esto quiere que para n hilos, donde cada hilo está identificado únicamente con un id incremental desde 0, se tiene que cada hilo procesa la columna $n+(c\times x)$ del proceso que calcula el promedio de los pixeles, donde c es el número de hilos, n es el id incremental y x es un escalar que determina cuál columna se ejecutará en la siguiente iteración del hilo dado. Esto quiere decir que para el hilo 0 entonces la clase de equivalencia se refiere a la clase de equivalencia del 0 que va a procesar las columnas que satisfagan $i + \sum_{i=i}^{x} c$, donde $0 \leq x \leq w$, siendo w el tamaño de columnas de la imagen. Análogamente se haría el mismo proceso para las filas limitado por el número de filas. Una forma gráfica de verlo es la siguiente, el hilo 0 representado por el color azul, hilo 1 representado por el color amarillo y hilo 2 representado por el color rojo. Y se ejecutan los pixeles de esa columna tal cual sea el hilo que corresponda (fig.1).

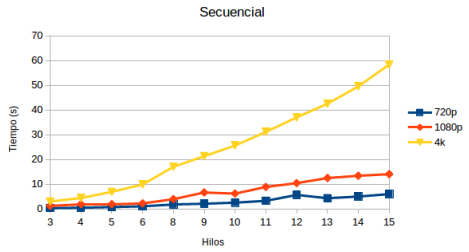

Figura 2: Ejecución programa secuencial.

IV. Detalles y pruebas

Se debe manejar una sección crítica de escritura, pero solo exclusivamente para la escritura debido a que el objeto Mat objetivo en dónde resultara la imagen con el efecto borroso aplicado ha sido clonado del objeto Mat original y reinicilizado con todos sus pixeles en blanco. Esto se hace con el fin de separar las operaciones de lectura y escritura en cada uno de los pixeles por cada uno de los hilos. La sección crítica de escritura puede ser entonces exclusivamente el acceso al pixel para sobreescribirlo en el nuevo objeto Mat.
Todas las pruebas realizadas para este ejercicio son para tres imagenes:

1. 720p $\Rightarrow$ 720 filas y 1280 columnas
2. 1080p $\Rightarrow$ 1080 filas y 1920 columnas
3. 4K $\Rightarrow$ 2160 filas y 3840 columnas

V. Programa secuencial.

Para el programa secuencial se puede variar solo el tamaño del kernel y el script ejecuta las tres imágenes con tamaños de kernel $3 \leq k \leq 15$ llevando a los resultados de la figura 1.

Se puede notar una progresión bastante clara en el tiempo del efecto borroso en términos del tamaño de la imagen y en términos del tamaño del kernel.

VI. Programa paralelizado.

Para la implementación del algoritmo se usaron dos funciones principales: BoxBlur y CheckBounds en la cual se aplica el BoxBlur. Se hacen pruebas para cada una de las imágenes y tamaños variables de kernel $3 \leq k \leq 15$ tomando el tiempo en segundos para ver como van bajando los tiempos, a veces, si la imagen es muy pequeña o el algoritmo no es tan complejo puede ser que los tiempos no reflejen una mejoría lineal. Los resultados son los siguientes

VI-A. Hilos POSIX

Figuras 2, 3 y 4

VI-B. OpenMP

Figuras 5, 6 y 7

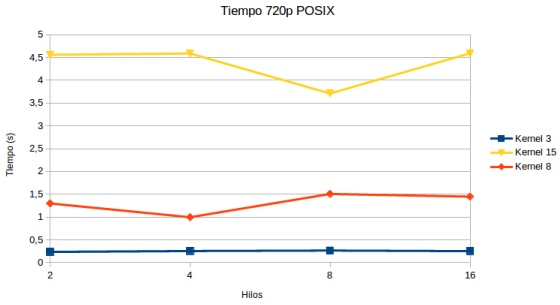

Figura 3: Tiempo para la imagen 720p con un kernel de tamaño 3, 8 y 15

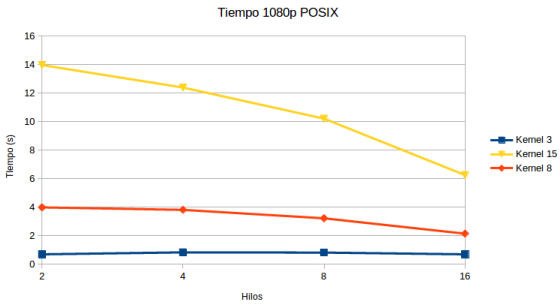

Figura 4: Tiempo para la imagen 1080p con un kernel de tamaño 3, 8 y 15

VII. Speed-up

El Speed-up es una medida de rendimiento mediante la cual se busca aumentar la eficiencia de ejecución de un proceso ejecutado de dos formas diferentes, en este caso, éstas serán de forma secuencial y para con diferentes cantidades de hilos, este valor es calculado con la siguiente función tomando los datos descritos en las anteriores tablas.

VII-A. Hilos POSIX

Figuras 8, 9 y 10

VII-B. OpenMP

Figuras 11, 12 y 13

Conclusiones.

1. El tiempo de ejecución del procedimiento mejora o empeora con diferentes cantidades de hilos, depende totalmente de los recursos que se tengan a disposición.
2. El efecto borroso de la imagen se incrementa con el tamaño del kernel usado.
3. El tiempo de ejecución aumenta a medida que se aumenta el tamaño del kernel, sin embargo para un mismo tamaño de kernel, el tiempo disminuye a medida que se aumenta el número de hilos.
4. La paralelización aplicada a ese problema aumenta el Speed up considerablemente.

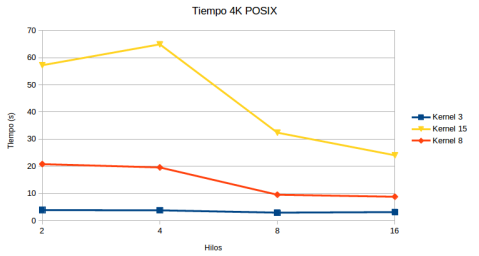

Figura 5: Tiempo para la imagen 4K con un kernel de tamaño 3, 8 y 15

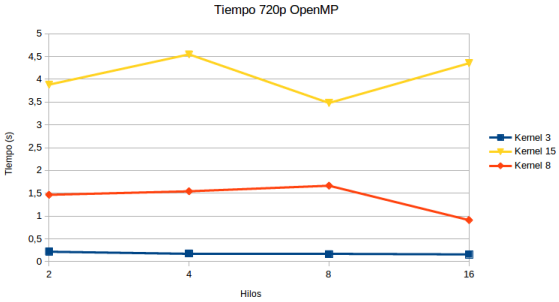

Figura 6: Tiempo para la imagen 720p con un kernel de tamaño 3, 8 y 15

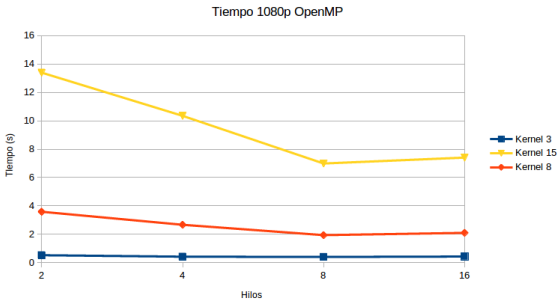

Figura 7: Tiempo para la imagen 1080p con un kernel de tamaño 3, 8 y 15

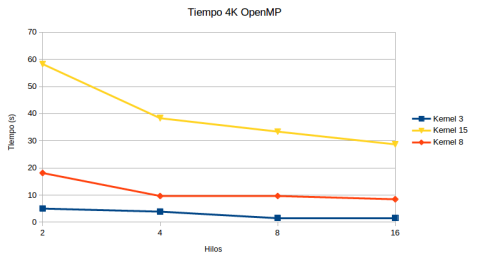

Figura 8: Tiempo para la imagen 4K con un kernel de tamaño 3, 8 y 15

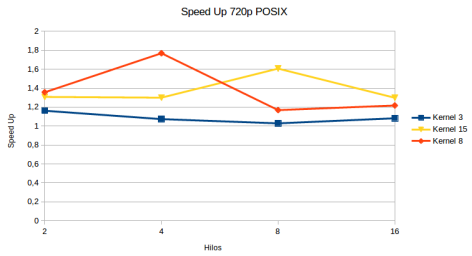

Figura 9: Tiempo para la imagen 4K con un kernel de tamaño 3, 8 y 15

Figura 10: Multiplicación del valor del pixel por la matriz gaussiana.

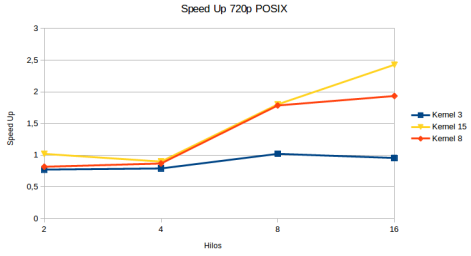

Figura 11: Resultado de la aplicación del efecto.

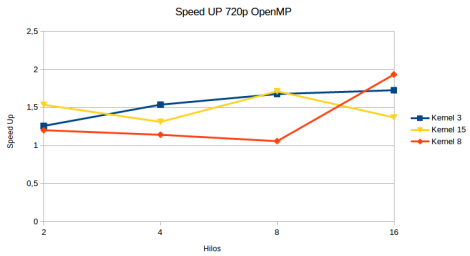

Figura 12: Tiempo para la imagen 4K con un kernel de tamaño 3, 8 y 15