Implementación efectos blanco y negro y borroso a imágenes de forma secuencial y paralela

Julián Beltrán, Julián Pereira, Sarah Lancheros

Universidad nacional de Colombia

***Resumen*—En esta práctica se pretende hacer la implementación de dos efectos, blanco y negro y efecto borroso. Se realizará el experimento con imágenes de diferentes tamaños, 720p, 1080p y 4K. La implementación se realizará en primera instancia de forma secuencial, y luego se paralelizará el proceso para analizar los tiempos promedios de ejecución, así como el SpeedUp correspondiente para cada caso; en primer lugar usando OpenMP y luego CUDA.**

***Abstract*—In this practice it is intended to implement two effects, black and white and blur effect. The experiment will be carried out with images of different sizes, 720p, 1080p and 4K. The implementation will be realized sequentially in the first place, and then it will be parallelized in order to analyze the execution times and the corresponding SpeedUp for each case; using OpenMP and then CUDA.**

Introducción

En este documento se presenta el informe correspondiente a la implementación de los efectos de blanco y negro y de "blur" para tres tipos de imágenes de diferente calidad, 720p, 1080p y 4k. Estas imágenes difieren en su resolución, en el primer caso, hay un total de 1280 píxeles en total, llegando hasta 8 megapíxeles para el último caso. A mayor resolución de la imagen, se tiene más calidad en la misma y por tanto, tal como se observará en los resultados, el proceso a realizar tomará más tiempo.

Las imágenes a tratar se encuentran en formato PNG con tres canales por píxel, estos son, RGB, las siglas hacen referencia a los colores primarios, rojo, verde y azul. Cada píxel toma un color que está representado por la mezcla por adición de estos colores, de forma que con la agrupación de todos los píxeles se genera una imagen.

En la primera práctica se realizaron varias muestras del tiempo de ejecución de ambos programas (uno para el efecto de blanco y negro y otro para el efecto blur) al aplicar estos efectos a las imágenes de diferente tamaño. Lo anterior con la implementación secuencial de los algoritmos.

En la segunda práctica, para ambos efectos se realizó una paralelización con el fin de disminuir los tiempos de ejecución del programa, para ello se reparten filas de píxeles de las imágenes entre los hilos que se usen, que en este caso fueron de 2, 4, 8 y 16. De forma que cada hilo realice el proceso borroso o blanco y negro en la región asignada, dividiendo la carga en los núcloes del CPU del computador usado.

I. METODOLOGÍA

I.I. Implementación secuencial

A. Efecto blanco y negro

Para el efecto de blanco y negro, se recorre la imagen pixel por pixel, recordando que la misma puede ser representada como una matriz en la que cada elemento de la misma corresponde a un pixel y por lo tanto, es de caracter tridimensional, conteniendo los 3 valores correspondientes al RGB. Por lo tanto, el proceso para aplicar el filtro consiste en recorrer la imagen pixel por pixel, hallar el valor promedio entre los valores del RGB y asignar este valor a cada uno de los canales del píxel. Se repite el proceso para todos los pixeles de forma que al final se obtiene la imagen a blanco y negro.

La fórmula que expresa la operación realizada para cada píxel es:

$$R, G, B = (R + G + B) / 3$$

El proceso es el mismo para las tres imágenes sin importar su tamaño. Sin embargo, dado el aumento de la cantidad de píxeles, entre más grande la imagen, este proceso tomará más tiempo.

B. Efecto borroso

Usando la misma representación de la imagen que para el caso anterior, para el efecto borroso se hace uso de un kernel de tamaño tres para estas prácticas. Es decir, se hace uso de los píxeles que quedan alrededor del píxel a modificar y que son abarcados por una matriz de tamaño 3x3 , en este caso, nueve píxeles. Se calcula un promedio de los valores de los tres canales RGB de estos nueve pixeles, y el nuevo pixel es generado con los mismos, para crear así una nueva imagen con el efecto deseado. Es decir, se calcula el promedio entre los valores del canal R (rojo) de los nueve pixeles, y este se le asigna al valor del mismo canal en el nuevo píxel; se repite este procedimiento para el color verde y azul. De forma que el píxel que se encuentra en la mitad tome valores de los píxeles de su alrededor, generando una imagen más borrosa.

De forma análoga a la implementación del filtro blanco y negro, el proceso es el mismo para las imágenes sin importar el tamaño de las misas, y se mantiene la relación proporcional entre el tamaño de la imagen y el tiempo de procesamiento de

esta.

I.II. Implementación paralela con omp

C. Paralelización

Para realizar la paralelización del procesamiento de la imagen se hizo uso de omp parallel con el fin de darle la instrucción al compilador de paralelizar un bloque de código específico, en este caso la función que procesa el archivo.
De forma que la imagen a procesar fue dividida en partes iguales dependiendo de los hilos a usar, definiendo filas de píxeles iniciales y finales, es decir, bloques para cada hilo. Como es común en la paralelización, a medida que un hilo modifica la memoria, esta se encuentra disponible para todos, así que se obtiene la misma imagen, aunque fue modificada por diferentes hilos.

Por ejemplo, para el caso de una imagen de 720p (1280 x 270 píxeles) si se ejecuta el programa con 4 hilos, la división de las filas de píxeles de la imagen entre los hilos se realiza de forma que cada hilo procesa una sección de la imagen así:

- Hilo 0: Entre la fila 0 y la fila 179.
- Hilo 1: Entre la fila 180 y la fila 359.
- Hilo 2: Entre la fila 360 y la fila 539.
- Hilo 3: Entre la fila 540 y la fila 720.

Esta división es representada también en la siguiente figura.

I.III.Implementación Paralela con CUDA

Para el caso de la implementación paralela en la GPU, usando CUDA, hay varios puntos diferentes que se ven reflejados en los resultados.

En primer lugar, el dispositivo en el cual se ejecutó el programa cambió, dado a que estas ejecuciones se llevaron a cabo en el ambiente de Google Colab, en el cual se cuenta con las siguientes especificaciones:

- Número de multiprocesadores: 13.
- Número máximo de hilos por multiprocesador: 2048.
- Número máximo de hilos por bloque: 1024.

Adicionalmente, fue necesario también hacer modificaciones en cuanto al algoritmo de procesamiento de la imagen, esto debido a que inicialmente, se manejaba la imagen directamente con la estructura png_bytep perteneciente a la librería de libpng. Sin embargo, se presentaron varias dificultades al intentar pasar dicha estructura al kernel de CUDA como parámetro. Por lo anterior, se tomó la decisión de pasar toda la información a un array plano, el cual resulta mucho más manejable en el kernel. De esta manera, el proceso en general consiste en:

- Leer la imagen haciendo uso de la función de la librería libpng.
- Copiar la información de la imagen a un array.
- Realizar el procesamiento del array con la información de la imagen de forma paralela, haciendo uso del kernel de CUDA.
- Copiar la información del array resultante de vuelta a la estructura png_bytep de la librería libpng.
- Escribir la nueva imagen haciendo uso de la función de la librería libpng.

Teniendo en cuenta lo anterior, la forma en que se divide y procesa la información también cambia. Ya que ahora se tiene únicamente un array plano con toda la información de la imagen. La siguiente figura representa el balance de carga entre los diferentes hilos para este caso, tomando como ejemplo una imagen de resolución 4k procesada por 512 hilos, teniendo en cuenta que esta cuenta con 3840 x 2160 = 8294400 píxeles.

Para el caso de la implementación con CUDA, se cubrió únicamente el caso del filtro blanco y negro.

II. RESULTADOS

Todos las tablas de resultados y gráficos pueden ser encontradas aquí.

II.I. Implementación secuencial

A continuación se presentan los tiempos de ejecución obtenidos al ejecutar los programas correspondientes a los dos filtros (Efecto Blur y Efecto Blanco y Negro) para el caso de las tres imágenes. Los tiempos son separados en las diferentes secciones del programa (lectura de la imagen, escritura o guardado de la imagen, procesamiento de la imagen y tiempo

total). Esto debido a que la paralelización será realizada únicamente en el procesamiento, por lo cual resulta útil tener el dato de cuánto tarda esta parte del programa en específico.

Efecto blanco y negro

A. Imagen 720p

EFECTO BLANCO Y NEGRO – 720p				
Ejecución	Tiempo Lectura	Tiempo Escritura	Tiempo Proceso	Tiempo Total
1	0,026264	0,241288	0,007344	0,274911
2	0,108007	0,525495	0,030342	0,66389
3	0,024018	0,22831	0,007184	0,259518
4	0,024022	0,232734	0,007262	0,264033
5	0,025877	0,240565	0,007894	0,274349
Promedio	0,0416376	0,2936784	0,0120052	0,3473398

B. Imagen 1080p

EFECTO BLANCO Y NEGRO – 1080p				
Ejecución	Tiempo Lectura	Tiempo Escritura	Tiempo Proceso	Tiempo Total
1	0,061883	0,562522	0,018299	0,64272
2	0,05576	0,562015	0,017734	0,635528
3	0,058729	0,510429	0,016968	0,58614
4	0,060281	0,557346	0,017648	0,635291
5	0,057491	0,525868	0,017208	0,600581
Promedio	0,0588288	0,543636	0,0175714	0,620052

C. Imagen 4k

EFECTO BLANCO Y NEGRO – 4k				
Ejecución	Tiempo Lectura	Tiempo Escritura	Tiempo Proceso	Tiempo Total
1	0,254337	1,969647	0,070379	2,294382
2	0,277462	2,120855	0,075001	2,473338
3	0,248504	1,965877	0,071193	2,285594
4	0,285306	2,001793	0,07536	2,362476
5	0,256201	2,024157	0,072206	2,352579
Promedio	0,264362	2,0164658	0,0728278	2,3536738

Efecto Blur

A. Imagen 720p

EFECTO BLUR – 720p				
Ejecución	Tiempo Lectura	Tiempo Escritura	Tiempo Proceso	Tiempo Total
1	0,054013	0,355136	0,030429	0,439598
2	0,053953	0,375564	0,040782	0,470315
3	0,055293	0,334864	0,030662	0,420836
4	0,055872	0,372359	0,032871	0,461118
5	0,054288	0,34729	0,030527	0,432121
Promedio	0,0546838	0,3570426	0,0330542	0,4447976

B. Imagen 1080p

EFECTO BLUR– 1080p				
Ejecución	Tiempo Lectura	Tiempo Escritura	Tiempo Proceso	Tiempo Total
1	0,125853	0,75099	0,070598	0,947459
2	0,115343	0,740282	0,068552	0,924194
3	0,114909	0,724658	0,066858	0,906438
4	0,135583	0,83724	0,078621	1,05146
5	0,117781	0,731516	0,073893	0,923203
Promedio	0,1218938	0,7569372	0,0717044	0,9505508

C. Imagen 4k

EFECTO BLUR– 4k				
Ejecución	Tiempo Lectura	Tiempo Escritura	Tiempo Proceso	Tiempo Total
1	0,527928	3,448715	0,28226	4,258922
2	0,536881	3,537523	0,292748	4,367166
3	0,514128	3,442889	0,273192	4,230226
4	0,508659	3,579995	0,274883	4,363556

5	0,505655	3,213731	0,268674	3,988081
Promedio	0,5186502	3,4445706	0,2783514	4,2415902

II.II. Implementación paralela con omp

Dado que la paralelización fue realizada únicamente para la sección del programa correspondiente al procesamiento de la imagen (y no para la lectura y escritura de esta), los tiempos de procesamiento son los únicos que presentan cambios significativos, por lo tanto, son los únicos consignados en las tablas a continuación. En estas se presentan los tiempos de ejecución del programa paralelizado para el procesamiento de la imagen con diferentes cantidades de hilos (2, 4, 8 y 16).

Efecto Blanco y Negro

A. Imagen 720

EFECTO BLANCO Y NEGRO – 720p – Tiempo Proceso				
Ejecución	2 Hilos	4 Hilos	8 Hilos	16 Hilos
1	0.003794	0.003497	0.002035	0.002261
2	0.004422	0.001957	0.002121	0.00207
3	0.00322	0.001872	0.001628	0.00193
4	0.003544	0.001739	0.001666	0.002135
5	0.003328	0.001925	0.001582	0.002046
Promedio	0.0036616	0.002198	0.0018064	0.0020878

B. Imagen 1080

EFECTO BLANCO Y NEGRO – 1080p – Tiempo Proceso				
Ejecución	2 Hilos	4 Hilos	8 Hilos	16 Hilos
1	0.009194	0.007898	0.004616	0.004931
2	0.009735	0.00447	0.004639	0.00453
3	0.007834	0.006456	0.00374	0.004315
4	0.007914	0.00416	0.003719	0.004077
5	0.007907	0.00413	0.003875	0.004166
Promedio	0.0085168	0.0054228	0.0041178	0.0044038

C. Imagen 4k

EFECTO BLANCO Y NEGRO – 4k – Tiempo Proceso				
Ejecución	2 Hilos	4 Hilos	8 Hilos	16 Hilos
1	0.034424	0.020248	0.017116	0.017748
2	0.036027	0.019393	0.020448	0.02149
3	0.031376	0.015636	0.014646	0.014903
4	0.031631	0.01571	0.01731	0.014948
5	0.032392	0.016044	0.015995	0.015188
Promedio	0.03317	0.0174062	0.017103	0.0168556

Efecto Blur

A. Imagen 720p

EFECTO BLUR – 720p – Tiempo Proceso				
Ejecución	2 Hilos	4 Hilos	8 Hilos	16 Hilos
1	0.01707	0.007434	0.007648	0.010042
2	0.015017	0.008125	0.009495	0.00700
3	0.013713	0.007298	0.011828	0.006891
4	0.014095	0.007162	0.007182	0.007059
5	0.01397	0.006878	0.006539	0.007058
Promedio	0.014773	0.0073794	0.0085384	0.007609

B. Imagen 1080p

EFECTO BLUR – 1080p – Tiempo Proceso				
Ejecución	2 Hilos	4 Hilos	8 Hilos	16 Hilos
1	0.035592	0.017515	0.015205	0.015964
2	0.035238	0.016673	0.01771	0.01795
3	0.032113	0.01783	0.015759	0.017093
4	0.032159	0.016938	0.015903	0.018097
5	0.032221	0.016938	0.015676	0.016236
Promedio	0.0334646	0.0171788	0.0160506	0.017067

C. Imagen 4K

EFECTO BLUR – 4k – Tiempo Proceso				
Ejecución	2 Hilos	4 Hilos	8 Hilos	16 Hilos
1	0.142632	0.074305	0.084177	0.084678
2	0.14585	0.092236	0.081795	0.07154
3	0.125993	0.075729	0.072841	0.060182
4	0.130486	0.068115	0.078184	0.075942
5	0.141986	0.081529	0.072591	0.073714
Promedio	0.1373894	0.0783828	0.0779176	0.0732116

II. III. Implementación Paralela con CUDA

Como en el caso anterior, se tomó únicamente en cuenta el tiempo de procesamiento, que es la región paralelizada del código. Los datos con los datos usados para hallar los promedios se encuentran en el documento cuyo enlace se encuentra al inicio de la sección de resultados. A continuación se presenta únicamente el promedio calculado.

EFECTO BLANCO Y NEGRO – Tiempo de respuesta			
	Tiempo promedio (s)		
Hilos	720p	1080p	4k
16	0,1877186	0,2697476	0,6796792
64	0,141844	0,1809154	0,3895036
128	0,123649	0,1483184	0,258786
512	0,1232596	0,1390018	0,2235272
1024	0,1259826	0,1398352	0,228675
2048	0,1152092	0,1189132	0,1435978

III. ANÁLISIS

III.I. Implementación secuencial

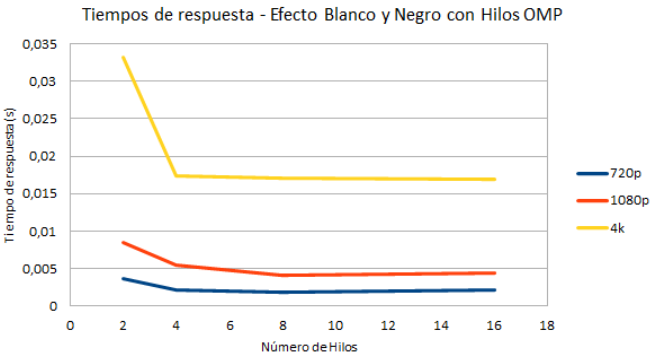
Para cada imagen se tomaron valores promedio al aplicar ambos filtros tanto en la implementación secuencial como en la paralela, con el fin de ser más precisos en la medición. Como era de esperar, para las imágenes de 720p su tiempo es menor dado la cantidad reducida de píxeles que tiene en comparación a las demás imágenes. Para la implementación de ambos efectos en una imagen 4k, el tiempo es considerablemente alto comparado con las dos imágenes antes probadas.

III.I. Implementación paralela con OMP

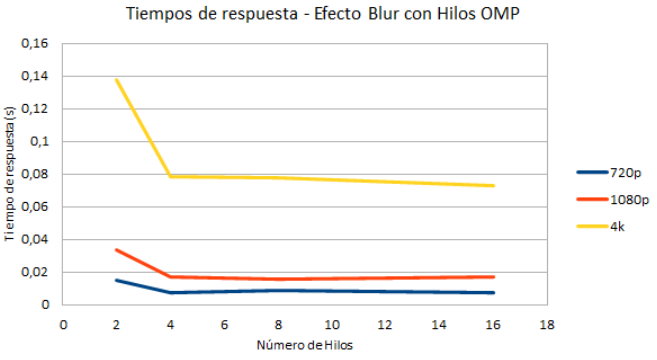
En cuanto a la implementación paralela del programa, las siguientes gráficas muestran: El tiempo de respuesta del programa para los diferentes tamaños de imagen y con diferente cantidad de hilos, el SpeedUp del programa al compararlo con la implementación secuencial.

Tiempos de respuesta

A. Efecto blanco y negro - Hilos OMP

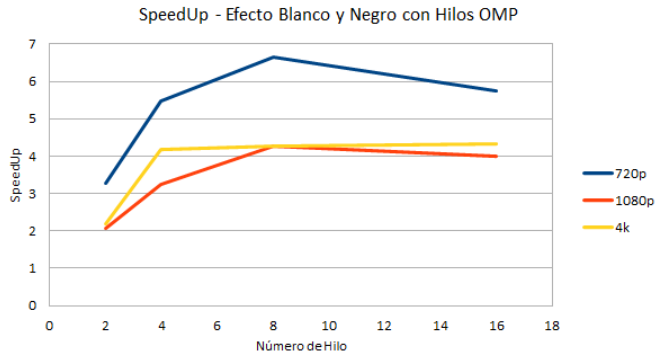


B. Efecto Blur - Hilos OMP

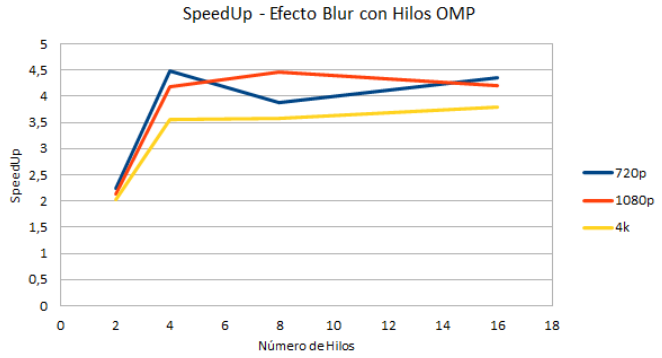


SpeedUp

A. Efecto blanco y negro - Hilos OMP



B. Efecto Blur - Hilos OMP



A primera vista, se puede observar que en efecto la paralelización del procesamiento de la imagen proporciona mejoras en los tiempos de ejecución de esta sección del programa. Adicionalmente, se identifica en todas las gráficas realizadas que el "codo" de la curva se presenta siempre alrededor de los 4 hilos, lo cual corresponde a la cantidad de núcleos físicos que posee el procesador en el cual fue ejecutado el programa.

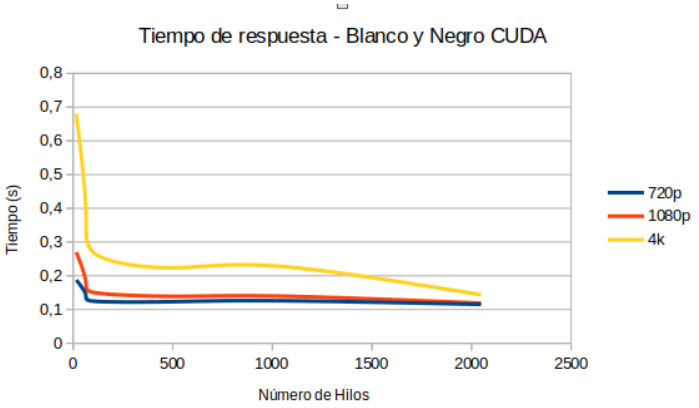
El procesador es un Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz, que posee 4 núcleos y soporta 8 subprocesos.

III.II. Implementación Paralela con CUDA

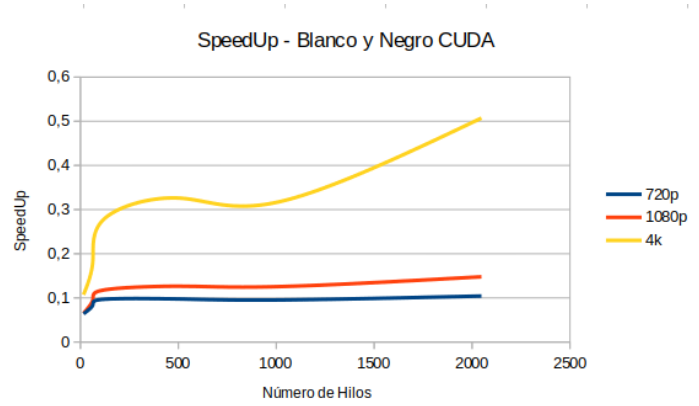
En cuanto a los resultados obtenidos para la paralelización con CUDA, resulta difícil realizar una comparación con las otras implementaciones, debido a que, como se mencionó anteriormente, fue necesario cambiar el algoritmo como tal para realizar la paralelización con esta tecnología. Sin embargo, se mantienen las conclusiones referentes a la relación entre el tamaño de las imágenes, el número de hilos y los tiempos de procesamiento, con la diferencia que la implementación en CUDA nos permite hacer uso de la GPU, ampliando en gran medida la cantidad de hilos disponibles y su impacto sobre el tiempo de ejecución.

A continuación las gráficas de tiempos de respuesta y SpeedUp para este caso.

Tiempos de respuesta



SpeedUp



IV. CONCLUSIONES

La implementación de ambos efectos fue satisfactoria para las tres imágenes propuestas.

Dado el tamaño del kernel usado (de tres), es difícil percibir el efecto borroso en las imágenes a primera vista, especialmente en las imágenes con mayor resolución. Sin embargo, cuando se detalla la imagen de cerca es posible visualizar el efecto borroso en esta. Consideramos que al emplear tamaños de kernel mayores se podría obtener un efecto más notorio.

El tiempo para realizar la conversión de las imágenes es proporcional a la complejidad del algorítmo de cada efecto, pues el efecto blanco y negro es más sencillo de calcular, y por lo tanto, su tiempo de ejecución es menor.

Como se esperaba, los tiempos de ejecución son mayores entre mayor resolución tenga la imagen, siendo el tiempo de 720p y 1080p (HD) casi similar y el de 2080p (4k) ya considerablemente mayor.

El “codo” de las gráficas fue en 4 hilos ya que este corresponde al número de núcleos físicos del procesador utilizado.

Evidentemente, al realizar cambios en el algoritmo, en cierta medida se pierde el objeto de realizar comparaciones entre la implementación en CUDA y el resto de implementaciones. Sin embargo, se aprecia el potencial de esta tecnología en la medida de la gran cantidad de paralelización que ofrece al hacer uso de la GPU directamente.

Es importante considerar la forma en que se va a implementar el algoritmo desde un inicio para evitar que, como en este caso, se presenten inconvenientes a la hora de paralelizarlo con cierta tecnología.

Referencias

Libpng.org. 2021. *libpng Home Page*. [online] Available at: <http://www.libpng.org/pub/png/libpng.html> [Accessed 16 November 2021].

Zarb.org. 2021. *A simple libpng example program*. [online] Available at: <http://zarb.org/~gc/html/libpng.html> [Accessed 10 November 2021].

En.wikipedia.org. 2021. *Box blur - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Box_blur> [Accessed 16 November 2021].