Práctica 1 - Efecto borroso de una Imagen

Tania Paola Hurtado Pinto, Nicolás Leonardo Maldonado Garzón
Universidad Nacional Bogotá, Colombia
Email: tphurtadop@unal.edu.co, nlmaldonadog@unal.edu.co

***Resumen*—El siguiente documento busca presentar la implementación hecha del algoritmo gaussiano para desenfocar imagenes implementado en el lenguaje de programación C. Para esto, se implementó el algoritmo de manera secuencial y luego con el uso de hilos para 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 respecto al algoritmo secuencial.**

I. Introducción.

El procesamiento de imágenes es un método usado desde hace alrededor de 60 años para mejorar el aspecto de las imágenes, haciendo evidente ciertos detalles que se desea hacer notar. Existen tanto métodos ópticos como digitales para realizar este proceso, donde actualmente los mas usados son los métodos digitales gracias a los recursos computacionales con los que se cuenta[3]. En muchas ocaciones se requiere procesar una imagen borrosa o desenfocada para poder observarla mucho mas clara y notable, como es el caso de la aplicación en imágenes obtenidas con fines de diagnostico médico o análisis de imágenes aéreas del terreno para determinar recursos naturales, fallas geológicas, entre otras. Este procesos se realiza a partir de la apliación de filtros que eliminan ruido, suavizan la imagen, realzan o detectan bordes realizando operaciones sobre los pixeles para conseguir el efecto especial que se desee.

La mayoría de estos métodos requieren de muchos recursos computacionales y en muchas ocaciones toma bastante tiempo el procesamiento de las imágenes, ante esto la paralelización es una muy buena alternativa al permitir dividir probemas grandes en mas pequeños que luego son resueltos simultaneamente. A pesar de que dicha alternativa no siempre es buena, puesto que en algunas ocaciones toma mas tiempo la paralelización que su mismo uso, en este caso es facilmente aplicable a la operación entre pixeles.

En esta práctica se usó un filtro gaussiano con el cual se obtiene un efecto de suavizado y borrosidad de la imagen. Este efecto, mezcla ligeramente los colores de los pixeles vecinos, lo cual hace que la imagen se vea mas suave aunque menos nítida. Para la implementación de este efecto, se usó el lenguaje de programación C y una librería para el procesamiento de imágenes, OpenCv, la cual permite majenar imágenes en todos los formatos y tamaños. Además de esto, dado que el procesamiento requiere recursos computacionales suficientes y tarda demasiado tiempo especialmente al trabajar con imágenes de gran tamaño como 4k, se realizó una paralelización del programa para el procesamiento de la imagen y se analizaron los tiempos alcanzados para las distintas imágenes procesadas y el número de hilos usados.

II. Marco Teórico.

II-A. Procesamiento Digital.

El procesamiento digital de imágenes es un conjunto de técnicas que se aplican a las imágenes con el fin de mejorar su calidad o modificar algo en ellas; éste ha tenido grandes avances y sumado a los que han tenido algunas tecnologías como la paralelización, estos cambios se pueden realizar de una manera rapida y eficaz.

II-B. Paralelización.

Podemos definir la paralelización o paralelismo como la forma de que un procesamiento, en nuestro caso el procesamiento de la imágen, pueda dividirse en subprocesos, realizar varios de estos al tiempo y asi agilizar el proceso.

II-C. Kernel.

Se entiende como una matriz de coeficientes donde el entorno del punto (x,y) que se considera en la imagen para obtener g(x,y) está determinado por el tamaño y forma del kernel seleccionado.

III. Librerías

Para el manejo de imagenes se empleo la libreria "OpenCV" en 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.

IV. Multiplicación de Matrices.

Consiste en el proceso de producir una matriz con dimensiones M x O, realizando el producto entre otras dos matrices, siempre y cuando cumplan la caracteristica de que la primera tenga dimensiones M x N y la segunda N x O. Este proceso se realiza, multiplicando cada elemento de una fila *i* de la primera matriz respectivamente con un elemento de una columna *j* de la segunda matriz, luego sumar los resultados de esos productos y asi obtener una valor para la casilla *i,j* de la nueva matriz.

En general una matriz de tamaño M x N se puede multiplicar con cualquier matriz que tenga un tamaño N x O lo cual da como resultado una matriz de tamaño M x O.

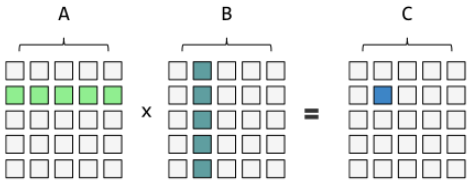


Figura 1: Multiplicación entre matrices.

V. Paralelización del algoritmo.

Para la implementación del algoritmo se usaron dos funciones principales: $matriz_filtro$ en la cual se aplica la función gaussiana a la matriz kernel, la cual será del tamaño que el usuario desee. Para esto se usó la ecuación:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 y^2}{2\sigma^2}}$$

En las siguientes imágenes se puede observar la aplicación de la función gaussiana a una matriz.

(-1,1)	(0,1)	(1,1)
(-1,0)	(0,0)	(1,0)
(-1,-1)	(0,-1)	(1,-1)

Figura 2: Matriz de posiciones para cambiar el efecto gaussiano.

0.0947416	0.118318	0.0947416
0.118318	0.147761	0.118318
0.0947416	0.118318	0.0947416

Figura 3: Matriz gaussiana con kernel = 3.

Posteriormente, para aplicar el efecto gaussiano, en la función $blur$ se realiza el proceso de multiplicar la matriz kernel obtenida de la función anterior, con los pixeles de la imagen. Esto se hace tomando cada pixel y los pixeles que lo rodean en un radio de $kernel/2$. Para cada uno de estos pixeles se obtiene el valor RGB, mediante la librería utilizada, y cada valor se multiplica con el valor de la celda de la matriz gaussiana correspondiente. Despúes de esto se suman los valores resultantes de la multiplicación y este será el valor correspondiente al RGB del pixel tomado. Este proceso se realiza para cada uno de los pixeles de la imagen. En las siguientes imágenes se observa la multiplicación realizada para cada pixel asumiendo los valores para un color.

14	15	16
24	25	26
34	35	36

Figura 4: Valores de color verde (0-25) en una región de la imagen.

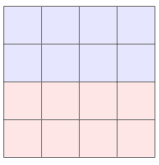
14x0.0947416	15x0.118318	16x0.0947416
24x0.118318	25x0.147761	26x0.118318
34x0.0947416	35x0.118318	36x0.0947416

Figura 5: Multiplicación del valor del pixel por la matriz gaussiana.

1.32638	1.77477	1.51587
2.83963	3.69403	3.07627
3.22121	4.14113	3.4107

Figura 6: Resultado de la aplicación del efecto.

Para realizar la multiplicación de manera paralela, se planteó dividir la matriz en el número de procesos o hilos a utilizar, y de esta manera cada unidad de procesamiento se encargaría de operar únicamente las filas que le correspondieran.



En la figura anterior, podemos ver un ejemplo de la subdivisión de cargas para dos unidades de procesamiento. En la primera, se dedicará a multiplicar las filas de color azul por la otra unidad de procesamiento rojas. Cada resultado dado, se agregará en la matriz resultado.

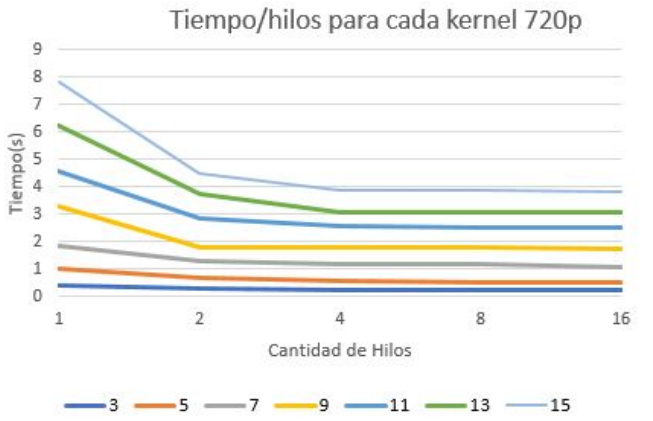
VI. Experimentos y resultados.

Se tomaron imágenes de 720p, 1080p y 4k para realizar el procedimiento del efecto borroso, ésto se realizó con kernels de 3, 5, 7, 9, 11, 13 y 15 para cada imagen y a su vez se paralelizó con 1, 2, 4, 8 y 16 hilos. El procedimiento se realizó un total de 10 veces para combinación, donde se tomó el tiempo de cada ejecucion del programa, se obtenia el promedio por combinacion, se llenaron las siguientes tablas con esos datos que posteriormente se gráficaron.

Cuadro I: Tiempo en segundos para aplicar el efecto en imagen de 720px

Hilos	Kernel 3	Kernel 5	Kernel 7	Kernel 9	Kernel 11	Kernel 13	Kernel 15
1	0,389	1,013	1,823	3,273	4,569	6,244	7,807
2	0,269	0,663	1,299	1,818	2,877	3,726	4,492
4	0,253	0,556	1,189	1,82	2,569	3,092	3,857
8	0,252	0,526	1,191	1,812	2,541	3,071	3,86
16	0,251	0,522	1,061	1,731	2,527	3,092	3,841

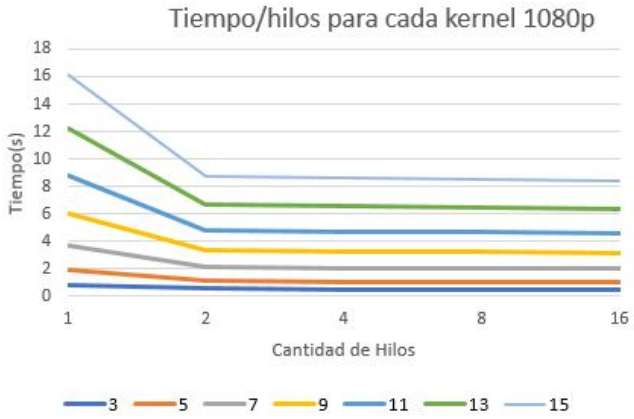
Figura 7: Grafica de tiempos para cada kernel en funcion de la cantidad de hilos.



Cuadro II: Tiempo en segundos para aplicar el efecto en imagen de 1080px

Hilos	Kernel 3	Kernel 5	Kernel 7	Kernel 9	Kernel 11	Kernel 13	Kernel 15
1	0,389	1,013	1,823	3,273	4,569	6,244	7,807
2	0,269	0,663	1,299	1,818	2,877	3,726	4,492
4	0,253	0,556	1,189	1,82	2,569	3,092	3,857
8	0,252	0,526	1,191	1,812	2,541	3,071	3,86
16	0,251	0,522	1,061	1,731	2,527	3,092	3,841

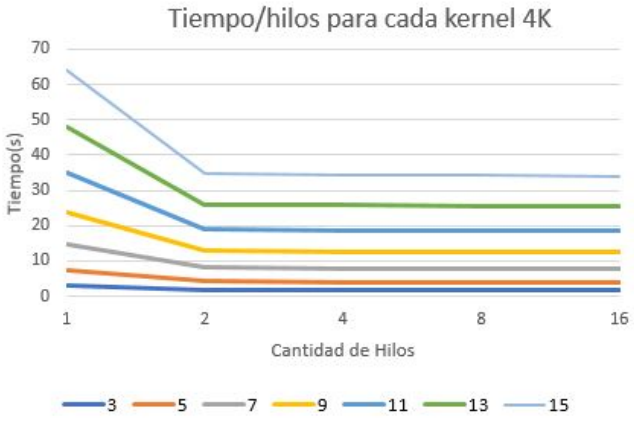
Figura 8: Grafica de tiempos para cada kernel en funcion de la cantidad de hilos.



Cuadro III: Tiempo en segundos para aplicar el efecto en imagen de 4k

Hilos	Kernel 3	Kernel 5	Kernel 7	Kernel 9	Kernel 11	Kernel 13	Kernel 15
1	0,802	1,908	3,724	5,96	8,814	12,255	16,206
2	0,604	1,165	2,091	3,295	4,83	6,672	8,741
4	0,481	1,06	2,02	3,217	4,713	6,527	8,582
8	0,46	1,054	1,994	3,197	4,628	6,495	8,549
16	0,45	1,035	1,956	3,128	4,583	6,34	8,432

Figura 9: Grafica de tiempos para cada kernel en funcion de la cantidad de hilos.



VII. Speed Up.

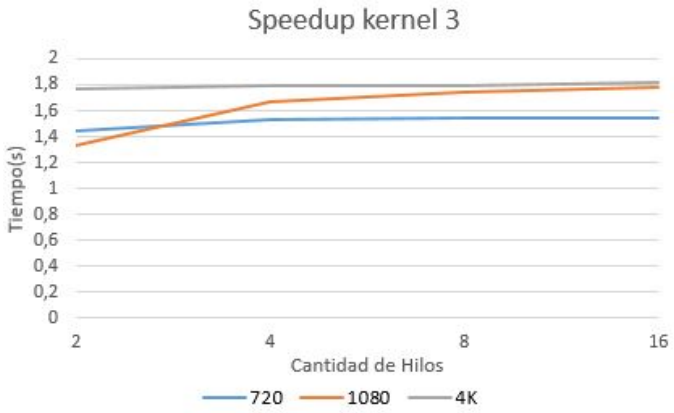
El Speedup es una medida de rendimiento mediante la cual se busca aumentar la eficiencia de ejecución de un proceso ejecutado de dos formas diferentes, en este caso, éstas serán de forma secuencial y paralelizada con diferentes cantidades de hilos, este valor es calculado con la siguiente función tomando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

$$S_p(n) = \frac{T^*(n)}{T_p(n)}$$

Cuadro IV: Resultado de realizar el Speedup con un kernel de tamaño 3

Hilos	720	1080	4k
2	1,45	1,33	1,77
4	1,54	1,67	1,79
8	1,54	1,74	1,80
16	1,55	1,78	1,82

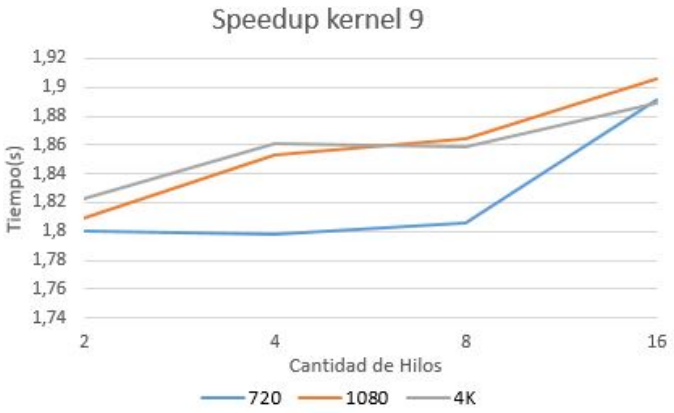
Figura 10: Gráfica del Speedup para cada imágen procesada con un kernel 3.



Cuadro V: Resultado de realizar el Speedup con un kernel de tamaño 9

Hilos	720	1080	4k
2	1,80	1,81	1,82
4	1,80	1,85	1,86
8	1,81	1,86	1,86
16	1,89	1,91	1,89

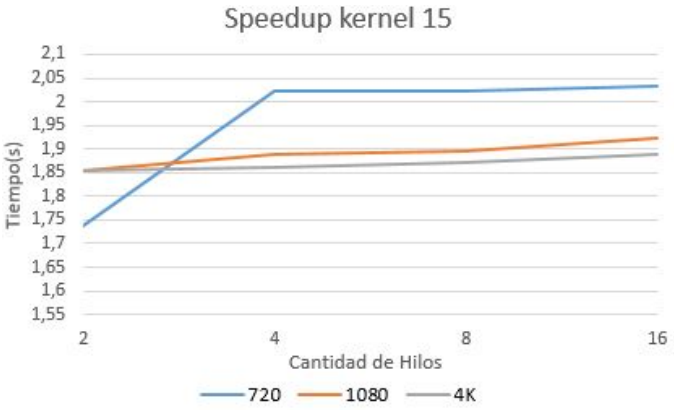
Figura 11: Gráfica del Speedup para cada imágen procesada con un kernel 9.



Cuadro VI: Resultado de realizar el Speedup con un kernel de tamaño 15

Hilos	720	1080	4k
2	1,74	1,85	1,86
4	2,02	1,89	1,86
8	2,02	1,90	1,87
16	2,03	1,92	1,89

Figura 12: Gráfica del Speedup para cada imágen procesada con un kernel 15.



Conclusiones.

1. La capacidad de medir si el tiempo de ejecucion del procedimiento mejora o empeora con diferentes cantidades de hilos, depende totalmente de los recursos que se tengan a disposición.
2. El efecto borrosos 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 reduce el speedup considerablemente.
5. En cuanto a las gráficas de speedup se puede observar que dados los limitados recusos computacionales con los que se cuentan, no se observa una gran diferencia conforme se incrementan la cantidad de hilos, debido a que al paralelizar solo encola los nuevos procesos.

Referencias

[1] http://www.pixelstech.net/article/1353768112-Gaussian-Blur-Algorithm.
[2] Gaussian Blur. Articulo de Wikipedia. https://en.wikipedia.org/wiki/Gaussian_blur.
[3] Daniel Malacara, Procesamiento de imágenes. Tomado de internet. http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen2/ciencia3/084/htm/sec_9.htm.