# Práctica 2

Juan Camilo Acosta Rojas, Simón Aparicio Bocanegra, Jhon Sebastián Rojas Rodríguez,

***Resumen*—La computación paralela es una importante herramienta disponible para los diseñadores de aplicaciones para el uso más eficiente de los recursos computacionales disponibles. En el caso de estudio a trabajar en la materia, algoritmos de procesamiento de imágenes, gracias a la disposición matricial de los datos permiten la paralelización de algoritmos de convolución usados. Filtros convolucionales pueden ser aplicados en diversos campos como: detección de bordes, contraste, eliminación de ruido, entre muchas otras. Poder distribuir los subprocesos mediante distintos métodos de paralelización es una solución efectiva y rápida para solucionar dichos problemas.**

***Index Terms*—proceso, paralelizar, imagen.**

## I. Introduction

La computación paralela ofrece grandes oportunidades en campos como la congestión de programas en la nube, optimización de procesos, escalabilidad en las páginas web, ejecución de algoritmos complejos. La computación paralela consiste en la división de procesos y su ejecución simultanea para un mejor aprovechamiento de los recursos computacionales. La aplicación de filtros convolucionales nos ofrecen un problema que puede ser dividido y optimizado usando la teoría de la computación paralela para obtener tiempo total de ejecución menor. El objetivo de este trabajo es diseñar, implementar y analizar un algoritmo que aplique un filtro convolucional sobre una imagen y posteriormente paralelizar la ejecución del mismo y comparar los tiempos de procesamientos finales.

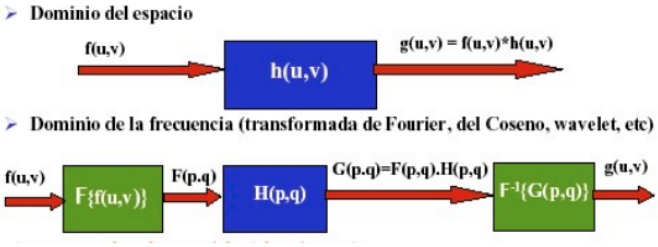
## II. Procesamiento de imágenes

Los métodos de procesamiento se pueden dividir en:

- Métodos del espacio: Trabajan directamente sobre el arreglo de píxeles de entrada, es decir, se modifican directamente los píxeles de la imagen. Se trata de algoritmos locales que transforman o bien el valor de cada píxel tomado de manera individual o bien el de un pequeño conjunto de ellos.

- Métodos de frecuencia: Son más costosos en términos computacionales y se utilizan métodos como transformaciones de Fourier.

Estos dos métodos no son mutuamente excluyentes, de hecho, algunos métodos provienen de la combinación de ellos. Un caso de aplicación de estos métodos son los filtros y las convoluciones. Existen además dos tipos de transformaciones: transformaciones globales, donde cada píxel de salida depende de un píxel de entrada, el resultado no varía de acuerdo a la vecindad de los píxeles, si son permutados aleatoriamente o si son reordenados; y transformaciones locales, donde el valor de cada píxel depende de la vecindad del mismo.




[1]

Figura 1. Procesamiento de imágenes

- De manera global:
  $R(x,y) := f(A(x,y))$ ó $R(x,y) := f(A(x,y), B(x,y))$
- De manera local:
  $R(x,y) := f(A(x-k, y-k), ..., A(x,y), ..., A(x+k, y+k))$

Mediante estos procedimientos ya sean globales o locales, se puede modificar una imagen.
Dentro de las convoluciones que se pueden aplicar en el procesamiento de imágenes las convoluciones discretas, las más usadas, son una combinación lineal de los valores de los píxeles vecinos de la imagen mediante una matriz de coeficientes que es conocida como la máscara o el kernel de convolución. Una operación de convolución modifica además el tamaño de la imagen reduciéndolo, razón por la cual es común encontrar padding, donde se asigna 0 a los píxeles que no pueden ser calculados.
Dentro de los efectos destacados que se pueden obtener con filtros convolucionales se encuentran:

- Suavizado: Se entiende como la difuminación de la imagen, reducir los contrastes abruptos dentro de esta.
- Perfilado: Resaltar los contrastes, lo contrario al suavizado.
- Bordes: Proceso de detección de variaciones bruscas dentro del arreglo de píxeles de la imagen.
- Detección: Detección de cierto tipos de rasgos característicos en la imagen como esquinas, segmentos, entre otros.

Para el proceso de detección de bordes en una imagen existen diferentes filtros que siguen un comportamiento estadístico específico. [1].

## III. Filtros Sobel

EL filtro de Sobel es uno de los filtros de convolución de imágenes basados en los filtros espaciales de realce, los cuales consisten en resaltar características de las imágenes que hayan podido quedar emborronadas. Estos filtros están asociados a la detección de lados o bordes. El filtro Sobel detecta bordes horizontales y verticales separadamente sobre

una imagen. El origen del filtro proviene del cálculo de un operador local de derivación ya que un píxel pertenece a un borde si se produce un cambio brusco entre niveles de grises con sus vecinos. Mientras más brusco el cambio es más fácil detectar el borde [2].
La derivada de una función digital se define en términos de las variaciones entre píxeles adyacentes. Para ello, la primera derivada debe ser 0 en zonas de intensidad constante y distinta a 0 en zonas de variaciones, además, la segunda derivada debe ser 0 en zonas de intensidad constante y a lo largo de rampas con pendiente constante y distinta de 0 si se presentan variaciones, escalones y el comienzo o fin de una rampa. Para el caso del operador Sobel, se suelen usar dos máscaras para modelizar el gradiente que se llaman operadores de Sobel en donde en la matriz de coeficientes tienen más peso los píxeles situados en la posición vertical y horizontal respecto a los píxeles que están situados en la diagonal, lo que lo hace menos sensible al ruido. El gradiente digital obtenido de acuerdo al concepto de gradiente digital es [3]:

$$\nabla f(x) = \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \tag{1}$$

Para el caso del filtro Sobel se tiene que:

$$G_x = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{2}$$

$$G_y = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{3}$$

[4]

## IV. Diseño e implementación

Para esta segunda entrega se planeó el diseño de un programa secuencial paralelizable que implementa el filtro Sobel sobre una imagen. Para poder procesar las imágenes se utilizó la librería **stbimage.h**, que es una librería para tratamiento de imágenes en el lenguaje de programación C y C++. Esta librería permite hacer el cargue de una imagen en una matriz de enteros sin signo, esta matriz del tamaño de la imagen y del número de canales de la misma, que es 1 para imagenes en escala de grises y 3 para imagenes a color. Los kernels escogidos son los siguientes:

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{4}$$

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{pmatrix} \tag{5}$$

$$\begin{pmatrix} 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & -2 & -3 & -3 & -3 & -3 & -3 & -2 & 0 \\ 0 & -3 & -2 & -1 & -1 & -1 & -2 & -3 & 0 \\ -1 & -3 & -1 & 9 & 9 & 9 & -1 & -3 & -1 \\ -1 & -3 & -1 & 9 & 19 & 9 & -1 & -3 & -1 \\ -1 & -3 & -1 & 9 & 9 & 9 & -1 & -3 & -1 \\ 0 & -3 & -2 & -1 & -1 & -1 & -2 & -3 & 0 \\ 0 & -2 & -3 & -3 & -3 & -3 & -3 & -2 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 \end{pmatrix} \tag{6}$$

Estos filtros siguen el comportamiento de los filtros Sobel para la detección de bordes de una imagen. En este caso, se desea utilizar un producto de matrices para realizar la convolución con cada uno de los filtros mostrados.

El algoritmo usado consiste en la iteración sobre subselecciones de píxeles de la imagen, su multiplicación convolucional con el kernel

Para ello se va a implementar una convolución por cada canal de los píxeles de la imagen de entrada, es decir, un procedimiento de multiplicación de matrices por cada canal e insertando el valor de cada convolución de cada píxel en una reserva de memoria del mismo tamaño de la imagen de entrada. Posteriormente, se escribe la imagen que obtiene la convolución y, por último, se carga.

A diferencia de la entrega anterior, se utilizó la librería **openMP** para poder implementar directivas y métodos de paralelización para el algoritmo de convolución, en este caso se decidió paralelizar cada selección de subselecciones para agilizar la multiplicación de matrices durante la convolución, para ello se implementan directivas de la librería variando el número de hilos con el cual se desea manejar el fragmento de código. La matriz en cuestión, para cada una de las operaciones, es dividida a lo largo de su eje horizontal en $4\dot{n}_{hilos}$ número de secciones. La directiva **for** de OMP es llamada sobre la loop encargada de la iteración de secciones.

## V. Análisis y Resultados

Para poder medir la eficiencia del programa, se decidieron realizar tres experimentos con cada uno de los kernel mencionado anteriormente con imágenes de distinta resolución (720p, 1080p y 4k), en donde para cada experimento se midieron los tiempos de ejecución. Además de esto, se varía el número de hilos para determinar los tiempos de respuesta en cada escenario.

| Kernel | Hilos | 720p | 1080p | 4k |
|---|---|---|---|---|
| **3** | 1 | 0.682041 | 1.955333 | 7.949599 |
| **3** | 2 | 0.396688 | 1.043194 | 5.264791 |
| **3** | 4 | 0.367686 | 1.08427 | 4.979557 |
| **3** | 8 | 0.379222 | 1.188295 | 5.002508 |
| **3** | 16 | 0.368746 | 1.24771 | 4.970138 |
| **2** | 1 | 0.101913 | 0.386891 | 1.296072 |
| **2** | 2 | 0.085151 | 0.223624 | 0.854712 |
| **2** | 4 | 0.089533 | 0.222499 | 0.843984 |
| **2** | 8 | 0.0795 | 0.22753 | 0.910337 |
| **2** | 16 | 0.06633 | 0.197748 | 0.869749 |
| **1** | 1 | 0.102759 | 0.298996 | 1.20328 |
| **1** | 2 | 0.087245 | 0.218345 | 0.894832 |
| **1** | 4 | 0.085866 | 0.32756 | 0.841081 |
| **1** | 8 | 0.091554 | 0.272443 | 0.849172 |
| **1** | 16 | 0.077653 | 0.220627 | 0.744093 |

Cuadro I

TABLA 1. TIEMPOS DE RESPUESTA PARA LOS 3 KERNELS Y 3 LAS TRES RESOLUCIONES ESCOGIDAS VARIANDO ENTRE EL NÚMERO DE HILOS

| Kernel | Hilos | 720p | 1080p | 4k |
|---|---|---|---|---|
| **3** | 1 | 1 | 1 | 1 |
| **3** | 2 | 1.719338624 | 1.874371402 | 1.509195522 |
| **3** | 4 | 2.719711928 | 1.803363553 | 1.595643749 |
| **3** | 8 | 2,636977812 | 1.645494595 | 1.588323097 |
| **3** | 16 | 2.711893824 | 1.567137396 | 1.598667683 |
| **2** | 1 | 1 | 1 | 1 |
| **2** | 2 | 1.196850301 | 1.730096054 | 1.517542751 |
| **2** | 4 | 1.138273039 | 1.73884377 | 1.536832452 |
| **2** | 8 | 1.281924528 | 1.700395552 | 1.424815206 |
| **2** | 16 | 1,536454093 | 1.956485021 | 1.491306112 |
| **1** | 1 | 1 | 1 | 1 |
| **1** | 2 | 1.177821079 | 1.369374156 | 1.34469934 |
| **1** | 4 | 1.196736776 | 0.9127976554 | 1.430635099 |
| **1** | 8 | 1.122386788 | 1.097462589 | 1.417003858 |
| **1** | 16 | 1.32331011 | 1.355210378 | 1.617109689 |

Cuadro II

TABLA 2. SPPED UP PARA LOS 3 KERNELS Y 3 LAS TRES RESOLUCIONES ESCOGIDAS VARIANDO ENTRE EL NÚMERO DE HILOS

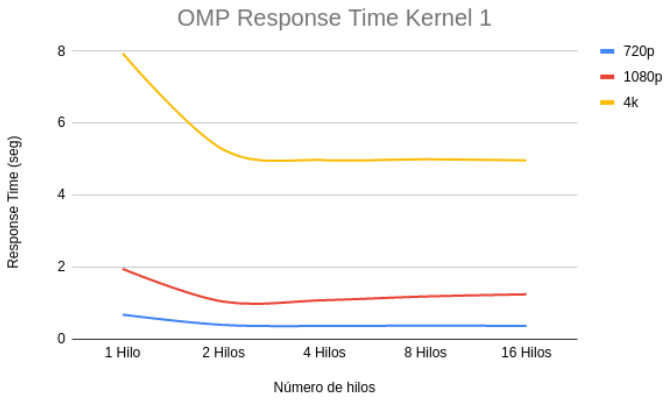



Figura 2. Tiempos de respuesta para el kernel 3 y las tres resoluciones escogidas variando el número de hilos empleados

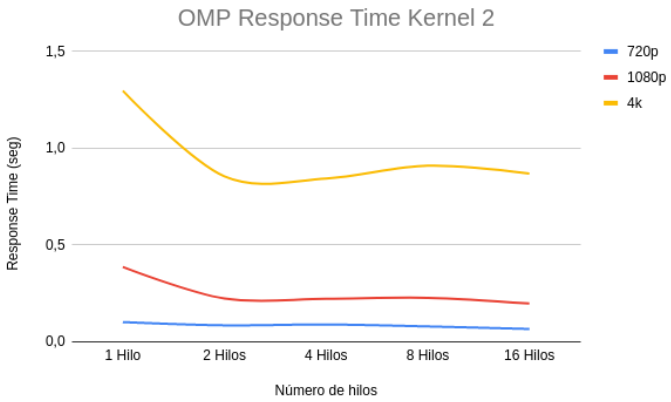



Figura 3. Tiempos de respuesta para el kernel 2 y las tres resoluciones escogidas variando el número de hilos empleados

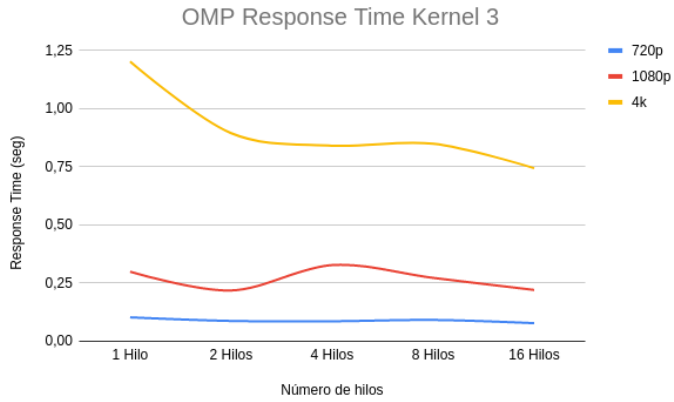



Figura 4. Tiempos de respuesta para el kernel 1 y las tres resoluciones escogidas variando el número de hilos empleados

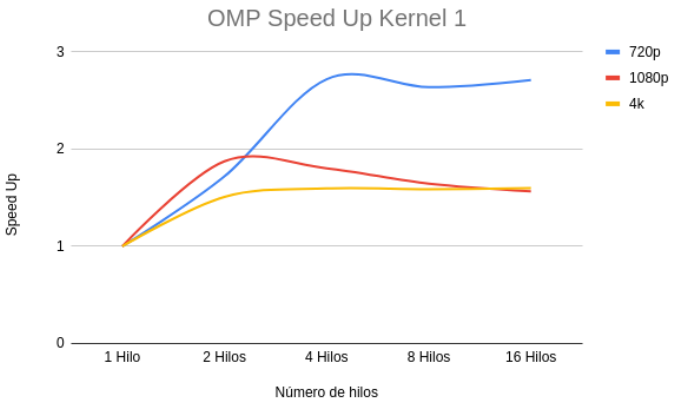



Figura 5. Speed Up para el kernel 3 y las tres resoluciones escogidas variando el número de hilos empleados

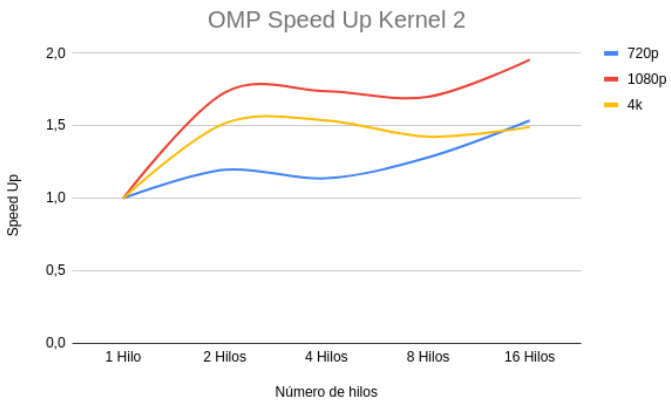



Figura 6. Speed Up para el kernel 2 y las tres resoluciones escogidas variando el número de hilos empleados

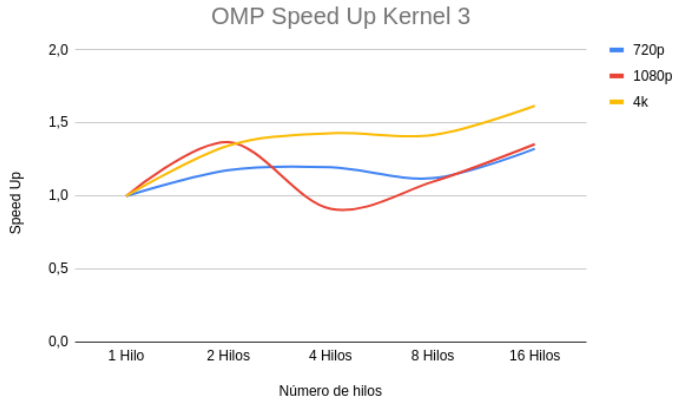



Figura 7. Speed Up para el kernel 1 y las tres resoluciones escogidas variando el número de hilos empleados

Observamos el comportamiento esperado para los tiempos de respuesta y los valores de Speed-Up, teniendo en cuenta el número de núcleos en los que corrieron las pruebas. Se observa una reducción de los tiempos de respuesta y un aumento del Speed-Up hasta los 4 nucleos, valor a partir del cual un aumento en el número de núcleos no representa una mejora notable.

## VI. Conclusion

- Los algoritmos de procesamiento de imágenes cumplen un papel muy importante en varios ámbitos de la ciencia y la investigación, y poder optimizar sus procesos para obtener una respuesta más rápida es un logro importante debido a las diferentes complejidades que tiene dicho procesamiento.
- La librería OMP se pueden paralizar diferentes procesos de diferentes maneras, para el caso de multiplicación de matrices, dividiendo en diferentes secciones se puede lograr un mejor resultado en términos de tiempos de respuesta y Speed Up del proceso.
- Con respecto al programa secuencial elaborado en la primera práctica, se observa una mejoría considerable usado métodos de paralelización, por lo que la implementación de métodos de la computación paralela son efectivos en este tipo de operaciones, sin embargo para ciertas tareas y ciertas formas de paralización no lo es, se debe tener un análisis previo antes de poder considerar un proceso como paralelizable.

## Referencias

[1] B.E.R, Jimena Olivares Montiel, Çonvolución y filtrado", LAPI [Online]. Available: http://lapi.fi-p.unam.mx/wp-content/uploads/6-Filtros-y-morfologia.pdf
[2] "Filtros de detectar bordes: Sobel"[Online]. Available: https://docs.gimp.org/2.8/es/plug-in-sobel.html
[3] "Tema 3: Filtros"[Online]. Available: http://grupo.us.es/gtocoma/pid/tema3-2.pdf
[4] I.P. Nacional, .Extracción de bordes: Operadores de Sobel, Prewit y Roberts", Boletín UPIITA [Online]. Available: http://grupo.us.es/gtocoma/pid/tema3-2.pdf