INFORME PRÁCTICA 2: PROCESAMIENTO DE IMÁGENES CON EFECTO DE DETECCIÓN DE BORDES UTILIZANDO EL OPERADOR SOBEL Y OpenMP.

Andres Camilo Correa Romero
correo: anccorrearo@unal.edu.co
Heidy Johanna Alayón Sastoque
correo: halayont@unal.edu.co

RESUMEN: *En este documento se muestra el desarrollo de la práctica 2 donde se compara el rendimiento de un algoritmo para el procesamiento de imágenes con efecto de detección de bordes utilizando el operador sobel, mediante la programación secuencial y luego con OpenMP variando la cantidad de hilos y mostrando los resultados obtenidos.para tiempo de respuesta y speedup.*
Las imágenes que se procesaron tienen 640p, 1280p, 1920p y 6K.

PALABRAS CLAVE: Procesamiento de imágenes, detección de bordes, sobel, openmp.

1 INTRODUCCIÓN

El filtrado de imágenes busca obtener un resultado con determinadas características a partir de una imagen origen, existen diferentes técnicas que se pueden aplicar y que han sido ampliamente documentados. Para el objeto de esta práctica se aplicará la detección de bordes utilizando el operador sobel que se centra en las diferencias de intensidad que se dan pixel a pixel. para obtener los contornos de objetos.

La detección de bordes se refiere al proceso de identificar y localizar discontinuidades nítidas en una imagen. Las discontinuidades son cambios abruptos en la intensidad de los píxeles que caracterizan los límites de un objeto en una escena. El método clásico de detección de bordes implica convolucionar una imagen con un operador a (filtro 2-D), que está construido para ser sensible a un gran gradiente en una imagen mientras devuelve valores de 0 en regiones uniformes.

1.1 OPERADOR SOBEL

El operador Sobel es utilizado en procesamiento de imágenes, especialmente en algoritmos de detección de bordes. Técnicamente es un operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad de una imagen. Para cada punto de la imagen a procesar, el resultado del operador Sobel es tanto el vector gradiente correspondiente como la norma de este vector.

El resultado muestra cómo de abruptamente o suavemente cambia una imagen en cada punto analizado y, en consecuencia, cuán probable es que este represente un borde en la imagen y, también, la orientación a la que tiende ese borde.

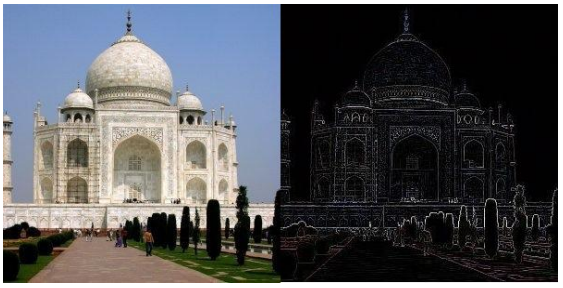


Figura 1. Imagen antes y después de aplicar el operador sobel

1.2 TRATAMIENTO MATEMÁTICO

Cuando se utiliza la detección de bordes Sobel, la imagen se procesa en las direcciones X e Y por separado primero y luego se combina para formar una nueva imagen que representa la suma de los bordes X e Y de la imagen.

Se usa lo que se llama convolución de kernel. Un kernel es una matriz de 3 x 3 que consta de índices ponderados de manera diferente (o simétrica).

X – Direction Kernel

-1	0	1
-2	0	2
-1	0	1

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

Se pretende implementar este algoritmo en el lenguaje de programación C++, primero de manera secuencial y luego particionar el problema para que pueda ser ejecutado en diferentes hilos OpenMP (2, 4, 8 y 16), con estas simulaciones se obtiene datos para el tiempo de respuesta en cada ejecución y el speedup. Se comparan estas medidas para cuantificar el desempeño de cada implementación y se muestra de manera gráfica el comportamiento de las mismas,

2 DESARROLLO

Para el operador de sobel, es necesario realizar una conversión de la imagen a escala de grises, posteriormente se realiza un blur gaussiano y finalmente se aplica el kernel de sobel en dirección horizontal y vertical.

Las operaciones que más consumen tiempo son el blur gaussiano y el operador de sobel, debido a esto se decidió paralelizar ambas funciones.
El procedimiento para ambas operaciones es similar, ya que se realiza la convolución de un kernel en cada una, por esto se explicará el desarrollo general.

La imagen es interpretada como un arreglo unidimensional, esto para que sea más sencillo repartir las cargas a cada hilo de manera dinámica. Para evitar el false sharing se realiza una copia de la imagen para cada hilo, de esta manera dos hilos no accederán a los mismos espacios de memoria. A medida que el hilo va realizando la convolución, se va marcando en una matriz auxiliar los píxeles modificados, de esta manera al finalizar se puede copiar a la imagen resultante únicamente la información modificada por el hilo.

El repositorio se encuentra en https://github.com/z4yross/sobel-paralel-kernel.

3 RESULTADOS Y ANÁLISIS

El ejercicio se realizó sobre la siguiente máquina:

Ram	16 GB
Procesador	i7 - 8700k 3.8-4.8GHZ 6 cores 12 threads
OS	POP OS basado en ubuntu

Para probar el código se usó la misma imágen con diferentes resoluciones (640p,1280p, 1920p y 6k), los resultados de tiempo de ejecución y SpeedUP se presentan en las siguientes gráficas:

3.1 Gráficas

Figura 2. Imagen original

3.1.1 Imagen de 640p

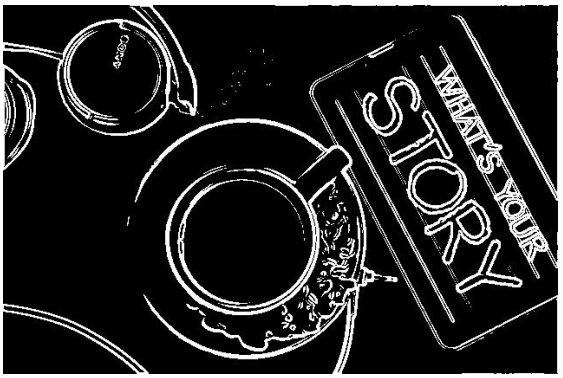
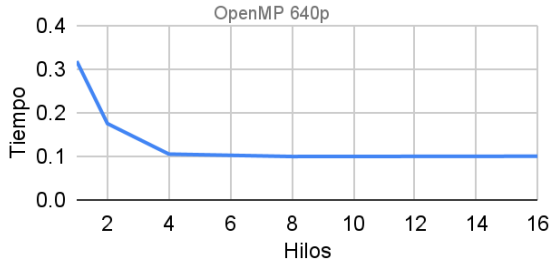


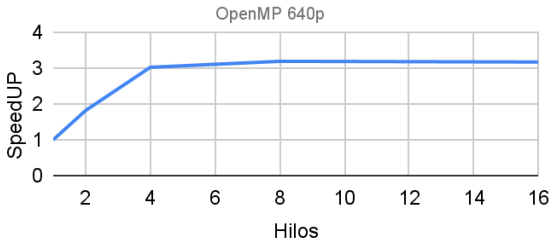
Figura 3: Resultado con imagen de 640p, 16 hilos,Con blur. Sobel threshold: 20

OpenMP 640p		
Hilos	Tiempo (s)	SpeedUP
1	0.318735	1
2	0.175404	1.817147842
4	0.105367	3.024998339
8	0.099855	3.191978369
16	0.100476	3.17225009

Tabla 1: Tiempo de ejecución promedio (10 intentos) 640p. Con blur. Sobel threshold: 20



Gráfica 1: Tiempo de ejecución promedio (10 intentos) vs Hilos 640p. Con blur. Sobel threshold: 20



Gráfica 2: SpeedUp promedio (10 intentos) vs Hilos 640p. Con blur. Sobel threshold: 20

3.1.1 Imagen de 1280p

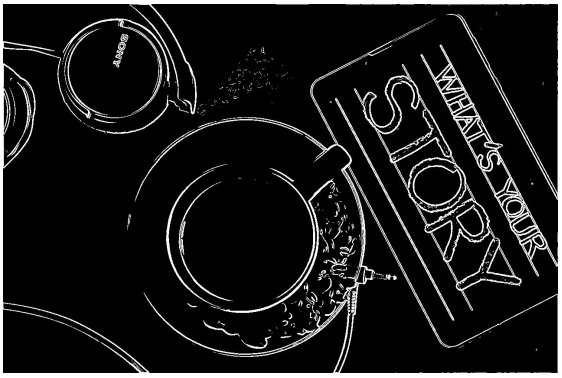
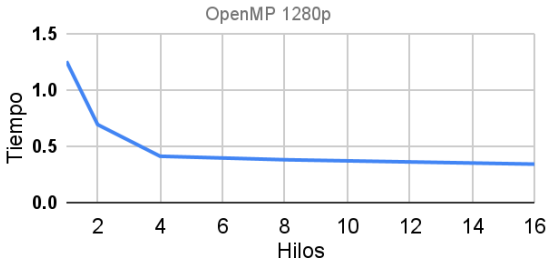


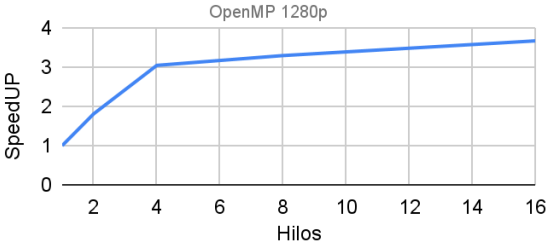
Figura 4. Resultado con imagen de 1280p 16 hilos. Con blur. Sobel threshold: 20

OpenMP 1280p		
Hilos	Tiempo (s)	SpeedUP
1	1.256039	1
2	0.693426	1.811352617
4	0.412298	3.04643486
8	0.381053	3.296231758
16	0.342137	3.671158045

Tabla 2: Tiempo de ejecución promedio (10 intentos) 1280p.Con blur. Sobel threshold: 20



Gráfica 3: Tiempo de ejecución promedio (10 intentos) vs Hilos 1280p. Con blur. Sobel threshold: 20



Gráfica 4: Tiempo de ejecución promedio (10 intentos) vs Hilos 1280p. Con blur. Sobel threshold: 20

3.1.1 Imagen de 1920p

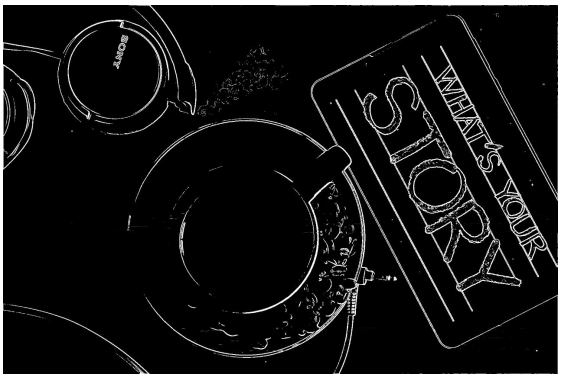
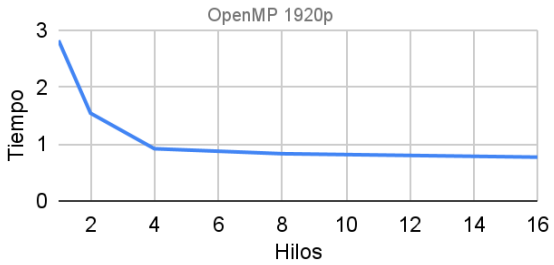


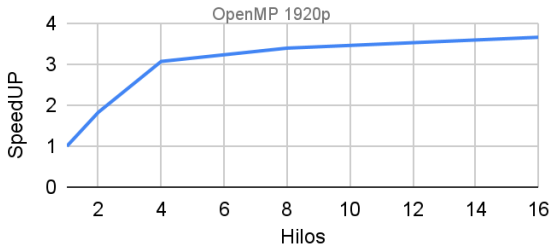
Figura 5. Resultado con imagen de 1920p 16 hilos. Con blur. Sobel threshold: 20

OpenMP 1920p		
Hilos	Tiempo (s)	SpeedUP
1	2.823733	1
2	1.544152	1.828662593
4	0.91919	3.071979678
8	0.83146	3.396114064
16	0.771118	3.661868871

Tabla 3: Tiempo de ejecución promedio (10 intentos) 1920p. Con blur. Sobel threshold: 20



Gráfica 5: Tiempo de ejecución promedio (10 intentos) vs Hilos 1920p. Con blur. Sobel threshold: 20



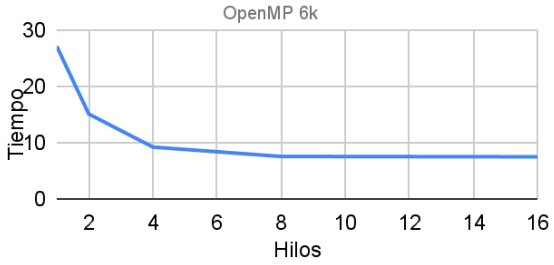
Gráfica 6: Tiempo de ejecución promedio (10 intentos) vs Hilos 1920p. Con blur. Sobel threshold: 20

3.1.1 Imagen de 6k

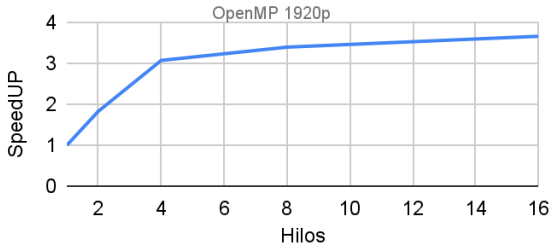
Figura 6. Resultado con imagen de 6k, 16 hilos. Con blur. Sobel threshold: 20

OpenMP 6k		
Hilos	Tiempo (s)	SpeedUP
1	27.132011	1
2	15.097596	1.79710803
4	9.226925	2.940525798
8	7.562745	3.587587708
16	7.50114	3.617051675

Tabla 3: Tiempo de ejecución promedio (10 intentos) 6000p. Con blur. Sobel threshold: 20.



Gráfica 7: Tiempo de ejecución promedio (10 intentos) vs Hilos 6000p. Con blur. Sobel threshold: 20



Gráfica 8: Tiempo de ejecución promedio (10 intentos) vs Hilos 6000p. Con blur. Sobel threshold: 20.

Se puede observar que los resultados obtenidos en cada ejecución son consistentes, mostrando disminución del tiempo de ejecución a mayor cantidad de hilos.

4 CONCLUSIONES

- Para el óptimo funcionamiento del operador sobel es preferible que la imagen esté en escala de grises ya que así no se genera tanto ruido y es más eficiente al detectar los bordes.
- Es necesario utilizar blur para mejorar la nitidez de la imagen final aunque el sobel es capaz de detectar los bordes por si mismo.
- El tiempo de ejecución con OpenMP fue similar aunque ligeramente mayor que con el uso de POSIX.
- El uso de OpenMP es menos complejo y hace mas legible el codigo.

- El resultado de la ejecución de la práctica fue consistente con la teoría
- El uso de hilos incrementa el rendimiento del procesador
- Es importante tener control de los hilos para que el resultado sea el esperado y no existan conflictos entre ellos.

5 REFERENCIAS

1 Preparación de informes en formato IEEE, http://utap.edu.co/ccys/wp-content/uploads/2018/Eventos/IEEE/Paper_IEEE_Oficial_Espa%C3%B1ol.pdf.
2 Derivados Sobel, https://programmerclick.com/article/61448242/.