Christian D. Guzmán Torres

802-15-3240

CIIC4030 Assignment #4: The median filter.

Estructura del proyecto

En mi implementación. El objeto "*Main*" actúa como el "*actor system"*. Es el encargado de pedir al usuario la imagen a ser procesada. Una vez el usuario somete la imagen, "*Main*" la envía al cliente. El cliente notifica que ha recibido la imagen y la envía al "*SequentialServer*" y al "*ParallelServer*" quienes procesan la imagen y le devuelven un "*ServerReport*" al cliente.

Implementación del algoritmo secuencial

El algoritmo es basado en el pseudo código provisto. La única diferencia es que el "*window*" se llena utilizando una función de "*BufferedImage*":

```
img.getRGB(startX, startY, w, h, rgbArray, offset, scansize)
```

Cada "*window*" se toma de tamaño 3X3. Para encontrar la mediana se hace un "*quickSort*" del arreglo. La mediana entonces, es el elemento encontrado en el centro del arreglo.

Es importante notar que gran parte del esfuerzo computacional se debe al cálculo de la media por lo cuál, de querer mejorar el "*performance*" del algoritmo se pudiera buscar una manera más rápida de calcular la media. Algunas alternativas que se sugiere en la página de Wikipedia incluyen utilizar un "selection sort" hasta encontrar la media (ya que es el único valor que necesitamos) o utilizar el método de histogramas.

Implementación del algoritmo en paralelo.

Para transformar el algoritmo en paralelo realicé dos cambios al algoritmo original.

El primer cambio es utilizar un objeto "*ParRange*" de "*scala parallel collections*" para que la iteración se ejecute en paralelo.

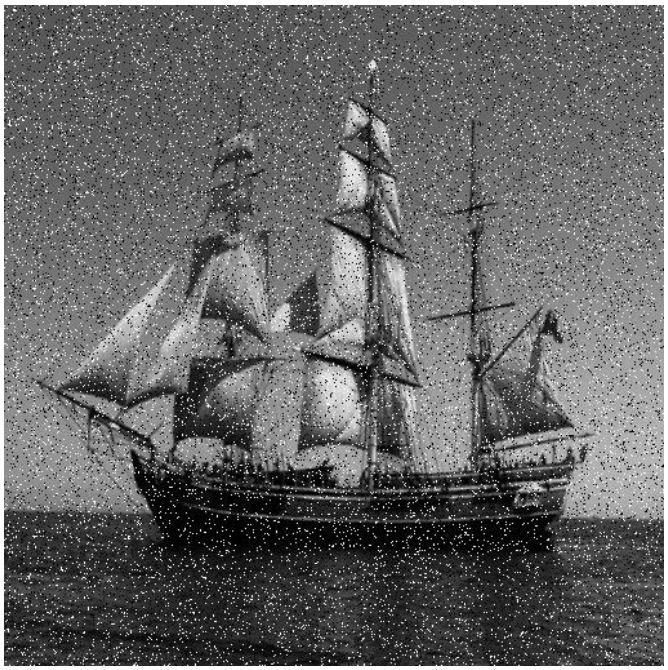
El segundo cambio es uno que tomó un poco más trabajo encontrar pero que ilustra uno de los "*Side effect operations*" que advierten en la documentación. En el algoritmo secuencial defino mi arreglo "window" fuera de la iteración y en cada iteración lleno el arreglo con los valores del nuevo "window". Al intentar lo mismo en el algoritmo en paralelo, los resultados no son los esperados. Esto se debe a que cada "*thread*" comparte el mismo arreglo "window" lo cual causa que se mezclen los valores de los diferentes "*windows*" corriendo en paralelo. Una vez

identifiqué el error, lo arreglé asegurándome que cada "*thread*" cree su propio arreglo "*window*". En este cambio además se puede observar el "*trade-off*" que ocurre al cambiar el algoritmo a paralelo. El algoritmo acelera el proceso, pero tiene un costo en memoria ya que cada "*thread*" necesita su propio espacio en memoria.

Un cambio adicional que se pudiera intentar es convertir el arreglo "*window*" a "*ParArray*". Sin embargo, debido a que en mi caso cada "*window*" tiene solo 9 elementos hay que considerar si el costo de transformarlo a paralelo es mayor que el beneficio. Para "*windows*" con pocos elementos no considero que sería una buena opción y, por lo tanto, no lo incluí en mi implementación.

"Benchmarks" del tiempo de procesamiento de cada algoritmo

- **Imagen 512X512 (Tamaños han sido reducidos en el documento para disminuir el espacio ocupado. Las imágenes originales se incluyen en el proyecto).**

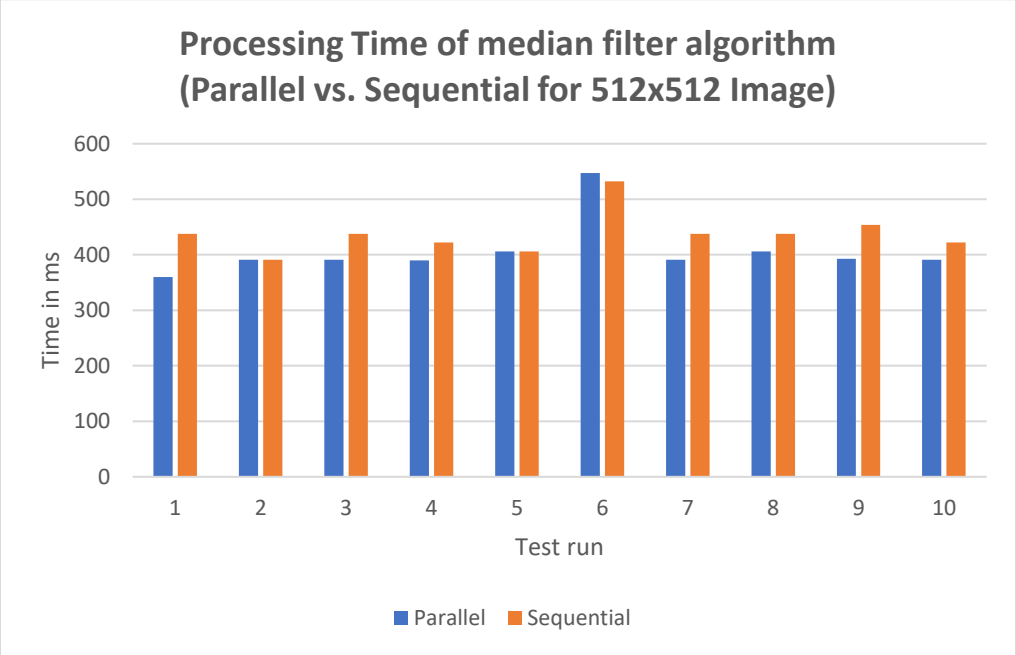


- Output de la implementación secuencial

- Output de la implementación paralela

- Resultados

		Test Run (Image with 512x512 pixels)										
		1	2	3	4	5	6	7	8	9	10	Promedio
Processing time in ms	Parallel	360	391	391	390	406	547	391	406	393	391	406.6
	Sequential	438	391	438	422	406	532	438	438	454	422	437.9
	Differencia	78	0	47	32	0	-15	47	32	61	31	31.3

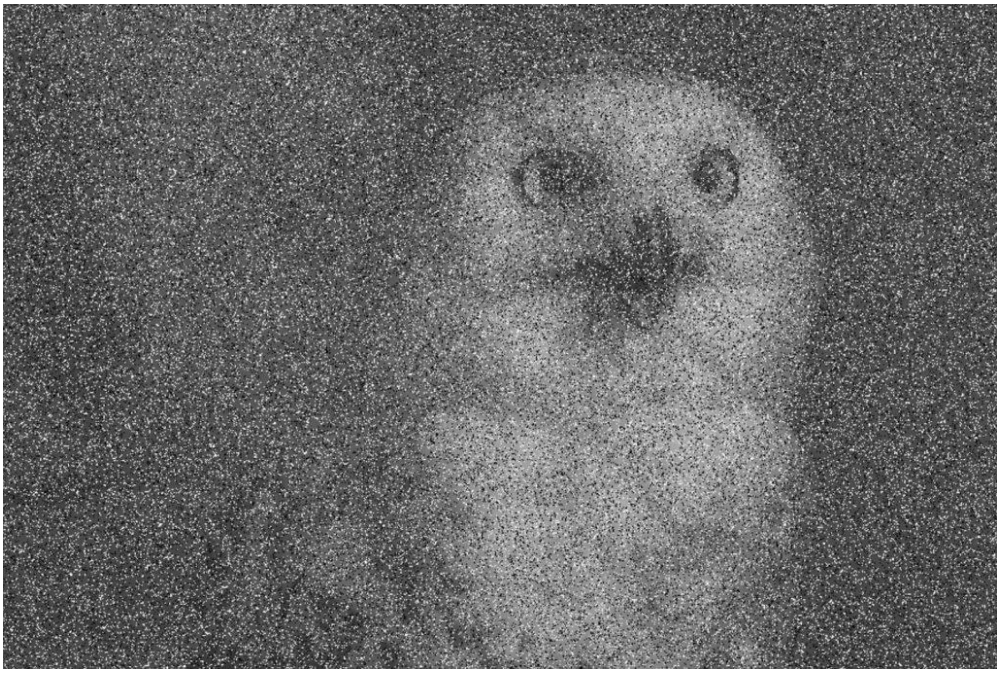


Según los resultados del "benchmark" se puede observar que la implementación en paralelo por lo general procesa la imagen en menos tiempo. En promedio ejecuta **31.2 ms** más rápido aunque, se pueden observar 2 casos en los cuáles ambos demoraron la misma cantidad de tiempo (corridas 2 y 5) y 1 en el cual el secuencial fue **15 ms** más rápido (corrida 6). Se puede observar una disminución del tiempo de procesamiento de **7.15%** en promedio.

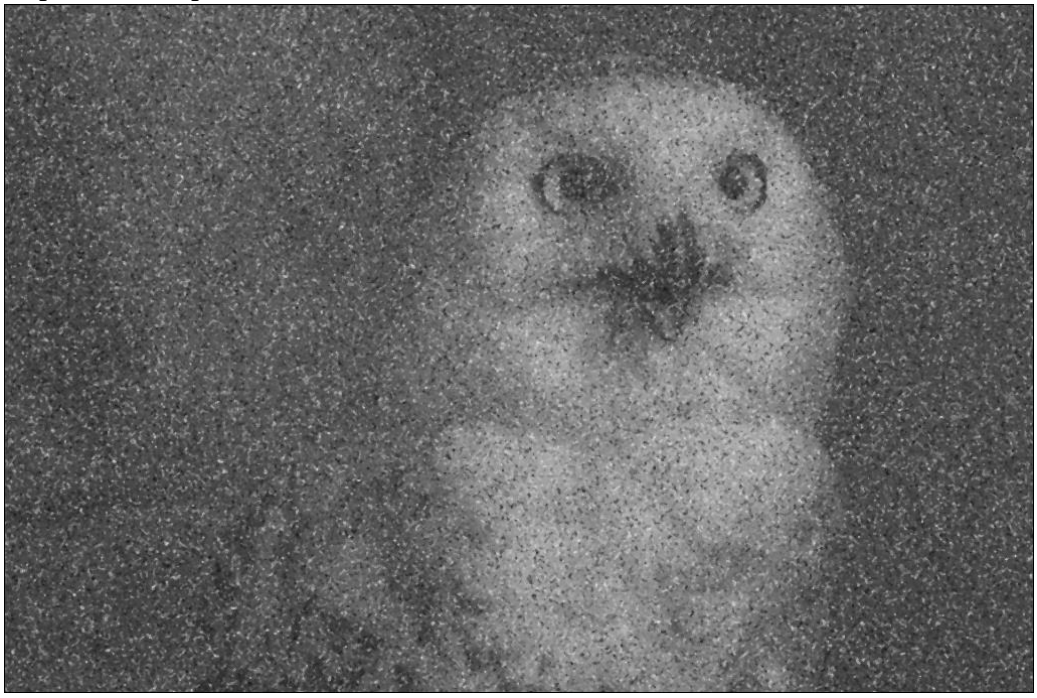
Es posible que la diferencia no sea muy notable debido a que la imagen tiene un tamaño relativamente pequeño. Con imágenes más grandes la diferencia debe ser más notable y consistente, como se puede observar en el próximo "*benchmark*"

Se puede observar que el algoritmo logró eliminar la mayoría del "*noise*" presente en la imagen.

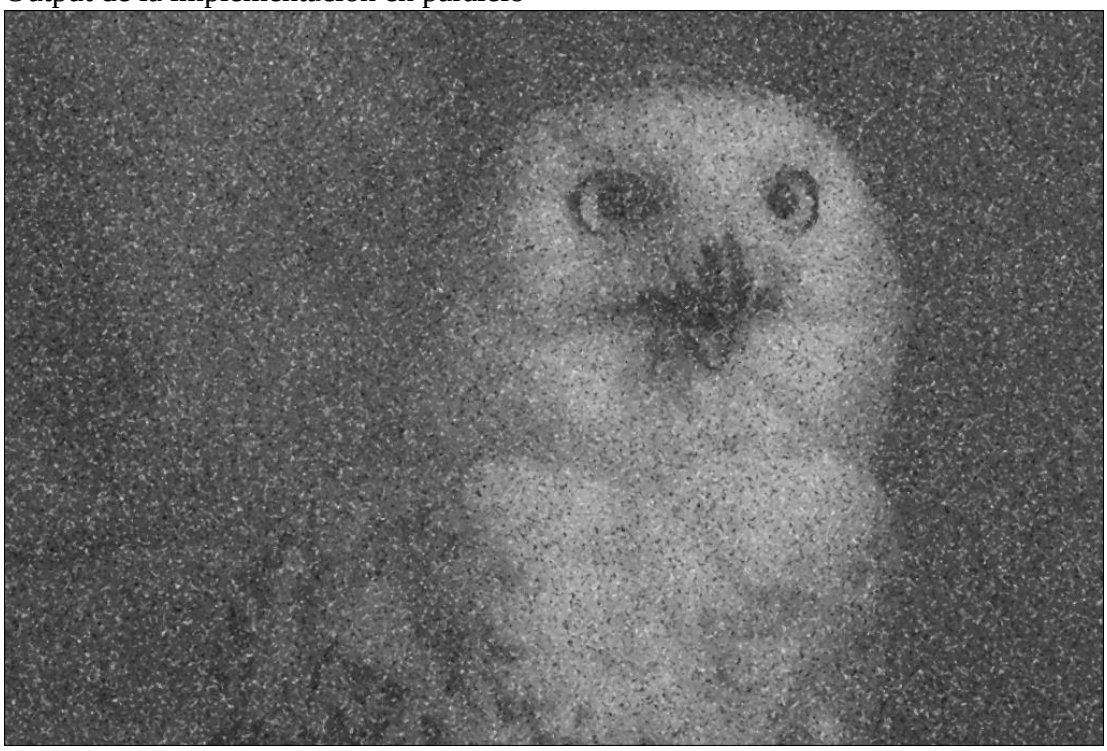
- **Imagen 1024X684 (Tamaños han sido reducidos en el documento para disminuir el espacio ocupado. Las imágenes originales se incluyen en el proyecto).**



  - Output de la implementación secuencial

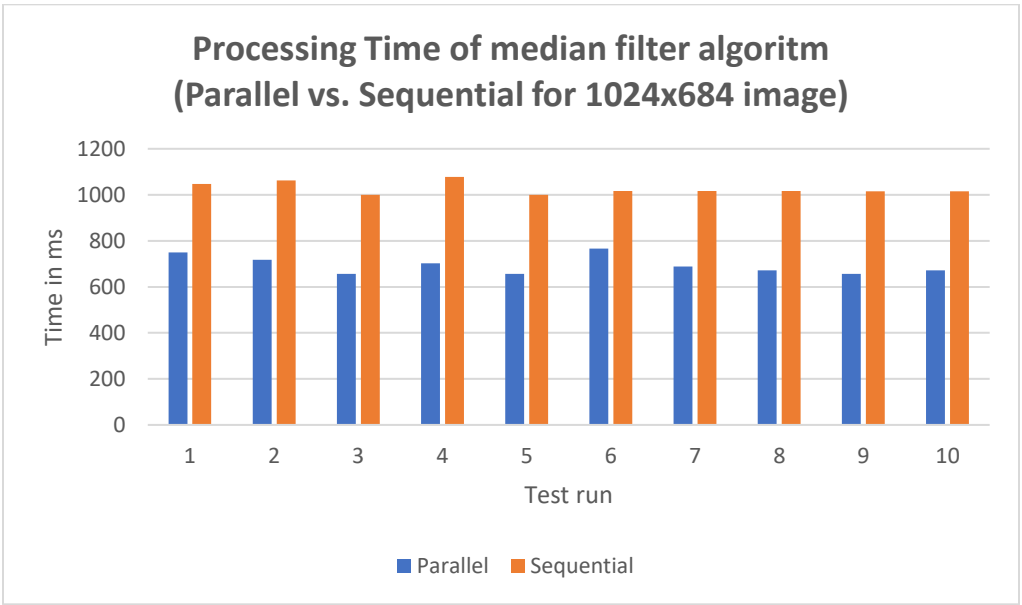


- Output de la implementación en paralelo



- Resultados

		Test Run (Image with 1024x684 pixels)										
		1	2	3	4	5	6	7	8	9	10	Promedio
Processing time in ms	Parallel	750	718	656	703	656	766	688	672	657	672	**693.8**
	Sequential	1047	1062	1000	1078	1000	1016	1016	1016	1015	1015	**1026.5**
	Differencia	297	344	344	375	344	250	328	344	358	343	**332.7**



En este caso, debido al tamaño de la imagen, se puede apreciar mejor la diferencia en procesamiento. La implementación en paralelo ejecuta en promedio **332.7 ms** (o **32.41%**) más rápido que la secuencial. La diferencia en este caso es más constante; no se observan casos en los cuales ambas ejecutan al mismo tiempo o que la secuencial ejecute más rápido. Entiendo que mientras más grande sea la imagen que se procese más notable será la diferencia en ejecución.

Sin embargo, en esta imagen el algoritmo no logra eliminar la mayoría del "*noise*" presente en la imagen. Esto se puede deber a que en esta hay mucho más "*noise*" presente. De todos modos, se puede observar una leve reducción en el "*noise*" por lo cual la imagen se ve un poco más clara. La diferencial es más notable en el área de los ojos.

Conclusión

La implementación en paralelo ejecuta más rápido que la secuencial. Sin embargo, la diferencia es más notable en imágenes grandes. Ejecutar el algoritmo en paralelo tiene un costo en la memoria y al transformar la colección a una en paralelo por lo cual, para procesar imágenes pequeñas puede ser más conveniente utilizar la implementación secuencial.

El algoritmo del "*median filter*" efectivamente reduce el "*noise*" en las imágenes. Sin embargo, en imágenes con una gran cantidad de "*noise*" la reducción no es muy notable.