

Practica 1, Computación Paralela y Distribuida

1st Cristian Chávez

Sistemas e industrial

Universidad Nacional de Colombia

Bogotá D.C, Colombia

cchavezb@unal.edu.co

2nd Leonardo Delgado

Sistemas e industrial

Universidad Nacional de Colombia

Bogotá D.C, Colombia

sdelgadom@unal.edu.co

3rd Daniel Mendivelso

Sistemas e industrial

Universidad Nacional de Colombia

Bogotá D.C, Colombia

dmendivelso@unal.edu.co

Resumen—En este documento se muestra la primera práctica de la asignatura de computación paralela y distribuida, el cual consiste en aplicar un filtro a una imagen. El filtro escogido fue el de detección de bordes. Para ello realizaremos una convolución con una matriz kernel.

Index Terms—convolución, filtro, imagen

I. INTRODUCTION

Dentro del marco de la computación paralela, uno de los campos donde se puede encontrar más ejemplos de aplicación es en el procesamiento de imágenes. Dentro del procesamiento de imágenes, la aplicación de filtros ha sido de gran relevancia en los últimos años, ya no sólo en la industria audiovisual sino que han trascendido incluso a las redes sociales. A pesar que existen una gran variedad de filtros, aquellos que se pueden expresar por medio de un kernel (matriz) son los que nos competen, ya que allí es donde se puede sacar el provecho a la parallelización de procesos. No obstante esta primera práctica no se hace uso de un método de parallelización de procesos, sino que se hacen las computaciones secuencialmente, esto con el fin de sentar con una base para hacer posteriores comparaciones con soluciones computadas en paralelo.

II. DISEÑO DE LA APLICACIÓN

El programa esta construido sobre C, haciendo uso de la librería stb_image. El programa recibe como parámetros el nombre de la imagen de entrada y el nombre como quiere que salga la imagen. El programa recibe la imagen y intenta leerla; si no puede leerla arrojara un error y terminara el programa. Ya que en este algoritmo nos interesan los bordes de la imagen realizamos una conversión a un tono monocromático, en este caso, escala de grises para tratarlo de manera mas fácil. Luego de realizar la conversión realizamos una convolución sobre la imagen en escala de grises a partir de una matriz kernel. Finalmente se exporta la imagen con el nombre puesto por el usuario.

-1	-1	-1	
-1	8	-1	
-1	-1	-1	

Figura 1. Matriz kernel de detección de bordes.

III. EXPERIMENTOS

Se plantearon 5 experimentos principalmente, aplicar el filtro a imágenes a color en 720p (figura 2), 1080p (figura 3), 1440p (figura 4), 4k (figura 5) y 8k (figura 6). Para cada una se ejecutó el programa y se tomó el tiempo.



Figura 2. Imagen de castillo 1280x720 píxeles.



Figura 3. Imagen de delfines 1920x1080 píxeles.

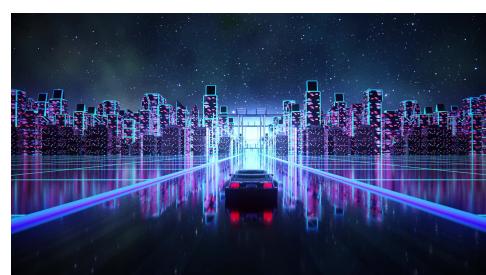


Figura 4. Imagen de carro 2560x1440 píxeles.



Figura 5. Imagen de ciudad 3840x2160 píxeles.



Figura 6. Imagen de ciudad futurista 7680x4224 píxeles.

IV. RESULTADOS

Luego de realizar los experimentos se observaron los siguientes resultados:

IV-A. *Imagen 720p*

Para una imagen de 1280 por 720 píxeles, se obtuvo el resultado que se ve en la figura 7, y el programa tardó 0.174 segundos en ejecutarse.



Figura 7. Imagen de castillo con filtro.

IV-B. *Imagen 1080p*

Para una imagen de 1920 por 1080 píxeles, se obtuvo el resultado que se ve en la figura 8, y el programa tardó 0.419 segundos en ejecutarse.

IV-C. *Imagen 1440p*

Para una imagen de 2560 por 1440 píxeles, se obtuvo el resultado que se ve en la figura 9, y el programa tardó 0.589 segundos en ejecutarse.



Figura 8. Imagen de delfines con filtro.



Figura 9. Imagen de carro con filtro.

IV-D. *Imagen 4k*

Para una imagen de 3840 por 2160 píxeles, se obtuvo el resultado que se ve en la figura 10, y el programa tardó 1.33 segundos en ejecutarse.



Figura 10. Imagen de ciudad con filtro.

IV-E. *Imagen 8k*

Para una imagen de 7680 por 4224 píxeles, se obtuvo el resultado que se ve en la figura 11, y el programa tardó 4.786 segundos en ejecutarse.

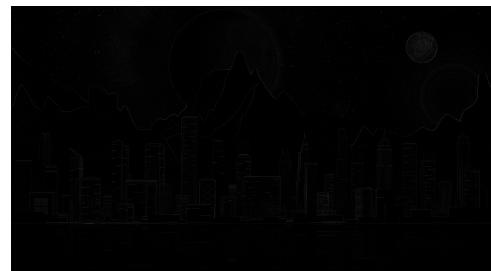


Figura 11. Imagen de ciudad futurista con filtro.

IV-F. Comparación de experimentos

Si hacemos una comparación de tiempos dada la resolución obtenemos el siguiente gráfico en el cual vemos como el tiempo aumenta de manera ligeramente exponencial

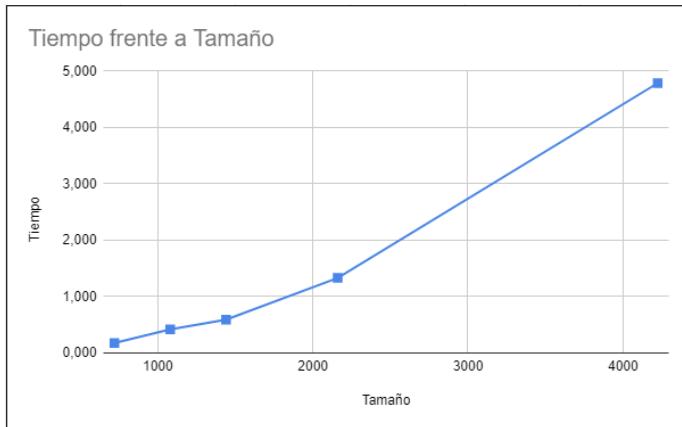


Figura 12. Comparación de tiempo por tamaño.

V. CONCLUSIONES

De la práctica anterior podemos concluir:

- La matriz kernel cumplió su función correctamente, lo cual se evidencia en los resultados vistos.
- La tendencia del tiempo necesario para aplicar el filtro parece ser ligeramente exponencial a medida que aumenta el tamaño de la imagen.
- Los tiempos que tarda el programa en aplicar el filtro para imágenes de alta resolución (+1080p) no son deseables si se espera manejar un número grande de imágenes de este tipo. Por ejemplo para aplicar el filtro a 1000 imágenes de resolución 1080p habría que esperar aproximadamente 10 minutos

REFERENCIAS

- [1] 8.2. Matriz de convolución”, Docs.gimp.org, 2021. [Online]. Available: <https://docs.gimp.org/2.6/es/plug-in-convmatrix.html>. [Accessed: 15- Nov- 2021]
- [2] Kernel (image processing) - Wikipedia”, En.wikipedia.org, 2021. [Online]. Available: [https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing)). [Accessed: 15- Nov- 2021].