UNIVERSIDAD TECNOLÓGICA METROPOLITANA
UTEM del Estado de Chile

Proyecto de Imágenes

Asignatura: Computación paralela y distribuida
Profesor: Sebastián Salazar
Fecha de entrega: 12 de Agosto de 2020
Integrantes:

- Iván Pérez
- Sebastián Pérez
- Lester Vásquez

Índice

Introducción

El procesamiento de grandes volúmenes de números es una tarea importante en computación moderna, el procesamiento de imágenes es una muestra de esto con lo que nos topamos diariamente, por lo que se busca en este proyecto poder procesar imágenes realizando cambios de difuminación, pasar una imagen en colores a una escala de grises y escalar una imagen sin que esta pierda su calidad.
Los objetivos específicos de este proyecto son los siguientes:

- Realizar el procesamiento de imágenes de forma distribuida en una red, se debe aprovechar al máximo la cantidad de computadores disponibles para el procesamiento.
- Paralelizar el procesamiento en cada equipo involucrado, con el fin de aprovechar al máximo las capacidades de cada computador en la red local.
- Procurar que el tratamiento de las imágenes no genere errores o distorsiones en la imagen resultante.

Problema

Se nos solicita crear un programa en el sistema operativo Ubuntu 20.04, este programa debe estar escrito en los lenguajes C, C++ o un mix entre los dos con su respectivo compilador.

La función de este programa es que debe recibir por líneas de comando cualquier tipo de imagen que no sea animada como por ejemplo jpg, png, bmp, tiff, etc. El programa debe resolver 3 problemas:

- **Difuminar la imagen:** Hacer que disminuya la claridad y exactitud del contorno de la imagen ingresada.
- **Imagen en colores pasar a escala de grises:** sustituir cada color a una escala que varia entre blanco y negro.
- **Escalar la imagen sin perder calidad:** la imagen ingresada se debe escalar sin perder calidad ni generar alteraciones.

El formato de salida para cada problema debe ser en formato **PNG** con la siguiente nomenclatura **operacion_numero_fecha.png** , la fecha debe estar en formato **yyyyMMddhhmmss.**

- **Ejemplo: operacion_1_20200704104917.png**

Forma de abordar el problema

El problema se puede subdividir en diferentes áreas.
Primero está la problemática de aplicar las transformaciones requeridas a la imagen de tal forma que no se generen anomalías en la imagen resultante.
Luego está el problema de distribuir el procesamiento, se debe implementar un sistema que permita dividir y enviar partes de la imágen a múltiples nodos de procesamiento, para finalmente combinar las imágenes ya procesadas en una imagen real la cual debe ser guardada en el equipo.
En cada nodo de procesamiento se deben utilizar todas las capacidades de procesamiento de este, es por esto que se requiere que el procesamiento sea paralelizado en los distintos procesadores que posea el nodo.
finalmente el problema de recibir las sub imágenes procesadas y unirlas correctamente para guardarlas en el disco.

Para el procesamiento de las imágenes se necesita difuminar, pasar a escala de grises y escalar geométricamente la imagen seleccionada, para esto se utilizará la librería OpenCV, la cual posee nativamente estas funcionalidades, además provee un mecanismo eficiente para poder particionar las imágenes, esto será de utilidad para distribuir el procesamiento, además provee un método de combinar secciones verticales de imágenes de una forma simple, es por esto que esta librería se utilizará frecuentemente en el programa.

Para la distribución de las imágenes se decide dividir las imágenes por filas en N particiones donde N es el número de nodos de procesamiento, con esto se logra que el trabajo sea distribuido equitativamente, lo cual da un mejor aprovechamiento de los recursos. Para lograr esto se requieren dos procedimientos, primero es subdividir la imágen y luego poder enviarla a través de OpenMPI hacia el nodo correspondiente.

Para la división de imágenes se particiona la cantidad de filas por el número de nodos de procesamiento, cuidando que la sumatoria de las particiones sea el número total de filas de la imagen. Junto a esto se debe almacenar la suma de filas anteriores, esto es para calcular la posición relativa en la imágen original, así la primera partición va desde 0 hasta n filas, luego la segunda partición va desde n+1 hasta 2n filas, este rango de filas se utiliza con la funcionalidad de OpenCV para obtener sub imágenes del tamaño requerido. Con esto se obtiene la funcionalidad de subdividir las imágenes.
Para el proceso de envío de datos a través de los nodos se obtiene la representación en bytes de la imágen junto con sus dimensiones y cantidad de canales. Esto permite enviar una cadena de bytes junto a los datos relevantes que permiten reconstruir la imagen en el nodo destino.

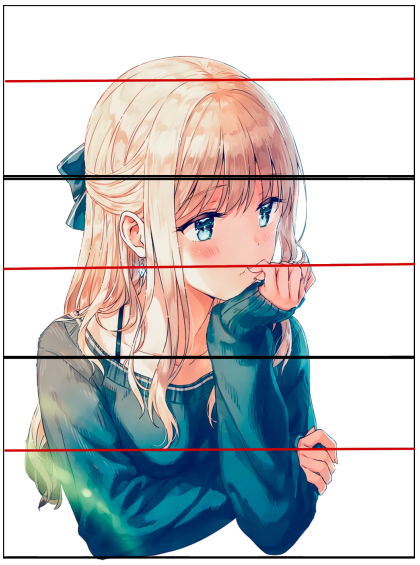
Una vez procesada la imagen en un nodo se necesita enviarla al nodo principal para ser unida, el nodo principal recibe las sub imágenes provenientes de los demás nodos y estas son apiladas una sobre otra usando el orden en que fueron enviadas a los nodos para evitar reconstruir la imagen fuera de orden, para el proceso de reconstrucción o unión de imágenes nuevamente se utilizan las funcionalidades de OpenCV. Finalmente la imagen

generada es guardada en el disco siguiendo el formato definido en el documento del proyecto.

Consideraciones.
Para el caso de escalar y pasar a escala de grises la imagen puede ser enviada y reconstruida sin mayores consideraciones puesto que las operaciones no usan píxeles extra más allá de los provistos.
En el caso del difuminado esto no se cumple, el algoritmo utilizado es la difuminación gaussiana, la cual utiliza una matriz cuadrada de dimensión impar para generar la imagen resultante. El problema radica en los bordes de las imágenes, puesto que si una imagen es subdividida, la partición no tiene información sobre los pixeles siguientes al borde, esto genera que la difuminación no sea la esperada, para solucionar este problema se deben proveer estos píxeles a la imágen con el fin de que la difuminación sea la correcta en los píxeles cerca del borde. Si se considera una matriz de difuminación o Kernel, de 5x5 píxeles, para que el difuminado sea correcto en los bordes de la sub imagen, se deben proveer dos filas de píxeles adicionales, con el fin de que el píxel al centro (el cual está ubicado en el borde de la sub imagen) pueda ser procesado correctamente. Para esto se calcula un offset el cual corresponde a la parte entera de la división de la dimensión del kernel por dos, así si el Kernel es de dimensión 21 se requieren 10 filas extras de píxeles. Este offset se aplica al límite superior e inferior de cada sub imagen, cuidando que en la primera no hay offset superior y en la última sub imagen no hay offset inferior, estos offsets son enviados a través de MPI al nodo correspondiente, una vez realizada la difuminación se eliminan estos píxeles extra y la imagen resultante es enviada al nodo principal donde puede ser regenerada de la misma forma que las imágenes escaladas y pasadas a escalas de grises.

Paralelización.
Una vez que la sub imágen ha sido enviada a un nodo, se debe realizar el procesamiento de esta, para esto se utiliza la misma lógica utilizada para sub dividir las imágenes y enviarlas a los nodos, solo que en este caso la subdivisión se realiza por los procesadores disponibles, cada sub imagen en el nodo luego es procesada de forma paralela y unida de la misma forma dentro del nodo para finalmente ser enviada al nodo principal. Se tienen las mismas consideraciones en el caso de la difuminación para incluir los píxeles extra y luego removerlos para evitar generar distorsiones en la imagen resultante.

Gráficamente el tratamiento del problema es el siguiente.



Donde los recuadros negros indican una subdivision enviada a 3 nodos de procesamiento, y los recuadros rojos indican subdivisiones manejadas por 2 procesadores en cada nodo.

Algoritmos.
Para el difuminado inicialmente se plantea el uso del difuminado Gaussiano, esto debido a que es el que presenta una mejor calidad de imagen resultante, el problema radica en su rendimiento, es por esto que se decide usar una aproximación. Para esto se utiliza una matriz que contiene el mismo escalar en todas sus posiciones,

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

la suma de los elementos será el nuevo píxel central, como todos los elementos de esta matriz son iguales no hace falta volver a calcular los valores ponderados de píxeles cercanos a diferencia de lo que ocurre con una matriz gaussiana donde los elementos de la matriz son diferentes entre sí. Esto abre la posibilidad de optimizar el algoritmo separando las sumas en columnas y utilizarlas cuando se mueve de un pixel al siguiente.
Esto se explica en las siguientes ecuaciones.

$$S[i,j] = \sum_{k=-r}^{+r} C[i, j+k]$$

Donde S[i,j] representa la suma del pixel en posición i,j, la cual es calculada como una suma de columnas.

$$S[i, j+1] = S[i, j] + C[i, j+r+1] - C[i, j-r]$$

La suma de columnas, al momento de mover el algoritmo una posición a la derecha, usa la suma de las columnas anteriores restando la última columna y agregando la nueva.
La siguiente ecuación describe el cálculo del valor de una columna cuando se mueve un píxel abajo.

$$C[i+1, j] = C[i, j] + x[i+r+1, j] - x[i-r, j]$$

Fuente.
https://pdfs.semanticscholar.org/b1e8/bfa7dabbeca901a97c87981540a331bf1162.pdf

Para la escala a grises el algoritmo usado es calcular el promedio de los valores RGB de cada píxel, la imagen resultante corresponde a una imagen de un solo canal.
fuente.
https://docs.opencv.org/2.4/doc/tutorials/introduction/load_save_image/load_save_image.html

Para escalar una imagen, debido a que se elige aumentar el tamaño original se utiliza un algoritmo de escalado de interpolación lineal, esto debido a que es más rápida que la interpolación cúbica y produce resultados aceptables en cuanto a calidad.
Fuente. https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html

Justificación de tecnologías usadas

Para poder realizar este proyecto, se nos solicitó realizarlo en lenguaje C o C++, en este caso se realizó en C++, por experiencia previa del grupo y documentación expuesta en la red, en este caso también se utilizaron las siguientes librerías de C++:

Tecnología	Razón de su uso
OpenMPI	MPI son las siglas de Message Passing Interface. Usada típicamente para computación paralela y distribuida, para la realización de clusters. Se usó para poder distribuir el trabajo entre los nodos a utilizar ya sean PCs físicos o en este caso máquinas virtuales, se utiliza por su facilidad de trabajo y previa experiencia en su uso para problemas similares.

Tecnología	Razón de su uso
OpenMP	Es una librería para paralelizar tareas en computadores con múltiples unidades de procesamiento. Su fácil uso y amplia documentación fueron las razones de utilizar esta tecnología sobre otros métodos para paralelizar procesamientos, como POSIX Threads.

Tecnología	Razón de su uso
OpenCV	Esta librería es bastante completa para poder trabajar con imágenes, en este caso se realizó para manipular imágenes por secciones, difuminar en forma de difusión Gaussiana, el escalar de la imagen y por último transformar la imagen a escala de grises, se utilizó por su amplia documentación y gran capacidad para realizar las tareas previstas en el proyecto.

Conclusión

Gracias al uso de OpenCV para lectura, representación, particionado y transformación de imágenes, el proyecto se logró completar de manera satisfactoria cumpliendo todos los requerimientos propuestos, se logró realizar la distribución y procesamiento paralelo para todas las transformaciones de imágenes de forma correcta sin errores perceptibles en las imágenes resultantes.

Al usar OpenCV para la mayoría de las operaciones sobre imágenes incluidas las tareas solicitadas de difuminado, coloración y tamaño, la implementación usando OpenMPI y OpenMP fue satisfactoria y no presentó mayores complicaciones durante el desarrollo. La implementación realizada es capaz de utilizar todos los computadores dentro de la red y a su vez usar todos los procesadores de cada computador de forma eficiente.