ASIGNATURA:
Computación de Altas Prestaciones

Paralelización de código con MPI + OpenMP

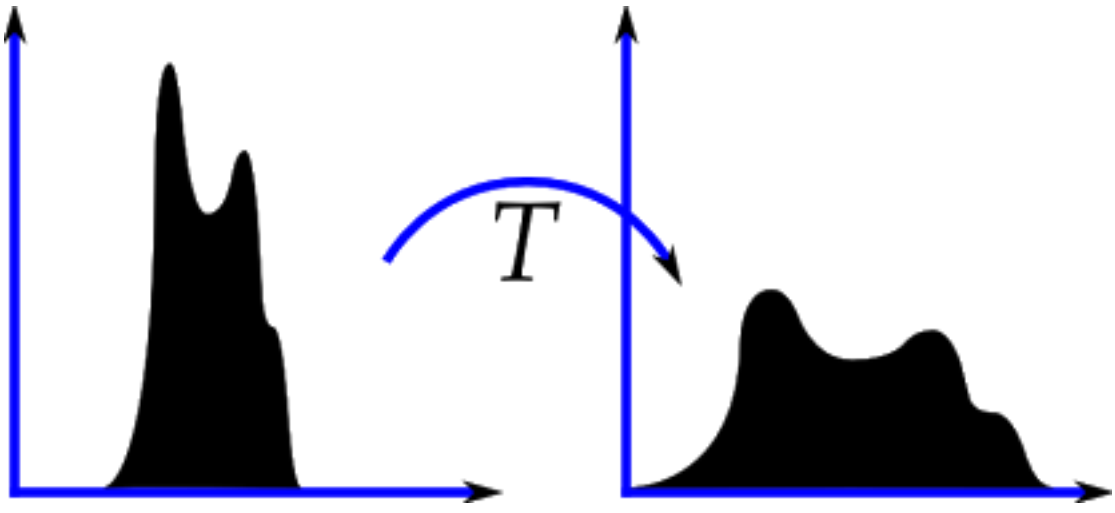
La mejora de contraste es una operación común en el procesamiento de imagen. Es un método útil para el procesamiento de imágenes científicas, tales como imágenes de rayos X o imágenes obtenidas por satélite. También es una técnica útil para mejorar los detalles en las fotografías que están sobre o subexpuestas.

El objetivo de este trabajo es desarrollar una aplicación de mejora de contraste que utiliza la aceleración sobre MPI y OpenMP.

En este documento, se dan las ideas básicas y los principios de mejora de contraste mediante modificación del histograma. Empezamos con el caso simple, realce de contraste para las imágenes en escala de grises. Entonces, tratamos de aplicar el método similar a las imágenes en color.

Una implementación sencilla C se proporciona como referencia y punto de partida.

El histograma de una imagen digital representa su distribución tonal. La ecualización del histograma indica los valores de intensidad más frecuentes. Esto permite conocer las áreas de menor contraste para obtener un mayor contraste sin afectar el contraste global de la imagen. La siguiente imagen muestra el efecto de la ecualización del histograma en el histograma de la imagen.



A continuación, se muestra un ejemplo de ecualización del histograma, la imagen de la izquierda es la imagen original y la inferior es el resultado después de aplicar la ecualización del histograma.

Ecualización de histograma se puede realizar en los siguientes pasos:

- Calcular el histograma de la imagen de entrada;
- Calcular la distribución acumulativa del histograma;
- Utilización de la distribución acumulativa para construir una tabla de búsqueda que mapea cada valor de gris a la ecualizada (este paso se puede combinar con la última);
- Actualización de la imagen utilizando la tabla de búsqueda construida en el último paso.

Requisitos de programación

Se debe implementar un programa paralelo en MPI y OpenMP que realice eficientemente los cálculos. Se proporciona una versión secuencial del programa.

Algunas observaciones sobre la ejecución de la aplicación:

- La aplicación utiliza pgm (para imágenes en escala de grises) y ppm (para imágenes en color). Por defecto, el nombre de archivo del archivo de entrada es *in.pgm* y *in.ppm* respectivamente.
- Dado que el tamaño de las imágenes ppm/pgm es demasiado grande, se dispone de la siguiente imagen en formato JPG. Una vez descargada, será necesario convertirla al formato ppm/pgm (se puede utilizar software convert).
- El ejecutable requiere de los archivos de entrada *in.pgm* y *in.ppm* y genera los archivos procesados *out_hsl.ppm* y *out_yuv.ppm*. No es necesario indicar parámetros de entrada.

El programa se ejecutará de la siguiente manera:

```
srun -p gpus -N 1 -n 1 ./contrast
```

Donde `N` corresponde el número de computadores y `n` con el número de núcleos por computador. Se recomienda el uso de los computadores de la cola *gpus* del laboratorio para los experimentos. Estos cuatro computadores cuentan con 12 núcleos cada uno, 48 en total en forma distribuida.

La versión paralela del programa consistirá de los siguientes pasos:

- Los procesos deben computar una sección reducida para reducir la carga de cómputo de cada proceso.
- Únicamente un proceso (*rank 0* por ejemplo) leerá las imágenes y escribirá los resultados en el disco. Además, un único proceso presentará el tiempo total de la aplicación.
- Se valorará positivamente presentar el código comentado.

Requisitos de entrega

Se deben tomar medidas para 1, 2, 4, 8 y 16 procesos y estudiar el efecto de la paralelización sobre el código inicial sobre las siguientes configuraciones:

- MPI.
- OpenMP.
- MPI + OpenMPI.

Se desea además realizar un estudio del comportamiento del tiempo de ejecución paralelo de los frameworks MPI y OpenMP. Para ello, se realizarán experimentos sobre un número creciente de procesos tanto en memoria compartida como en memoria distribuida.

Entrega

Se debe entregar:

- Una descripción de la metodología de paralelización basada en cuatro fases: descomposición, asignación, orquestación y reparto.
- Gráficas de las medidas de tiempo y aceleración.
- Interpretación de las gráficas.

Pistas

- Usar el comando "convert" para poder generar y visualizar las imágenes procesadas. Por ejemplo

```
convert highres.jpg in.pgm
```

- Para tomar tiempos se recomienda usar la función `MPI_Wtime` () de la siguiente forma:

```
double tstart = MPI_Wtime();
        …
        código
        …
double tfinish = MPI_Wtime();
double TotalTime = tfinish - tstart;
```

- Para asegurarse del correcto funcionamiento de la versión paralela, en comparación con la versión secuencial, seguid los siguientes pasos:
  1. Ejecutar la versión secuencial con un conjunto de datos de entrada.
  2. Ejecutar las diferentes versiones paralelas con el mismo conjunto de datos de entrada que la versión secuencial.
  3. Comparar los conjuntos de datos de salida de cada versión paralela con el obtenido en la versión secuencial usando el comando *diff* (man diff).
  4. Si el comando *diff* detecta diferencias entre el fichero de salida de la versión secuencial y alguno de los ficheros de salida de las versiones paralelas se considerará que el programa no funciona correctamente y, por tanto, habrá que revisar el algoritmo paralelo.

Compilación

```
$ tar zxvf CAP2024.tgz
$ cd CAP2024
$ mkdir build
$ cd ..
$ cmake .. -DCMAKE_CXX_COMPILER=mpicxx.mpich
$ make
```