Práctica de programación orientada al rendimiento

J. Daniel Garcia (coordinador)
Arquitectura de Computadores
Departamento de Informática
Universidad Carlos III de Madrid

2022

1. Objetivo

Est proyecto tiene como objetivo fundamental hacer el que los estudiantes se familiaricen con la **optimización de programas secuenciales**.

En concreto, la práctica se centrará en el desarrollo de software secuencial en el lenguaje de programación C++ (incluyendo las mejoras de C++20).

2. Descripción del proyecto

En este proyecto se construirá una aplicación de procesamiento de imágenes.

En las siguientes secciones se presentan los requisitos que deben cumplir con los programas desarrollados.

2.1. Visión general

Se desarrollará una aplicación que aplica diversas operaciones sobre una imagen en formato BMP. Para cada imagen se implementarán las siguientes operaciones:

- Copia de imágenes.
- Obtención del histograma.
- Conversión a escala de grises.
- Difusión gaussiana (*Gaussian blur*).

La aplicación tomará los siguientes parámetros:

- Directorio con imágenes de entrada.
- Directorio en el que dejar las imágenes de salida.
- Operación a ejecutar.

La operación a ejecutar será una de las siguientes:

- **copy**: No realiza ninguna transformación. Solamente copia los archivos.

- **histo**: Genera un archivo de texto con el histograma de cada canal (RGB).
- **mono**: Convierte a tonos de grises.
- **gauss**: Aplica el filtro de suavizado gaussiano.

NOTA: Se implementarán dos programas distintos **image-aos** e **image-soa**. No obstante, por simplicidad en el resto de esta sección se utiliza el nombre **image** para hacer referencia indistintamente a los dos programas. Para más detalles sobre estos dos programas vaya a la sección 3.1.1.

En cualquiera de los casos, la aplicación leerá todos los archivos del directorio de entrada, aplicará las transformaciones correspondientes y escribirá los archivos correspondientes con el mismo nombre en el directorio de salida.

El programa deberá comprobar que el número de parámetros es 3:

```
$image
Wrong format:
  image in_path out_path oper
    operation: copy, histo, mono, gauss
$
```

Si el tercer argumento no toma un valor adecuado (**copy**, **histo**, **mono** o **gauss**), se presentará un mensaje de error y se terminará:

```
$image indir outdir gaus
Unexpected operation:gaus
  image in_path out_path oper
    operation: copy, histo, mono, gauss
$
```

Si el directorio de entrada no existe o no puede abrirse, se presentará un mensaje de error y se terminará:

```
$image inx outdir copy
Input path: inx
Output path: out
Cannot open directory [inx]
  image in_path out_path oper
    operation: copy, histo, mono, gauss
$
```

Si el directorio de salida no existe, se presentará un mensaje de error y se terminará:

```
$image-seq sobel indir outx
Input path: indir
Output path: outx
Output directory [outx] does not exist
  image in_path out_path oper
    operation: copy, histo, mono, gauss
$
```

En cualquier otro caso, se procesarán todos los archivos del directorio de entrada y se dejarán los resultados en el directorio de salida. Para cada archivo procesado se imprimirá la información del tiempo dedicado a cada tarea. Todos los valores de tiempo se expresarán en microsegundos:

```
$image-seq indir outdir gauss
Input path: indir
Output path: outdir
File: "indir/62096.bmp"(time: 43994)
  Load time: 3442
  Gauss time: 38909
  Store time: 1642
File: "indir/169012.bmp"(time: 37209)
  Load time: 2602
```

```
  Gauss time: 32950
  Store time: 1655
$
```

2.2. El formato BMP

Se utilizará el formato de imágenes BMP. Los ficheros de este tipo contienen una cabecera de al menos 54 bytes con la información especificada en el cuadro 1.

Comienzo	Longitud	Descripción
0	2	Caracteres **'B'** y **'M'**.
2	4	Tamaño del archivo.
6	4	Reservado.
10	4	Inicio de datos de imagen.
14	4	Tamaño de la cabecera de bitmap.
18	4	Anchura en pixeles.
22	4	Altura en pixeles.
26	2	Número de planos.
28	2	Tamaño de punto en bits.
30	4	Compresión.
34	4	Tamaño de la imagen.
38	4	Resolución horizontal.
42	4	Resolución vertical.
46	4	Tamaño de la tabla de color.
50	4	Contador de colores importantes.

Cuadro 1: Campos de cabecera para el formato BMP.

Se tendrán en cuenta las siguientes restricciones para que un fichero se pueda considerar válido.

- El número de planos debe ser **1**.
- El tamaño de cada punto debe ser de **24** bits.
- El valor de compresión debe ser **0**.

Obsérvese que no será necesario utilizar ciertos campos.

Los pixeles se almacenan a partir de la posición indicada por el campo *Inicio de datos de imagen*. Cada pixel se representa por 3 bytes consecutivos que representan el nivel de azul, verde y rojo para ese pixel. Dichos valores estarán siempre en el rango de **0** a **255**.

Al finalizar los píxeles de una fila se dejarán bytes de relleno, de forma que la siguiente fila de píxeles comience al principio de una palabra.

2.3. Histograma de una imagen

Por cada imagen se generará un archivo de texto con extensión **.hst** que contendrá consecutivamente la información de los tres histogramas. Esto es, se generará un histograma para el canal R, otro histograma para el canal G, y un tercer histograma para el canal B.

Un histograma representa la frecuencia absoluta (número de apariciones) de cada valor numérico (de 0 a 255).

El archivo de salida será un archivo de texto que contendrá un total de 768 (256×3) líneas. Las primeras 256 líneas contendrán los valores de frecuencia absoluta de cada valor para la componente R. Las siguientes 256 líneas serán para la componente G. Las últimas 256 líneas serán para la componente B.

2.4. Conversión a escala de grises

Una imagen en escala de grises tiene el mismo valor para los tres valores RGB del pixel.
La transformación se realiza en cuatro pasos:

1. Normalización: Se normalizan los valores a una escala real de **0** to **1**.

$$n_r = \frac{R}{255}$$

$$n_g = \frac{G}{255}$$

$$n_b = \frac{B}{255}$$

2. Transformación a intensidad lineal: Para cada color normalizado (n_r, n_g, n_b) se obtiene un nuevo color (c_r, c_g, c_b). Para ellos se aplica la misma transformación en los tres canales:

$$c_i = \frac{n_i}{12{,}92} \qquad \text{si} \quad n_i \leq 0{,}04045$$

$$c_i = \left(\frac{n_i + 0{,}055}{1{,}055}\right)^{2{,}4} \qquad \text{si} \quad n_i > 0{,}04045$$

3. Transformación lineal: Se aplica una transformación lineal a cada color.

$$c_l = 0{,}2126 \cdot c_r + 0{,}7152 \cdot c_g + 0{,}0722 \cdot c_b$$

4. Corrección gamma:

$$c_g = 12{,}92 \cdot c_l \qquad \text{si} \quad c_l \leq 0{,}0031308$$

$$c_g = 1{,}055 \cdot c_l^{\frac{1}{2{,}4}} - 0{,}055 \qquad \text{si} \quad c_l > 0{,}0031308$$

5. Desnormalización: Se transforma de la escala $[0, 1]$ a la escala $[0, 255]$

$$g = c_g \cdot 255$$

NOTA: Tenga en cuenta que debe almacenarse el valor obtenido g, para las tres componentes (R, G y B) del pixel.

2.5. Difusión gaussiana

La difusión gaussiana tiene por objeto disminuir la nitidez de una imagen y se utiliza como paso previo a la detección de bordes.

En general, se utiliza una matriz cuadrada que actúa como máscara. Para una máscara de 5×5 la imagen resultado res, se obtiene a partir de la máscara m y de la imagen original im aplicando la siguiente expresión:

$$res(i,j) = \frac{1}{w} \sum_{s=-2}^{2} \sum_{t=-2}^{2} m(s+3, t+3) \cdot im(i+s, j+t)$$

donde la matriz de la máscara m y el peso w son:

$$m = \begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix} \qquad w = 273$$

Importante: Tenga en cuenta que al aplicar la máscara en los bordes de la imagen, se considerará que la matriz de la imagen está rodeada de pixeles imaginarios de valor **0**.

3. Tareas

3.1. Desarrollo del software

3.1.1. Versiones del programa

Esta tarea consiste en el desarrollo de la versión secuencial de la aplicación descrita en C++20. Se prepararán dos programas ejecutables: **image-soa** y **image-aos** que implementarán dos estrategias:

- **SOA – Structure of Arrays**: Se representará una imagen mediante tres vectores independientes de valores entre **0** y **255**.
- **AOS – Array of Structures**: Se representará una imagen con un único vector de valores de tipos pixel. Un pixel representa una estructura con tres valores entre **0** y **255**.

3.1.2. Componentes

Deberá desarrollar los siguientes componentes:

- **common**: Biblioteca con los archivos fuente comunes a las dos versiones del programa.
- **soa**: Biblioteca con los archivos fuente específicos a loscomponentes de la versión **AOS**.
- **aos**: Biblioteca con los archivos fuente específicos a loscomponentes de la versión **AOS**.
- **image-soa**: Ejecutable con el programa para la versión **SOA**.
- **image-aos**: Ejecutable con el programa para la versión **AOS**.

A continuación se describe con más detalle cada uno de los cinco componentes mencionados:

common : Componentes comunes.
Contendrá como mínimo los siguientes archivos fuente:

- **progargs.cpp**: Tratamiento de los argumentos del programa recibidos por **main**.

soa : Biblioteca de componentes **SOA**.
Contendrá como mínimo los siguientes archivos fuente:

- **imagesoa.cpp**: Tratamiento de imágenes con una representación **SOA**.

aos : Biblioteca de componentes **AOS**.
Contendrá como mínimo los siguientes archivos fuente:

- **imageaos.cpp**: Tratamiento de imágenes con una representación **AOS**.

image-soa : Programa para la versión **SOA**.
Contendrá un único archivo fuente:

- **imgsoa**: Contendrá exclusivamente la función **main()** para esta versión. No podrá incluir ninguna función ni tipo adicional.

image-aos : Programa para la versión **AOS**.
Contendrá un único archivo fuente:

- **imgaos**: Contendrá exclusivamente la función **main()** para esta versión. No podrá incluir ninguna función ni tipo adicional.

3.1.3. Compilación del proyecto

Todos sus archivos fuente deben compilar sin problemas y no deben emitir ninguna advertencia del compilador.

Deberá incluirse un archivo de configuración CMake con las siguientes opciones:

CmakeLists.txt

```
cmake_minimum_required(VERSION 3.23)
project(image LANGUAGES CXX)

set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_CXX_EXTENSIONS_OFF)
add_compile_options(-Wall -Wextra -Werror -pedantic -pedantic-errors)

set(CMAKE_CXX_FLAGS_RELEASE "-march=native")

add_library(common file1-common.cpp file2-common.cpp file3-common.cpp)
add_library(aos file1-aos.cpp file2-aos.cpp)
add_library(soa file1-soa.cpp file2-soa.cpp)

add_executable(image-aos imgaos.cpp)
target_link_libraries(imgaos common aos)
add_executable(imgsoa image-soa.cpp)
target_link_libraries(imgsoa common soa)
```

Tenga también en cuenta que deberá realizar todas las evaluaciones con las optimizaciones del compilador activadas con la opción de CMake **-DCMAKE_BUILD_TYPE=Release**.

Importante: No se podrá utilizar ninguna biblioteca externa a la biblioteca estándar de C++.

3.1.4. Normas de cálidad del código

El código fuente debe estar bien estructurado y organizado, así como apropiadamente documentado. Se recomienda, pero no es de obligado cumplimiento, hacer uso de las **C++ Core Guidelines** (`http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines`).

En cualquier caso, las siguientes reglas si son de obligado y estricto cumplimiento:

- No se podrán utilizar variables globales que no sean constantes.
- No está permitido el uso del patrón *Singleton*.
- No se podrá pasar arrays a una función como parámetros de tipo puntero.
- Ninguna función ni función miembro podrá tener más de cuatro parámetros.
- Ninguna función podrá tener más de 25 líneas estando apropiadamente formateadas.
- Todos los parámetros se pasarán a las funciones por valor, por referencia o por referencia constante.
- No se podrá usar explicitamente las funciones **malloc()** o **free()** ni los operadores **new** o **delete**.
- No se permite el uso de macros, excepto para la definición de guardas de inclusión.
- No se permite ningún cast excepto **static_cast**, **dynamic_cast**, **const_cast** o **reinterpret_cast**.

3.1.5. Pruebas unitarias

Debe definir y entregar un conjunto de pruebas unitarias. Se recomienda el uso de GoogleTest (`https://github.com/google/googletest`), aunque no es imprescindible.

En cualquier caso debe aportar evidencias de haber realizado suficientes pruebas unitarias.

3.2. Evaluación del rendimiento y energía

Esta tarea consiste en realizar una evaluación comparativa del rendimiento de las dos estrategias de implementación **image-aos** e **image-soa**.

Para evaluar el rendimiento debe medir el tiempo de ejecución de la aplicación. También debe medirse el consumo energético de la aplicación y derivar el uso de potencia.

Todas las evaluaciones de rendimiento se realizarán en un nodo del clúster **avignon**.

Represente en una gráfica todos los tiempos totales de ejecución, uso de energía y potencia obtenidos para imágenes de distinto tamaño. A tal efecto se suministrarán imágenes para la evaluación.

Incluya en la memoria de esta práctica las conclusiones que pueda inferir de los resultados. No se limite simplemente a describir los datos. Debe buscar también una explicación convincente de los resultados.

4. Calificación

La puntuación final obtenida en este proyecto se obtiene teniendo en cuenta el siguiente reparto:

- Rendimiento alcanzado: 20 %.
- Consumo energético alcanzado: 20 %.

- Pruebas unitarias: 10 %.
- Puebas funcionales: 5 %.
- Calidad del diseño: 5 %.
- Calidad del código: 10 %.
- Evaluación del rendimiento y energía en la memoria: 20 %.
- Organización del trabajo: 5 %.
- Conclusiones: 5 %.

Advertencias:

- Si el código entregado no compila, la nota final de la práctica será de 0.
- Si se ignora injustificadamente alguna norma de calidad de código, la nota final de la práctica será de 0.
- En caso de copia todos los grupos implicados obtendrán una nota de 0. Además se notificará a la dirección de la EPS para la correspondiente apertura de expediente disciplinario.

5. Procedimiento de entrega

La entrega del proyecto se realizará a través de Aula Global.
Para ello se habilitarán 2 entregadores separados:

- **Entregador de memoria**. Contendrá la memoria del proyecto, que será un archivo en formato pdf con el nombre **memoria.pdf**.
- **Entregador de código**: Contendrá todo el código fuente necesario para compilar la versión secuencial.
  - Debe ser un archivo comprimido (formato zip) con el nombre **image.zip**.

La memoria no deberá exceder de 15 páginas con una fuente mínima de 10 puntos incluyendo la portada y todas las secciones. no se tendrá en cuenta en la corrección los contenidos a partir de la página 16 si fuese el caso. deberá contener, al menos, las siguientes secciones:

- **Página de título**: contendrá los siguientes datos:
  - Nombre de la práctica.
  - Nombre del grupo reducido en el que están matriculados los estudiantes.
  - Número de equipo asignado.
  - Nombre y nia de todos los autores.
- **Diseño original**. Debe incluir el diseño de cada uno de los componentes en que se estructuran las bibliotecas **common**, **aos**, **soa**. en esta sección se deben explicar y justificar cada una de las principales decisiones de diseño tomadas.

- **Optimización**. Debe contener una discusión de las optimizaciones aplicadas y su impacto. En particular, deberán indicarse optimizaciones realizadas sobre el código fuente original, así como optimizaciones activadas con flags de compilación adicionales que se estime oportuno.
- **Pruebas realizadas**: Descripción del plan de pruebas realizadas para asegurar la correcta ejecución. Debe incluir pruebas unitarias, así como pruebas funcionales de sistema.
- **Evaluación de rendimiento y energía**: Deberá incluir las evaluaciones de rendimiento y energía llevadas a cabo.
- **Organización del trabajo**: Deberá describir la organización del trabajo entre los miembros del equipo haciendo explícita las tareas llevadas a cabo por cada persona. Debe incluir el tiempo dedicado por cada persona a cada tarea. No se podrá asignar más de una persona a una tarea.
- **Conclusiones**. Se valorará especialmente las derivadas de los resultados de la evaluación del rendimiento, así como las que relacionen el trabajo realizado con el contenido de la asignatura.