Asignación de Prácticas Número 12
Programación Concurrente y de Tiempo Real

Kevin J. Valle Gómez

Departamento de Ingeniería Informática
Universidad de Cádiz

2020

Contenido

Objetivos de la práctica

- Aplicar los conocimientos adquiridos en las sesiones prácticas hasta ahora.
  - Comprender la diferencia entre Java y C++ para programar soluciones paralelas.
  - Someter los programas a un análisis de rendimiento.
- Adaptar soluciones escritas en Java a C++.
- Trabajar sobre la convolución de una matriz de datos.

¿C++?

- La programación multihilo no es exclusiva del lenguaje Java.
- En esta practica desarrollaremos una solución a un problema usando el lenguaje C++.
  - Para lo cual será conveniente recordar las diferencias.

Recursos del lenguaje - C++

- Biblioteca para la gestión de hebras concurrentes: `<thread>`.
- La creación y ejecución de hebras se hace como objetos de la clase `thread`
- Proporciona sincronización de hebras mediante cerrojos de la clase `mutex`, variables atómicas, monitores con semántica de señalización SC y variables de condición.

Antes de seguir, trabaja el seminario sobre programación concurrente en C++ de esta asignatura.

Convolución de matrices: `convolucion.pdf`

MSEL
Modelling in Science Education and Learning

Modelling in Science Education and Learning
Volume 9(2), 2016 DOI: 10.4995/msel.2016.4524.
Instituto Universitario de Matemática Pura y Aplicada
Universitat Politècnica de València

Aplicación de la convolución de matrices al filtrado de imágenes

Application to convolution of matrices to image filtering

F. Giménez-Palomares, J. A. Monsoriu, E. Alemany-Martínez
Universitat Politècnica de València
fgimenez@mat.upv.es, jmonsori@fis.upv.es, ealemany@mat.upv.es

Abstract

Aclaración importante

Desde esta diapositiva, todas las fuentes y definiciones están sacadas de la referencia [1].

[1] Palomares, F. G., Serrá, J. A. M., & Martínez, E. A. (2016). Aplicación de la convolución de matrices al filtrado de imágenes. Modelling in Science Education and Learning, 9(1), 97-108.

Convolución

- Convolución: ¿en qué consiste?
  - *"Operación matemática de dos funciones, que produce una tercera función considerada como una versión modificada de una de las dos anteriores."*
  - *"Técnica del tratamiento de imágenes en la cual cada pixel es alterado en función de los píxeles circundantes."*
- A partir de una imagen, podemos obtener otra modificada.
- Es posible resaltar o suprimir, de manera selectiva, la información de una imagen.

Convolución - Tipos de filtro

Filtros de paso bajo:

- Suavizado de imagen
- Eliminación de ruido
- Ejemplo: Desenfoque

Filtros de paso alto:

- Resalto de zonas de mayor variabilidad
- Sustracción de la media
- Ejemplo: Enfoque

Filtros direccionales:

- Resalto de píxeles que determinan direcciones

Filtros para la detección de bordes:

- Aplicación en ingeniería, estudio del terreno
- Ejemplo: Detectar bordes

Figure 1: Tipos de filtro [1]

Convolución de matrices

- Se aplica al reconocimiento de imágenes.
- Una imagen puede interpretarse como una función bidimensional
  - $z = F(x, y)$
  - x e y son las coordenadas espaciales.
  - z el valor de la intensidad de la imagen en el punto (x, y)

Convolución de matrices (II)

- Una imagen digital en escala de grises viene dada a partir de la matrix
  - $z_ij = F(x_i, y_j)$
- Cada entrada de la matriz es un pixel.
- El número de pixels (el orden de la matriz) define la resolución de la imagen.
- Habitualmente, el rango de intensidades varía de 0 (negro) a 255 (blanco).

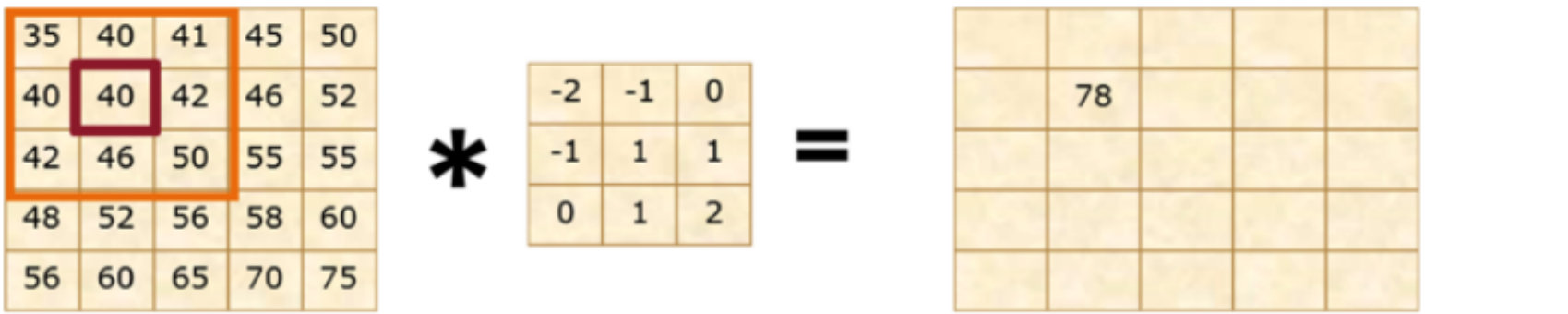
Convolución de matrices (III)

- El filtrado de imágenes consiste en la modificación de las matrices correspondientes a esta. Un procedimiento habitual para ello es la *convolución de matrices*.
- **Definición:** Dada una matriz A_{m*n} y una matriz $C_{(2N+1)*(2N+1)}$ con $2N+1 < m, n$ se define la convolución de las matrices A y C como una nueva matriz $D = A * C$ definida a partir de la expresion de la siguiente página [1].
- La matriz C se denomina *núcleo* o *kernel* de la función.

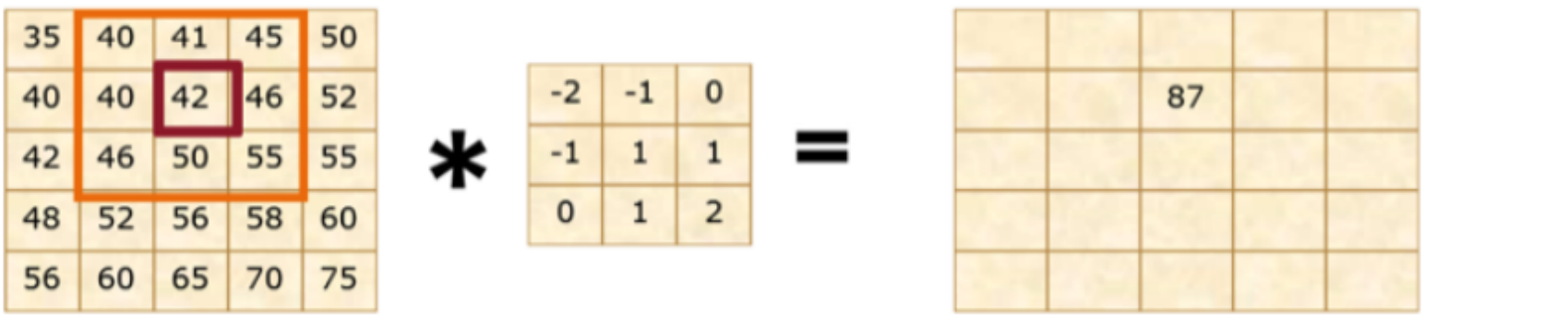
Convolución de matrices - Ecuación

$$d_{ij} = \frac{1}{c} \sum_{r=1}^{2N+1} \sum_{s=1}^{2N+1} a_{i-N+r-1,i-N+r-1} cr, s,$$

Proceso de convolución de matrices



$35 \cdot (-2) + 40 \cdot (-1) + 41 \cdot 0 + 40 \cdot (-1) + 40 \cdot 1 + 42 \cdot 1 + 42 \cdot 0 + 46 \cdot 1 + 50 \cdot 2 = 78$



$40 \cdot (-2) + 41 \cdot (-1) + 45 \cdot 0 + 40 \cdot (-1) + 42 \cdot 1 + 46 \cdot 1 + 46 \cdot 0 + 50 \cdot 1 + 55 \cdot 2 = 87$

Proceso de convolución de matrices (II)

35	40	41	45	50
40	40	42	46	52
42	46	50	55	55
48	52	56	58	60
56	60	65	70	75

$*$

-2	-1	0
-1	1	1
0	1	2

$=$

35	40	41	45	50
40	78	87	94	52
42	98	283	108	55
48	120	125	127	60
56	60	65	70	75

¿Y qué ocurre con los bordes?

Tenemos tres opciones:

1. Completar con ceros los valores de alrededor.
2. Repetir los valores del borde.
3. Completar con los valores de la parte simétrica opuesta.

Recuerda lo siguiente:

- Algunos filtros usados en el tratamiento de la imagen usan un valor c arbitrario en la ecuación.

Ejercicio 1 - Aplicación de los conocimientos

Se pide:

- Calcula la convolución de una matriz de datos de 10^3 filas y columnas.
- La matriz se cargará con números enteros **aleatorios entre 0 y 255**.

Debe leer atentamente el documento `convolucion.pdf`. Estas diapositivas solo son una ayuda para comprender el concepto.

Ejercicio 1 - Convolución secuencial

- Implementa una solución para la convolución de matrices.
  - En Java y C++
- El kernel será elegido por el usuario en un menú de opciones:
  - Enfocar.
  - Realzar bordes.
  - Detectar bordes.
  - *Sobel*.
  - *Sharpen*.

Ejercicio 1 - Kernels de convolución

Enfoque

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Desenfoque

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Realce de bordes

$$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Repujado

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Detección de bordes

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Filtro de tipo Sobel

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Filtro de tipo Sharpen

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Filtro Norte

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

Filtro Este

$$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$$

Filtro de tipo Gauss

$$\begin{bmatrix} 1 & 2 & 3 & 1 & 1 \\ 2 & 7 & 11 & 7 & 2 \\ 3 & 11 & 17 & 11 & 3 \\ 2 & 7 & 11 & 7 & 1 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

Ejercicio 1 - Consejo

Ten en cuenta...

- La matriz se carga con números enteros aleatorios entre 0 y 255.
- Los kernels de convolución son fijos: no cambian.
- Será útil implementar un método que reciba la matriz de datos y la matriz del kernel, devolviendo la matriz resultante del cálculo de la convolución.
- El usuario introducirá por pantalla, a través de un menú, el kernel de convolución.

Ejercicio 1 - Implementación en Java

Es sencillo: aplica todo lo visto sobre generación de números aleatorios y paso de parámetros por desde la línea de comandos ;-)

Ejercicio 1 - Implementación en C++

Puedes partir de estos dos ejemplos:

- `parlincomandoscpp.cpp`: paso de parámetros desde la línea de comandos.
  - `scanf("%d", &entero);`
- `aleatorioscpp.cpp` y `naleatoriosc++.cpp`: generación de números aleatorios.
  - `(1 + rand() % 20); //mostramos por pantalla los numeros generados del 1 al 20`

Ejercicio 1 - Segunda parte

Se pide:

- Escribe versiones paralelas de los códigos anteriores.
- Utiliza el paralelismo mediante la **división automática del dominio de datos**.
- El número de tareas paralelas se leerá desde la línea de comandos.

Ejercicio 1 - Implementación en Java (segunda parte)

Partiendo de la solución secuencial, transforma el código para que el cálculo sea paralelo. Ya conoces los mecanismos para ello.

Ejercicio 1 - Implementación en C++ (segunda parte)

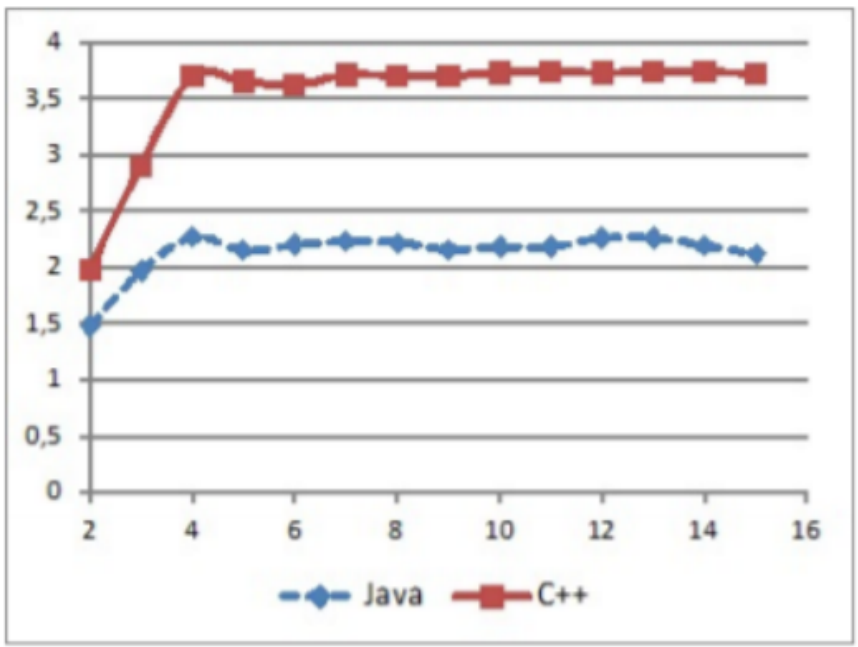
- `hebrasCargaParametrica.cpp`: ejemplo sobre cómo pasar funciones con parámetros al constructor de hebras en C++.
  - Abre el fichero en el editor y comprueba como se crean y ejecutan las hebras.

Ejercicio 2 - Rendimiento de las soluciones paralelas

- Se desea comparar el desempeño de ambos lenguajes para resolver el problema de forma paralela.
- Desarrolla dos curvas:
  - *Speedup = f(tareas)*: representa el rendimiento en proporción del número de tareas.
  - *Tiempo = f(tareas)*: representa el tiempo de ejecución en función del número de tareas.

Ejercicio 2 - Ejemplo de curva

NOTA: **La curva de dicha figura se presenta a modo de ejemplo de cómo integrar los datos, y no presuponc nada accrca dc los rcsultados que usted debe obtener.**



Ejercicio 2 - Toma de tiempos

- Java: ya conoces los métodos para tomar el tiempo, utilízalos.
- C++:
  - `medtemporal.cpp`: cómo tomar tiempos en C++
  - Abre el fichero en el editor y comprueba cómo se toma el tiempo y se muestra por pantalla.

Ejercicio 2 - ¿Qué debo hacer?

- Desarrolla dos gráficas: una sobre Linux y otra sobre Windows.
- Analiza los resultados:
  - Comprueba la diferencias en ambos casos.
  - Justifícalos en el contexto de lo aprendido en la asignatura.
- Guarda las curvas junto con la explicación y justificación en un fichero `ConvolucionAnalisis.pdf`.

Referencias


F. G. Palomares, J. A. M. Serrá y E. A. Martínez
Aplicación de la convolución de matrices al filtrado de imágenes.
Modelling in Science Education and Learning, 9(1), 97-108..