Proyecto 1 - Sistema distribuido

Informe Técnico

Fecha: 7 de octubre de 2026

Integrantes:
Javier Gamboa
Rafael Gonzalez
Victor Cornejo

Contents

1 Introducción

La morfología matemática es una técnica de procesamiento de imágenes ampliamente utilizada para la eliminación de ruido y la mejora de estructuras en fotografías digitales. Entre sus operaciones fundamentales se encuentran la **erosión**, que tiende a eliminar los detalles claros reduciendo regiones brillantes, y la **dilatación**, que expande estas regiones y elimina detalles oscuros.

En este proyecto se implementó un sistema en **Java** que permite aplicar ambas operaciones sobre imágenes en formato **PNG** y en modelo de color **RGB**. La solución fue desarrollada en dos variantes: una **secuencial** y otra **paralela**, con el objetivo de analizar las diferencias de rendimiento y validar los resultados generados bajo distintos elementos estructurantes.

2 Objetivos

- Implementar en Java las operaciones de erosión y dilatación en imágenes a color.
- Diseñar una solución **secuencial** que sirva como referencia para validar la corrección de la versión paralela.
- Desarrollar una implementación **paralela** mediante división de la imagen en submatrices (tiles) utilizando múltiples hilos.
- Comparar el rendimiento entre las versiones secuencial y paralela utilizando distintos tamaños de imagen y elementos estructurantes.
- Elaborar un informe técnico que documente el proceso, las decisiones de diseño y los resultados experimentales obtenidos.

3 Desarrollo

3.1 Estructura del proyecto

El proyecto se organizó siguiendo la convención de Maven, con los siguientes módulos principales:

- **core**: contiene las implementaciones de morfología secuencial y paralela, además del módulo de benchmark.
- **model**: define las estructuras de datos utilizadas como la política de borde, el tipo de operación y los elementos estructurantes.
- **util**: funciones auxiliares para la lectura/escritura de imágenes y medición de tiempos.
- **menu**: controlador de la aplicación, encargado de recibir parámetros de ejecución o interactuar con el usuario mediante consola.

3.2 Implementación secuencial

La versión secuencial recorre cada píxel de la imagen y aplica el elemento estructurante sobre sus vecinos, reemplazando el valor del píxel central por el mínimo o máximo dependiendo de la operación (erosión o dilatación). El manejo de bordes se realizó bajo dos políticas:

- **Ignore**: se ignoran los vecinos que caen fuera de los límites de la imagen.
- **Pad**: se rellenan los valores fuera de rango con un valor neutro (255 en erosión, 0 en dilatación).

3.3 Implementación paralela

Para la versión paralela se dividió la imagen en **submatrices horizontales** o tiles. Cada tile incluye un **halo** adicional para evitar inconsistencias en los bordes al aplicar el elemento estructurante. La paralelización se implementó mediante un `ExecutorService` en Java, asignando a cada hilo el procesamiento de un bloque independiente.

Esta solución puede clasificarse en la **arquitectura MIMD** de Flynn, ya que diferentes hilos ejecutan instrucciones distintas sobre distintos datos.

3.4 Elementos estructurantes

Se implementaron cinco elementos estructurantes:

1. Cuadrado 3x3
2. Cruz 3x3
3. X 3x3
4. Línea horizontal 1x3
5. Diamante 5x5

3.5 Resultados experimentales

Se evaluaron distintas imágenes con tamaños de hasta 10.000 x 10.000 píxeles. Los experimentos incluyeron mediciones de tiempo para:

- **Erosión vs. Dilatación**
- **Secuencial vs. Paralelo** con 2, 4, 8 y 16 hilos.
- Distintos elementos estructurantes.

Los tiempos fueron registrados únicamente durante la fase de cómputo, excluyendo lectura y escritura de archivos. Para cada caso se realizaron tres ejecuciones, almacenando el promedio y desviación estándar.

4 Conclusiones

El desarrollo de este proyecto permitió comprender en detalle cómo la **paralelización** mejora el rendimiento en operaciones intensivas de procesamiento de imágenes. Las principales conclusiones fueron:

- La implementación paralela mostró mejoras significativas en imágenes grandes (superiores a 2000 x 2000 píxeles).
- El tiempo de ejecución decrece al aumentar los hilos, aunque la ganancia se estabiliza a partir de un número similar al de núcleos físicos de la CPU.
- La política de borde **Pad** asegura mayor consistencia visual en los resultados, mientras que **Ignore** es más eficiente en tiempo.
- El uso de un diseño modular permitió mantener la equivalencia entre los resultados de la versión secuencial y paralela.