UNIVERSIDAD DE COSTA RICA

EIE

Escuela de
Ingeniería Eléctrica

Implementación en paralelo del algoritmo de segmentación de imágenes, *Watershed Transform*, para sistemas de memoria compartida y distribuida

Mediante el uso de las herramientas *OpenMP* y *MPI*.

Jose E. Flores
B82994

Profesor:
Elvis Rojas

SP-2136: Programación Avanzada
II-24
4 de Octubre del 2024

1 Título

Implementación en paralelo del algoritmo de segmentaci´on de im´agenes, *Watershed Transform*, para sistemas de memoria compartida y distribuida. Mediante el uso de las herramientas *OpenMP* y *MPI*.

2 Resumen

Para el proyecto final del curso *SP-2136: Programación Avanzada*, se propone una implementación en paralelo del algoritmo de segmentación de imágenes *Watershed Transform* utilizando las herramientas *OpenMP* y *MPI*. El objetivo es optimizar el rendimiento del algoritmo en sistemas de memoria compartida y distribuida, reduciendo los tiempos de procesamiento mediante la paralelización de sus componentes clave.

3 Descripción General del Problema

La transformación *Watershed* es un algoritmo ampliamente utilizado en segmentación de imágenes, especialmente en procesamiento de imágenes digitales y visión por computadora. Su objetivo principal es dividir una imagen en regiones homogéneas basadas en criterios como la intensidad de píxeles o gradientes. Sin embargo, la implementación secuencial del algoritmo puede ser computacionalmente costosa, especialmente para imágenes de alta resolución o grandes conjuntos de datos. El problema que abordaremos es la optimización de la transformación *Watershed* mediante el uso de paralelismo, con el fin de acelerar el proceso de segmentación y hacer viable su aplicación en tiempo real o en sistemas con recursos limitados.

4 Justificación del Uso de Paralelismo

La necesidad de implementar paralelismo en este problema resulta de la complejidad computacional inherente del algoritmo *Watershed*. Dado que cada píxel de la imagen puede influir en la segmentación final, el algoritmo requiere un gran número de operaciones, lo que resulta en tiempos de procesamiento prolongados. Al paralelizar el algoritmo, es posible distribuir las cargas de trabajo entre múltiples núcleos o máquinas, reduciendo significativamente el tiempo de ejecución. Esto es crucial en aplicaciones prácticas donde la rapidez y eficiencia son esenciales, como en procesamiento de video en tiempo real, análisis médico o sistemas autónomos. Sin embargo, este algoritmo presenta desafíos a la hora de realizar la parelelización debido a la dependencia de computaciones pasadas. Es uno de los objetivos de este proyecto detectar áreas en las cuales esto no sea un problema, y que resulte en un aumento de velocidad significativo con respecto a las implementaciones secuenciales del mismo.

5 Funcionalidad y Algoritmos a Paralelizar

Se planea paralelizar las siguientes componentes clave del algoritmo *Watershed*:

- **Computación del Gradiente**: Cálculo paralelo de los gradientes de intensidad en la imagen para resaltar las fronteras entre regiones.
- **Inundación Simultánea**: Implementación de un proceso paralelo de inundación desde múltiples mínimos locales para construir los *catchment basins*.
- **Actualización de Etiquetas**: Asignación concurrente de etiquetas a regiones segmentadas, evitando condiciones de carrera mediante sincronización adecuada.
- **Integración de Resultados**: Combinación eficiente de los resultados parciales obtenidos en los diferentes hilos o nodos de procesamiento.

6 Bibliotecas y Lenguajes de Programación Paralela a Utilizar

Para la implementación del paralelismo se consideran las siguientes opciones:

- **Memoria Compartida**:
  - **OpenMP**: Para paralelismo a nivel de CPU en sistemas multinúcleo, facilitando la creación de hilos y el manejo de bucles paralelos.
- **Memoria Distribuida**:
  - **MPI (Message Passing Interface)**: Para distribuir el procesamiento entre múltiples máquinas en un clúster, gestionando la comunicación entre nodos.
- **Lenguajes de Programación**:
  - **C/C++**: Por su eficiencia y amplio soporte de bibliotecas como OpenMP y MPI.
  - **Python con NumPy y MPI4Py (Opcional)**: Para realizar una comparación entre la parelización del algoritmo en lenguaje de bajo y alto nivel.

La elección final dependerá de los recursos disponibles y el balance entre complejidad de implementación y rendimiento obtenido. Existe la posibilidad de combinar memoria compartida y distribuida.

References

[1] The watershed transform: Definitions, algorithms and parallelization strategies. *Fundam. Informaticae*, 41:187–228, 2000.

[2] Yosra Braham, Y. Elloumi, M. Akil, and M. Hedi. Parallel computation of watershed transform in weighted graphs on shared memory machines. *Journal of Real-Time Image Processing*, 17:527–542, 2018.

[3] V. Grau, A. Mewes, M. A. Raya, R. Kikinis, and S. Warfield. Improved watershed transform for medical image segmentation using prior information. *IEEE Transactions on Medical Imaging*, 23:447–458, 2004.

[4] A. Kornilov, I. Safonov, and I. Yakimchuk. A review of watershed implementations for segmentation of volumetric images. *Journal of Imaging*, 8(5):127, April 2022.

[5] Alina N. Moga and M. Gabbouj. Parallel image component labeling with watershed transformation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:441–450, 1997.