ALGORITMIA – Grado en Ingeniería en Informática del Software 2022

GUIÓN DE PRÁCTICAS. SESIÓN 6

OBJETIVOS:

- **Vuelta atrás (Backtracking): reconstrucción de imágenes a alta resolución.**

1. Reconstrucción de imágenes de alta resolución para crio-microscopía electrónica

La crio-microscopía electrónica (cryo-EM) es una técnica de adquisición y procesado de imágenes que ha conseguido resolver la estructura atómica de muchas proteínas incluyendo la proteína *spike* del virus COVID19, lo que ha acelerado enormemente el desarrollo de vacunas. El éxito de cryo-EM se basa en la aplicación de algoritmos de computación, como prueba Joachim Frank (Alemania 1940) fue uno de los galardonados con el Nobel de Química en 2017 (compartido con otros dos investigadores) por el desarrollo de los algoritmos de procesado de imágenes originalmente involucrados en cryo-EM.

Un microscopio electrónico genera imágenes de alta resolución que contienen muchas copias de una misma proteína. Sin embargo, esas imágenes están corrompidas por altos niveles de ruido y distorsiones (Fig 1.A). Al principio las partículas, sombras de la proteína proyectadas en diferentes orientaciones, son localizadas, cajas verdes de la Fig 1.A. El paso clave viene después, donde las partículas con un patrón parecido son agrupadas y promediadas para obtener una versión libre de ruido y distorsiones (Fig 1.B). Finalmente, a partir de estas proyecciones en diferentes orientaciones una imagen 3D es reconstruida a escala atómica (Fig 1.C y D).

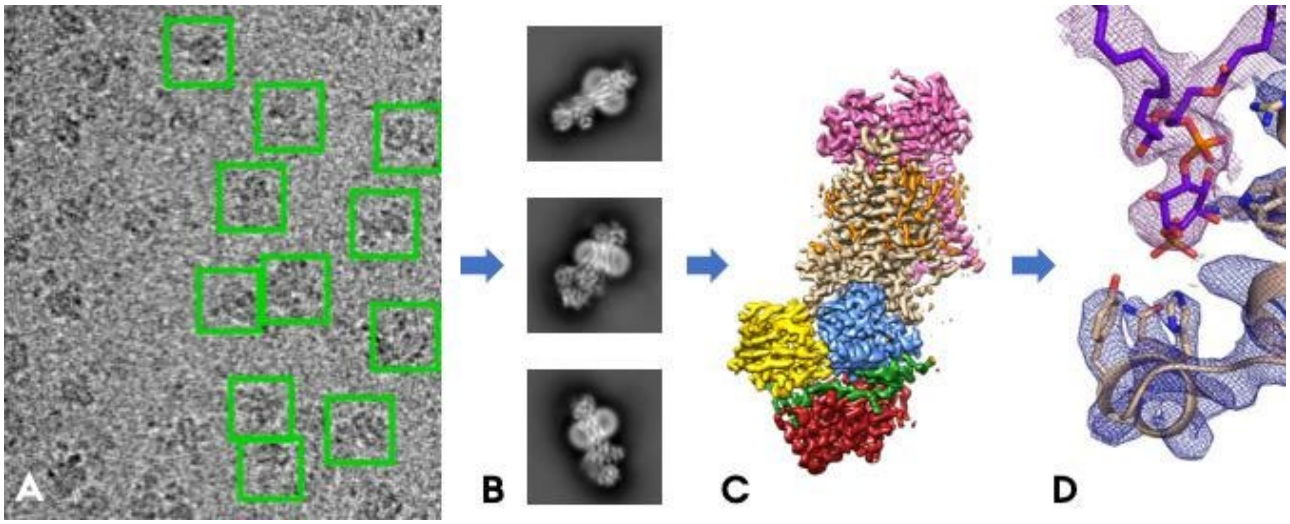

Figure 1. A) Crío-micrografía con algunas particulas seleccionadas. B) Imágenes de tres proyecciones diferentes libres de distorsiones. C) Reconstrucción 3D y D) zoom para ver como ajusta el modelo atómico.

2. Promediado de imágenes

Aquí proponemos resolver el problema del promediado de imágenes necesario para reducir el ruido, eliminar distorsiones y descartar posibles errores en la selección de partículas.

Este problema se puede expresar en términos de optimización combinatoria, ya que se trataría de buscar el vector $X = \{x_1, x_2, \dots, x_n\}$ que genere un valor de *Zero Mean Cross-Correlation* (ZNCC) máximo. Los valores de un elemento x_k del vector X, solo pueden ser: 0 imagen de una partícula errónea, 1 imagen para el subgrupo 1, 2 imagen para el subgrupo 2.

La similitud entre dos imágenes en escala de grises, I_1 y I_2 con el mismo tamaño, se puede medir mediante:

$$ZNCC(I_1, I_2) = \frac{\sum_{i,j}(I_1(i,j) - \mu_1)(I_2(i,j) - \mu_2)}{N \cdot \sigma_1 \cdot \sigma_2}$$

Donde N es número de píxeles de una imagen, $I(i,j)$ es el valor de gris del píxel in la posición i y j, μ es el valor medio de los píxeles y σ su desviación estándar.

Como el patrón dominante en las imágenes no es conocido se crearán dos subconjuntos de imágenes, imágenes etiquetadas como 1 o 2 en X. La función a maximizar es $ZNCC(\bar{I}_1, \bar{I}_2)$, donde $\bar{I}_1$ y $\bar{I}_2$ son el promedio de las imágenes del subconjunto 1 y 2 respectivamente.

2.1. El conjunto de datos

Se proporcionan dos clases **Imagen.java** e **PromediadoImagen.java** que ya tienen implementada la funcionalidad para generar un conjunto de imágenes de entrada (a partir de una imagen de referencia), y los métodos para calcular μ, σ y promediar imágenes. El conjunto de datos generados contendrá imágenes con el patrón de referencia (buenas) y con su negativo (malas, en un número menor) corrompidas por distorsiones:

- Se elimina (poner a 0) una mitad de los píxeles para simular la pérdida de información por las distorsiones, hay cuatro posibilidades: eliminar la parte de arriba, abajo, izquierda y derecha.
- Se añade ruido Gaussiano con una cierta desviación estándar (usa un valor de 5).

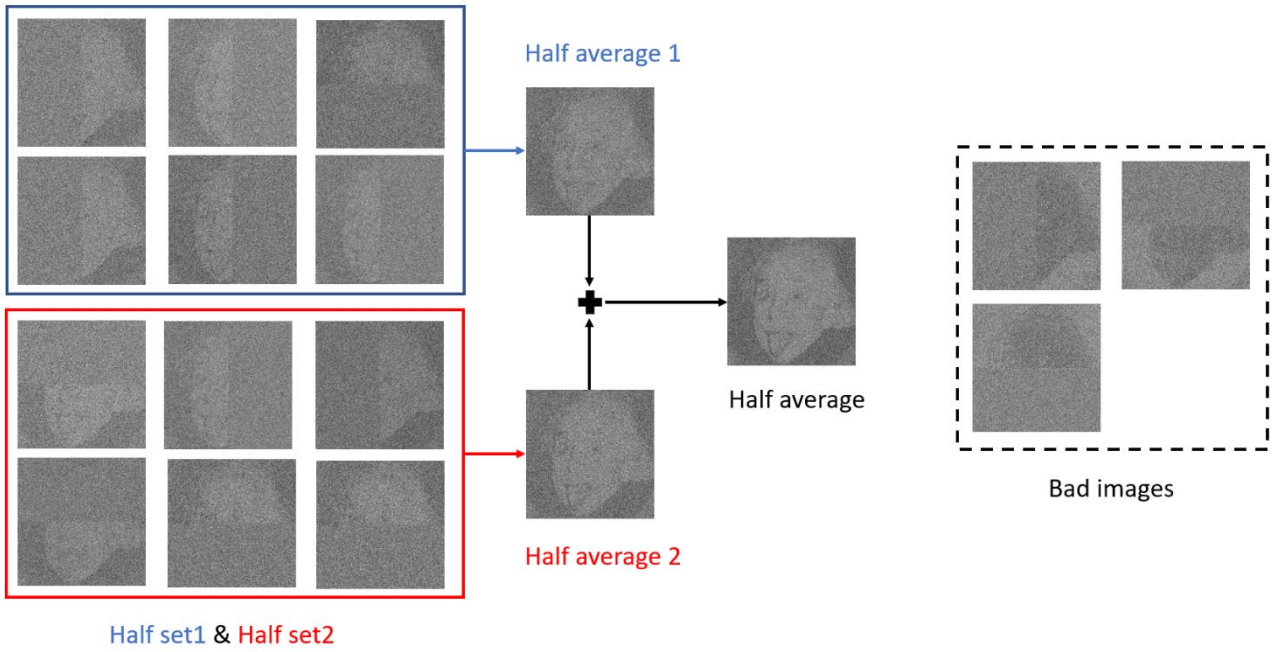

Figure 2. Una posible solución de un conjunto de 15 imágenes (6 en el primer subconjunto, 6 en el segundo, 3 identificadas como imágenes malas). Observa como las imágenes individuales solo contienen parcialmente el patrón de referencia, se trata de buscar que los promedios contengan imágenes completas en donde el ruido se ha reducido, cuanto más parecidos sean el promedio de 1 y 2 mayor será el valor de ZNCC y mejor será el resultado final.

2.2. Algoritmo voraz

Actualmente este problema se suele resolver mediante el uso de algoritmos voraces. En esencia, se van generando instancias aleatorias de X y nos quedamos con la que proporcione un valor de $ZNCC$ más alto.

2.3. Backtracking

Puesto que un algoritmo voraz no garantiza la exploración completa del espacio de soluciones en X, aquí proponemos utilizar la técnica de backtracking.

Para reducir el espacio de búsqueda proponemos la **condición de balaceo** para los subconjuntos, esto es la diferencia en el número de imágenes que pertenecen al subconjunto 1 y 2 no puede ser mayor que uno. La intuición nos dice que una partición balanceada tiende a estar próxima a la óptima ya que las imágenes promedio tendrán niveles de ruido parecidos.

3. Trabajo a realizar

3.1. Programación

Implementa el siguiente código (el resto ya está proporcionado):

- El método que implementa el algoritmo voraz en la clase **PromediadorImagen.java**.
- El método que implementa el algoritmo backtracking en la clase **PromediadorImagen.java**.
- El método que implementa el algoritmo backtracking con la condición de balanceo en la clase **PromediadorImagen.java**.

3.2. Medidas

Completa la siguiente tabla teniendo en cuenta que **n** es el número de imágenes (para el algoritmo voraz ajusta el número de instancias aleatorias generadas a **n** también). Para ello completa el código del método *main* la clase **PromediadorImagenBench.java**:

n	Tiempo_BT	Tiempo_BT_balanceo	ZNCC_voraz	ZNNC_BT	ZNNC_BT_balanceo
2					
3					
4					
5					
6					
…					
Hasta intratable					

Puesto que el conjunto de datos es generado mediante un proceso aleatorio, los valores de ZNCC pueden variar para diferentes ejecuciones.

3.3. Preguntas

Escribe una respuesta justificada a las siguientes cuestiones:

a) ¿Qué algoritmo proporciona mejores resultados y por qué?

b) ¿Qué algoritmo usarías para procesar un conjunto de datos con un millón de imágenes? Explica por qué.
c) Determina la complejidad temporal del algoritmo backtracking sin considerar la condición de balanceo. Valida este análisis utilizando las medidas experimentales.
d) En términos de tiempo, ¿es ventajoso incluir la condición de balanceo? ¿afecta esta condición a la calidad de los resultados?

Incluye en tu proyecto un nuevo paquete `algstudents.s6_es` con el siguiente contenido:

- El código fuente del paquete
- Un documento con nombre `sesion6.pdf` con las actividades propuestas.

Esta sesión consta de dos clases (o semanas), la fecha de entrega será el día anterior al comienzo de la siguiente práctica.