Guía de Ejercicios: Técnicas Computacionales para Síntesis de Imágenes en Radioastronomía (GPU)

DIINF–USACH

30 de octubre de 2025

Objetivos

Al finalizar, el/la estudiante será capaz de:

- Ejecutar y comparar versiones CPU (NumPy) y GPU (CuPy, Numba–CUDA) para tareas típicas de formación de imágenes (gridding, FFT, convoluciones).
- Orquestar cómputo con `dask` para datos grandes y/o múltiples GPUs.
- Medir tiempos correctamente en GPU y analizar escalamiento con tamaño del problema.

Requisitos de entorno

- Python 3.10+ y CUDA Toolkit compatible con su GPU NVIDIA.
- `numpy`, `cupy-cuda`, `numba`, `dask[array]`, opcional: `dask-cuda`, `matplotlib`.
- Verifique GPU disponible con `nvidia-smi` y desde Python: `cp.cuda.runtime.getDeviceCount()`.

Medición (orientado a Jupyter). Use las magics `%%time` o `%%timeit`. En GPU, sincronice al final de la celda para medir correctamente. Realice un calentamiento previo antes de medir.

1. Ejercicio 1: Operaciones vectorizadas y comparación CPU vs GPU

Implemente operaciones básicas en 2D sobre una imagen sintética (tamaños: $N \in \{1024, 2048, 4096, 8192\}$).

a) Genere una imagen con varias fuentes puntuales y una gaussiana extendida.
b) Aplique ufuncs y filtros simples (p.ej., $I \mapsto \log(1 + I)$; convolución 2D con kernel pequeño 5×5).
c) Compare: NumPy (CPU) vs CuPy (GPU). Reporte tiempos y speedup $= t_{CPU}/t_{GPU}$.

2. Ejercicio 2: FFT 2D e imagen sucia en grilla

Construya una **imagen sucia** a partir de visibilidades en una malla w regular.

a) Genere visibilidades $V(u, v)$ en grilla para una escena simple (dos fuentes puntuales separadas).[1]
b) Obtenga la imagen sucia con IFFT 2D: NumPy vs CuPy (`np.fft.fftshift`, `cp.fft.fftshift`).
c) Mida tiempos para varios tamaños y grafique speedup vs N.

[1] Puede simular la transformada directa de la imagen conocida.

3. Ejercicio 3: Gridding con Numba–CUDA (no uniforme w)

Implemente **gridding** por vecino más cercano (NN) de visibilidades no regulares sobre una malla w.

a) Versión CPU (doble bucle) como referencia y validación.
b) Kernel Numba–CUDA que, por visibilidad, acumule en el píxel w más cercano (con pesos naturales). Evite colisiones con `cuda.atomic.add`.
c) Compare tiempos para M visibilidades (p.ej. $10^5, 10^6$) y resolución de malla. Reporte error RMS entre CPU y GPU.

4. Ejercicio 4: Convolutional gridding (Kaiser–Bessel) opcional

Extienda el gridding con un kernel de convolución compacto (p.ej., Kaiser–Bessel) en una vecindad $K \times K$. Compare calidad (PSF) y costo computacional CPU vs GPU.

5. Ejercicios aplicados

A. Escena de doble fuente y lóbulos laterales de la PSF

Simule dos fuentes puntuales separadas en la imagen; obtenga $V(u, v)$ muestreando una cobertura w realista (pistas: trayectoria de Tierra, subconjunto de líneas de base). Forme imagen sucia y analice lóbulos laterales de la PSF; cuantifique el pico secundario.

B. Imagen de fuente extendida y restauración

Genere una galaxia gaussiana; forme imagen sucia y luego **restáurela** con un haz gaussiano (convolución). Compare FWHM y flujo total antes/después.

C. Pesos natural vs uniforme

Implemente pesos natural y uniforme en el gridding. Compare PSF (ancho de haz) y ruido en la imagen sucia.

Reporte y rúbrica

Entregue:
(i) tablas de tiempos CPU/GPU (con desviación estándar en 5 repeticiones), speedup y tamaño de problema;
(ii) verificación de exactitud (norma o RMS entre CPU y GPU);
(iii) figuras de imagen sucia y PSF para casos aplicados;
(iv) discusión breve de cuellos de botella y recomendaciones (tamaño de lote, chunks, transferencia host↔device, atomicidad en kernels).

Caso	Tamaño	CPU [s]	GPU [s]	Speedup
Op. vectorizadas	$N = 2048$			
IFFT 2D	$N = 4096$			
Gridding NN	$M = 10^6$			
Dask+FFT (bloques)	$N = 8192$			

Notas

- Sincronice siempre antes de leer tiempos en GPU. Evite incluir transferencias host↔device salvo que se evalúe el pipeline completo.
- Caliente (*warm-up*) para amortiguar JIT (Numba) y planificaciones (FFT).
- Documente versiones de librerías y modelo de GPU. Si no dispone de GPU, puede correr sólo CPU y comentar resultados esperados.