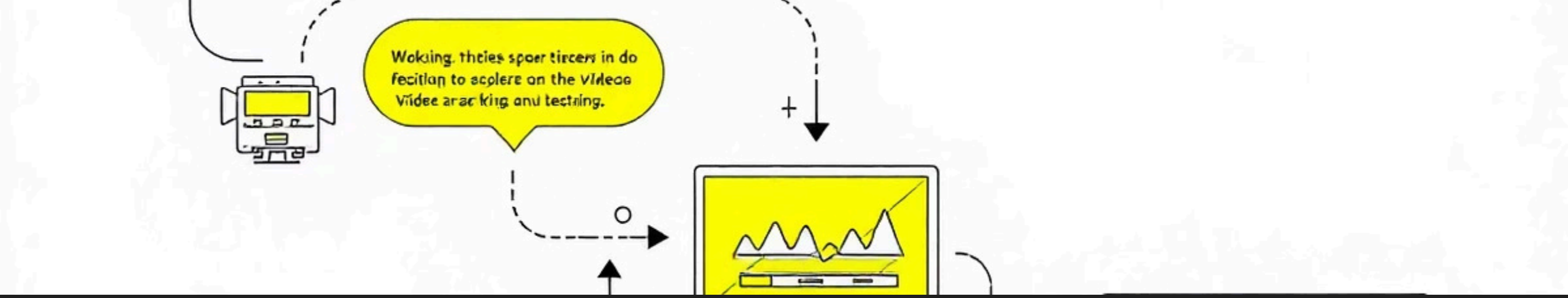

# Procesamiento de Video con CUDA

Transformación de video a escala de grises usando GPU




# Introducción

El procesamiento de video es fundamental en ingeniería e informática, permitiendo manipular y transformar secuencias de imágenes digitales para compresión, análisis de movimiento, visión por computadora y mejora visual.

Las GPU aceleran significativamente tareas computacionalmente demandantes mediante arquitecturas paralelas.

## Objetivo

Comparar algoritmo secuencial (CPU) vs paralelo (GPU con CUDA)

# Objetivos del Proyecto

01

## Implementación Secuencial

Algoritmo en Python que convierte video a escala de grises frame por frame

02

## Implementación Paralela

Algoritmo CUDA con Numba aprovechando la GPU

03

## Medición de Tiempos

Registro de tiempos de ejecución en CPU y GPU

04

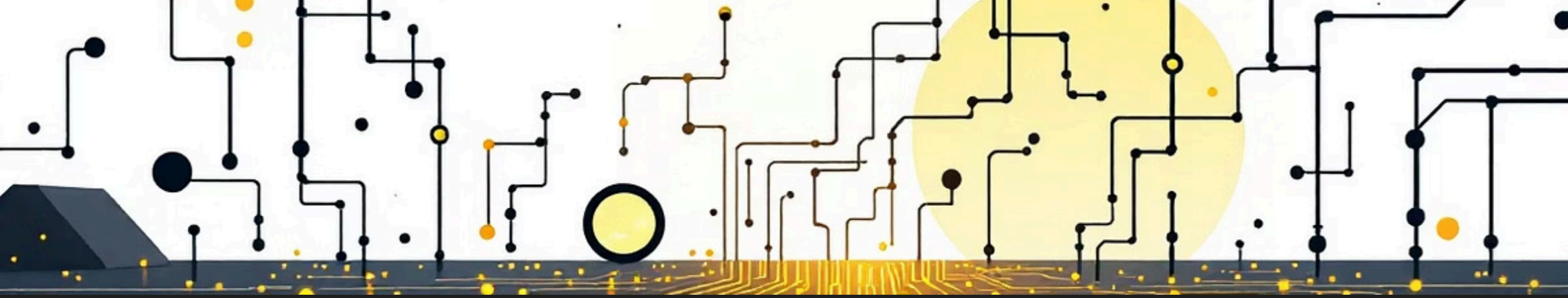
## Cálculo de Speedup

Análisis del rendimiento paralelo vs secuencial

05

## Análisis de Resultados

Discusión de ventajas del procesamiento paralelo



# Marco Teórico

## Procesamiento de Video

Secuencia de imágenes (frames) mostradas a determinada tasa de FPS. Cada frame es una matriz de píxeles con información de color RGB.

## Escala de Grises

Conversión mediante fórmula ponderada: Gray = 0.299·R + 0.587·G + 0.114·B, aproximando la percepción humana de luminancia.

## CUDA y GPU

Plataforma NVIDIA para computación paralela. Ejecuta kernels en miles de hilos organizados en bloques y cuadrículas.

# Configuración Experimental

## Hardware

- Google Colab con GPU NVIDIA
- Tesla T4 o similar
- Memoria de video suficiente para secuencias moderadas

## Software

- Python 3.x
- OpenCV para manejo de video
- NumPy para operaciones numéricas
- Numba CUDA para kernels GPU

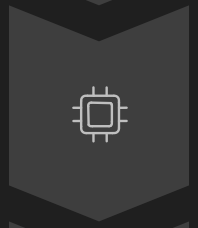
# Metodología de Procesamiento

## Carga de Video

Video original en color

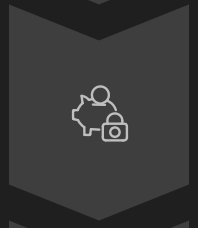
## Extracción

Frames individuales (.jpg)



## Procesamiento

CPU y GPU paralelos



## Guardado

Frames en escala de grises

## Reconstrucción

Video final procesado

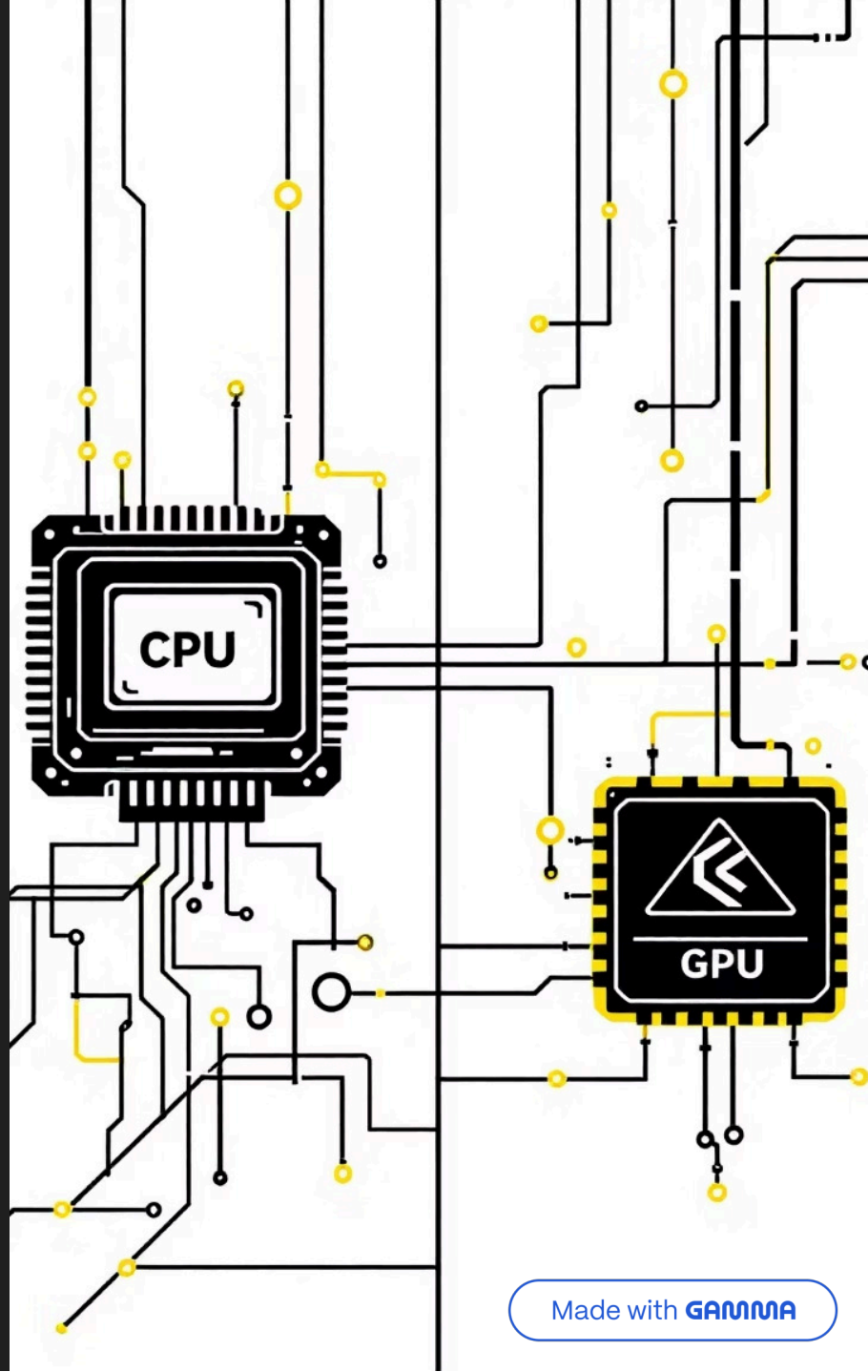
# Algoritmos Implementados

## Algoritmo Secuencial (CPU)

1. Abrir video con cv2.VideoCapture
2. Leer cada frame en ciclo iterativo
3. Convertir a escala de grises con cv2.cvtColor
4. Guardar frame procesado
5. Medir tiempo total de ejecución

## Algoritmo Paralelo (GPU)

1. Convertir frame a arreglo float32
2. Reservar memoria en GPU
3. Definir kernel CUDA para cada píxel
4. Configurar bloques y cuadrículas
5. Ejecutar kernel y copiar resultado



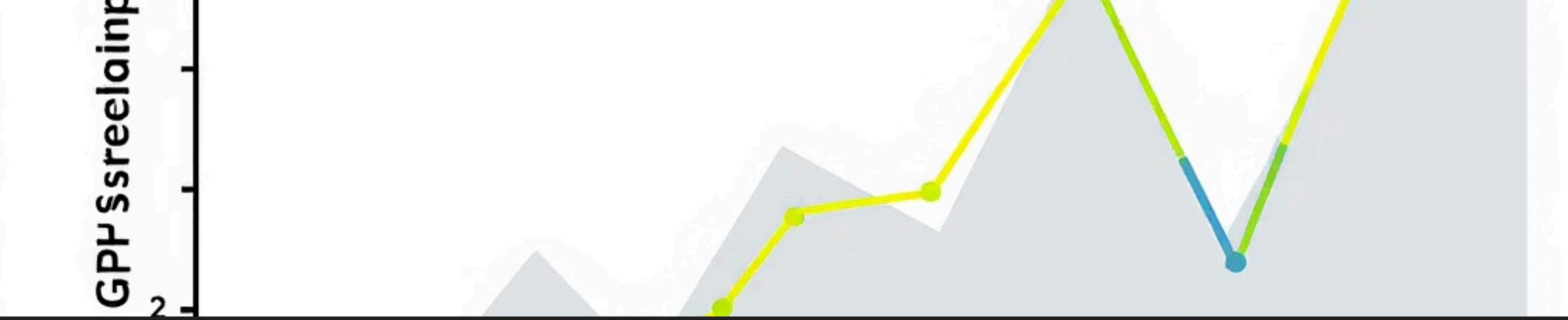
# Fórmula de Speedup

> El speedup mide la ganancia de rendimiento del algoritmo paralelo frente al secuencial

$$Speedup = \frac{T_{secuencial}}{T_{paralelo}}$$

Donde $T_{secuencial}$ es el tiempo en CPU y $T_{paralelo}$ es el tiempo en GPU. Un speedup mayor que 1 indica que el algoritmo paralelo es más rápido.

| X | Y | S |
| --- | --- | --- |
| Tiempo CPU | Tiempo GPU | Speedup |
| Segundos de procesamiento secuencial | Segundos de procesamiento paralelo | Ganancia de rendimiento obtenida |




# Análisis de Rendimiento

El algoritmo paralelo en GPU muestra tiempos considerablemente menores que la versión secuencial cuando el video tiene suficientes frames.

### Ventajas del Paralelismo

Miles de núcleos GPU ejecutan operaciones simultáneas sobre grandes conjuntos de datos, ideal para procesamiento de imágenes.

### Factores a Considerar

Costo de transferencia CPU-GPU, sobrecarga en videos pequeños, y configuración óptima de bloques y cuadrículas.

### Eficiencia Comprobada

Speedup significativamente mayor que 1 confirma que CUDA es estrategia adecuada para volúmenes grandes de datos.

# Conclusiones

## Paralelización Natural

El procesamiento de video es inherentemente paralelizable: cada píxel o frame puede tratarse independientemente.

## Limitaciones de CPU

La implementación secuencial es simple pero presenta limitaciones de tiempo cuando el volumen de datos crece.

## Ventaja de GPU

CUDA con Numba distribuye trabajo entre miles de hilos, obteniendo tiempos de ejecución mucho menores.

## Aprendizaje Práctico

El ejercicio demuestra la importancia del paralelismo y la diferencia entre ejecutar algoritmos en CPU versus GPU.