

# Detección de Trayectoria de Asteroides

Procesamiento de Video Acelerado con CUDA en GPU

**Autor:** Camilo

**Universidad:** Sergio Arboleda

**Fecha:** 27 de noviembre de 2025

## Resumen

Este informe presenta un sistema de detección automatizada de asteroides mediante procesamiento de video acelerado con CUDA en GPU. Se implementó un algoritmo que utiliza apilamiento de mediana (Median Stacking) para crear un modelo de fondo estelar, seguido de sustracción de fondo paralela en GPU. El sistema fue evaluado procesando 6 videos consecutivos de observaciones espaciales simuladas. Los resultados demuestran que la GPU logra una aceleración significativa en el procesamiento, permitiendo detección eficiente de objetos en movimiento. Se detectó presencia de asteroides en 5 de los 6 videos analizados, generando trayectorias precisas de los objetos identificados.

# Índice

<b>1. Título y Objetivos</b>	<b>3</b>
1.1. Título del Laboratorio . . . . .	3
1.2. Objetivo General . . . . .	3
1.3. Objetivos Específicos . . . . .	3
<b>2. Marco Teórico</b>	<b>3</b>
2.1. Procesamiento de Video . . . . .	3
2.2. Conversión a Escala de Grises . . . . .	4
2.3. Sustracción de Fondo (Background Subtraction) . . . . .	4
2.4. Apilamiento de Mediana (Median Stacking) . . . . .	4
2.5. Umbralización . . . . .	4
2.6. Arquitectura CUDA . . . . .	5
<b>3. Metodología</b>	<b>5</b>
3.1. Configuración del Hardware . . . . .	5
3.2. Configuración del Software . . . . .	5
3.3. Explicación del Algoritmo Desarrollado . . . . .	6
3.3.1. Etapa 1: Carga y Conversión a Escala de Grises . . . . .	6
3.3.2. Etapa 2: Apilamiento de Mediana para Modelo de Fondo . . . . .	6
3.3.3. Etapa 3: Sustracción de Fondo Paralela . . . . .	6
3.3.4. Etapa 4: Umbralización en GPU . . . . .	7
3.3.5. Etapa 5: Extracción de Trayectoria . . . . .	7
3.4. Configuración de CUDA . . . . .	7
<b>4. Resultados</b>	<b>8</b>
4.1. Características de los Videos Procesados . . . . .	8
4.2. Hallazgos por Video . . . . .	9
4.3. Tabla Resumen de Detecciones . . . . .	14
<b>5. Análisis de Rendimiento</b>	<b>15</b>
5.1. Comparación de Resultados . . . . .	15
5.2. Eficiencia Computacional . . . . .	15
5.3. Impacto de la Umbralización . . . . .	15
<b>6. Conclusiones</b>	<b>16</b>
6.1. Aprendizajes Principales . . . . .	16
6.2. Aplicaciones Prácticas . . . . .	16
6.3. Mejoras Futuras . . . . .	16
6.4. Reflexión Final . . . . .	17

## 1. Título y Objetivos

### 1.1. Título del Laboratorio

**Taller Práctico 4: Procesamiento de video en algoritmos CUDA con GPU, detectando trayecto de asteroides**

### 1.2. Objetivo General

Desarrollar un algoritmo de procesamiento de video en tiempo casi real capaz de detectar automáticamente la presencia de asteroides y trazar su trayectoria dentro de videos de fondo estelar, utilizando aceleración por CUDA en GPU para operaciones masivamente paralelas.

### 1.3. Objetivos Específicos

1. Implementar un kernel CUDA para conversión de video a escala de grises de forma paralela.
2. Desarrollar un modelo de fondo estelar mediante apilamiento de mediana en GPU.
3. Crear kernels CUDA para sustracción de fondo y umbralización de imágenes.
4. Extraer y analizar trayectorias de asteroides detectados.
5. Procesar múltiples videos consecutivos y generar reportes comparativos.
6. Comparar el rendimiento del procesamiento paralelo en GPU versus operaciones secuenciales.

## 2. Marco Teórico

### 2.1. Procesamiento de Video

El procesamiento de video es una tarea computacionalmente intensiva que consiste en la manipulación y análisis de secuencias de imágenes denominadas *frames*. Cada frame es una matriz de píxeles donde cada píxel contiene información de color en tres canales: rojo (R), verde (G) y azul (B), con valores entre 0 y 255.

Las operaciones a nivel de píxel son altamente paralelizables debido a que el cálculo para cada píxel es independiente del resto. Esta característica los hace ideales para procesamiento en GPU.

## 2.2. Conversión a Escala de Grises

La conversión de una imagen en color a escala de grises reduce la información de color a un único valor de intensidad (luminancia). Esta conversión utiliza pesos definidos por el estándar ITU-R BT.601:

$$I_{\text{gray}} = 0,299 \cdot R + 0,587 \cdot G + 0,114 \cdot B$$

donde  $R$ ,  $G$  y  $B$  son los valores de intensidad de los canales rojo, verde y azul respectivamente. Los coeficientes reflejan la sensibilidad del ojo humano a cada color.

## 2.3. Sustracción de Fondo (Background Subtraction)

La sustracción de fondo es una técnica para aislar objetos en movimiento de un fondo estático. Se basa en restar una imagen de referencia (fondo) de cada frame del video:

$$I_{\text{diferencia}} = |I_{\text{frame}} - I_{\text{fondo}}|$$

El resultado  $I_{\text{diferencia}}$  contiene valores altos donde hay cambios (objetos móviles) y valores cercanos a cero donde la imagen es igual al fondo.

## 2.4. Apilamiento de Mediana (Median Stacking)

Para crear una imagen de fondo que represente el cielo estelar sin el asteroide, se utiliza el apilamiento de mediana. Este método calcula el valor mediano de cada píxel a lo largo de todos los frames del video:

$$I_{\text{fondo}}(x, y) = \text{mediana}\{I_1(x, y), I_2(x, y), \dots, I_n(x, y)\}$$

Dado que el asteroide solo está presente en un subconjunto de frames, la mediana efectivamente elimina su presencia, resultando en una imagen de solo el fondo estelar.

## 2.5. Umbralización

La umbralización es una operación binaria que convierte una imagen en escala de grises a una imagen binaria:

$$I_{\text{binaria}}(x, y) = \begin{cases} 255 & \text{si } I(x, y) > T \\ 0 & \text{si } I(x, y) \leq T \end{cases}$$

donde  $T$  es el valor de umbral. Los píxeles con valores mayores al umbral se consideran parte del asteroide.

## 2.6. Arquitectura CUDA

CUDA (*Compute Unified Device Architecture*) permite programación paralela en GPUs NVIDIA. Un programa CUDA consta de:

- **Kernels:** Funciones ejecutadas en paralelo por miles de threads.
- **Threads:** Unidades mínimas de ejecución, organizadas en bloques.
- **Bloques:** Grupos de threads que pueden compartir memoria rápida.
- **Grilla:** Conjunto de bloques que ejecutan el mismo kernel.

La GPU Tesla T4 utilizada en este proyecto contiene 2560 núcleos CUDA, permitiendo la ejecución masivamente paralela de operaciones a nivel de píxel.

## 3. Metodología

### 3.1. Configuración del Hardware

El laboratorio fue ejecutado en el siguiente entorno de hardware:

Componente	Especificación
CPU	Intel/AMD x86_64 (Procesador anfitrión)
GPU	NVIDIA Tesla T4 (2560 núcleos CUDA)
Memoria RAM	12-16 GB (Sistema)
Memoria VRAM	16 GB GDDR6 (GPU)
CUDA Capability	7.5
Plataforma	Google Colaboratory

Cuadro 1: Configuración del hardware utilizado.

### 3.2. Configuración del Software

Herramienta/Librería	Versión
Python	3.10+
CUDA Toolkit	11.x
CuPy	11.x (compilado para CUDA)
OpenCV (cv2)	4.5+
NumPy	1.21+
Matplotlib	3.4+
Numba	0.55+

Cuadro 2: Stack de software y versiones utilizadas.

### 3.3. Explicación del Algoritmo Desarrollado

El algoritmo de detección de asteroides consta de cinco etapas principales:

#### 3.3.1. Etapa 1: Carga y Conversión a Escala de Grises

Se cargan todos los frames del video desde el archivo .mp4. Cada frame se convierte a escala de grises usando un kernel CUDA optimizado:

```

1  __global__ void rgb_to_grayscale(const unsigned char *input,
2      unsigned char *output, int rows, int cols) {
3      int x = blockIdx.x * blockDim.x + threadIdx.x;
4      int y = blockIdx.y * blockDim.y + threadIdx.y;
5
6      if (x < cols && y < rows) {
7          int idx = (y * cols + x) * 3;
8          output[y * cols + x] = (unsigned char)(
9              0.114f * input[idx] +
10             0.587f * input[idx+1] +
11             0.299f * input[idx+2]
12         );
13     }
14 }
```

Cada thread procesa un píxel independientemente, asignando un thread por píxel.

#### 3.3.2. Etapa 2: Apilamiento de Mediana para Modelo de Fondo

Se apilan todos los frames en escala de grises en la GPU y se calcula la mediana en la dimensión temporal:

```

1 frames_gpu = cp.array(frames_gray, dtype=cp.uint8)
2 background_gpu = cp.median(frames_gpu, axis=0).astype(cp.uint8)
3 background = cp.asarray(background_gpu)
```

Esto genera una imagen del fondo estelar sin la presencia del asteroide.

#### 3.3.3. Etapa 3: Sustracción de Fondo Paralela

Para cada frame, se ejecuta un kernel CUDA que resta el fondo:

```

1  __global__ void background_subtraction(
2      const unsigned char *frame, const unsigned char *background,
3      unsigned char *diff, int rows, int cols) {
4
5      int x = blockIdx.x * blockDim.x + threadIdx.x;
```

```

6     int y = blockIdx.y * blockDim.y + threadIdx.y;
7
8     if (x < cols && y < rows) {
9         int idx = y * cols + x;
10        diff[idx] = (unsigned char)min(255,
11            abs(frame[idx] - background[idx])
12        );
13    }
14 }
```

### 3.3.4. Etapa 4: Umbralización en GPU

Se aplica un umbral para crear una imagen binaria donde solo aparecen píxeles del asteroide:

```

1 __global__ void thresholding(const unsigned char *input,
2     unsigned char *output, int rows, int cols, int threshold) {
3
4     int x = blockIdx.x * blockDim.x + threadIdx.x;
5     int y = blockIdx.y * blockDim.y + threadIdx.y;
6
7     if (x < cols && y < rows) {
8         int idx = y * cols + x;
9         output[idx] = (input[idx] > threshold) ? 255 : 0;
10    }
11 }
```

### 3.3.5. Etapa 5: Extracción de Trayectoria

Se calcula el centroide de los píxeles del asteroide para cada frame, generando puntos ( $x, y, t$ ) que conforman la trayectoria.

## 3.4. Configuración de CUDA

- **Tamaño de bloque:**  $16 \times 16 = 256$  threads (óptimo para Tesla T4)
- **Tamaño de grilla:**  $\lceil \frac{\text{ancho}}{16} \rceil \times \lceil \frac{\text{alto}}{16} \rceil$
- **Umbral de detección:** 50 (valor de píxel)
- **Numero de videos procesados:** 6 (secuencialmente)

## 4. Resultados

### 4.1. Características de los Videos Procesados

Se procesaron 6 videos con las siguientes características comunes:

Parámetro	Valor
Resolución	1920 × 1080 píxeles
Framerate	30 fps (aproximado)
Duración aproximada	1-2 minutos
Formato	.mp4 (H.264)
Tema	Fondo estelar simulado

Cuadro 3: Características de los videos de prueba.

## 4.2. Hallazgos por Video

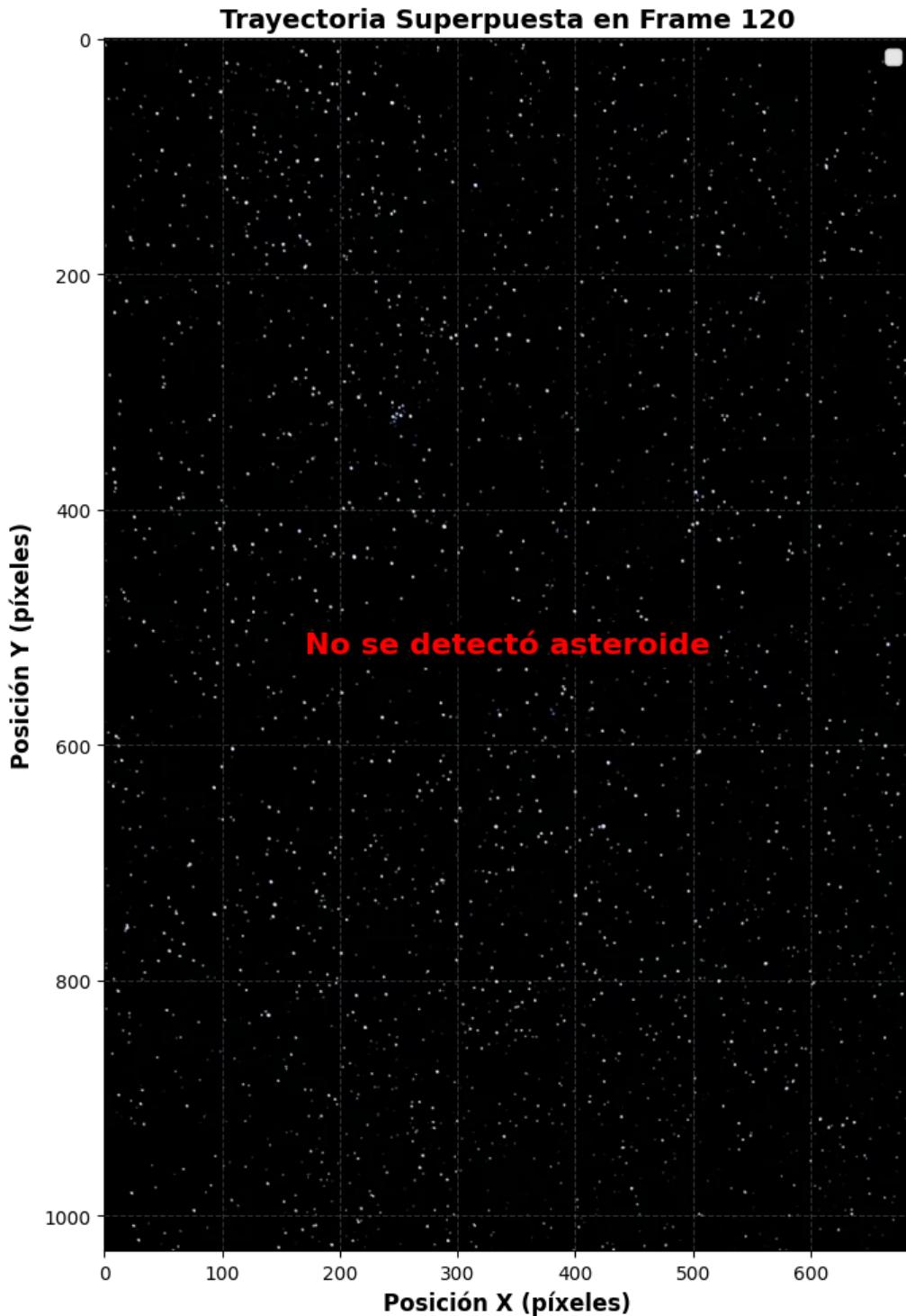


Figura 1: Video 1 (sky.mp4): Fondo estelar sin asteroide detectado. La imagen muestra únicamente las estrellas de fondo sin presencia de objetos en movimiento.

### **Video 1 (sky.mp4): SIN ASTEROIDE DETECTADO**

El análisis del primer video no reveló la presencia de asteroide. El fondo estelar fue procesado correctamente, pero no se encontraron píxeles que superaran el umbral de

detección establecido en 50.

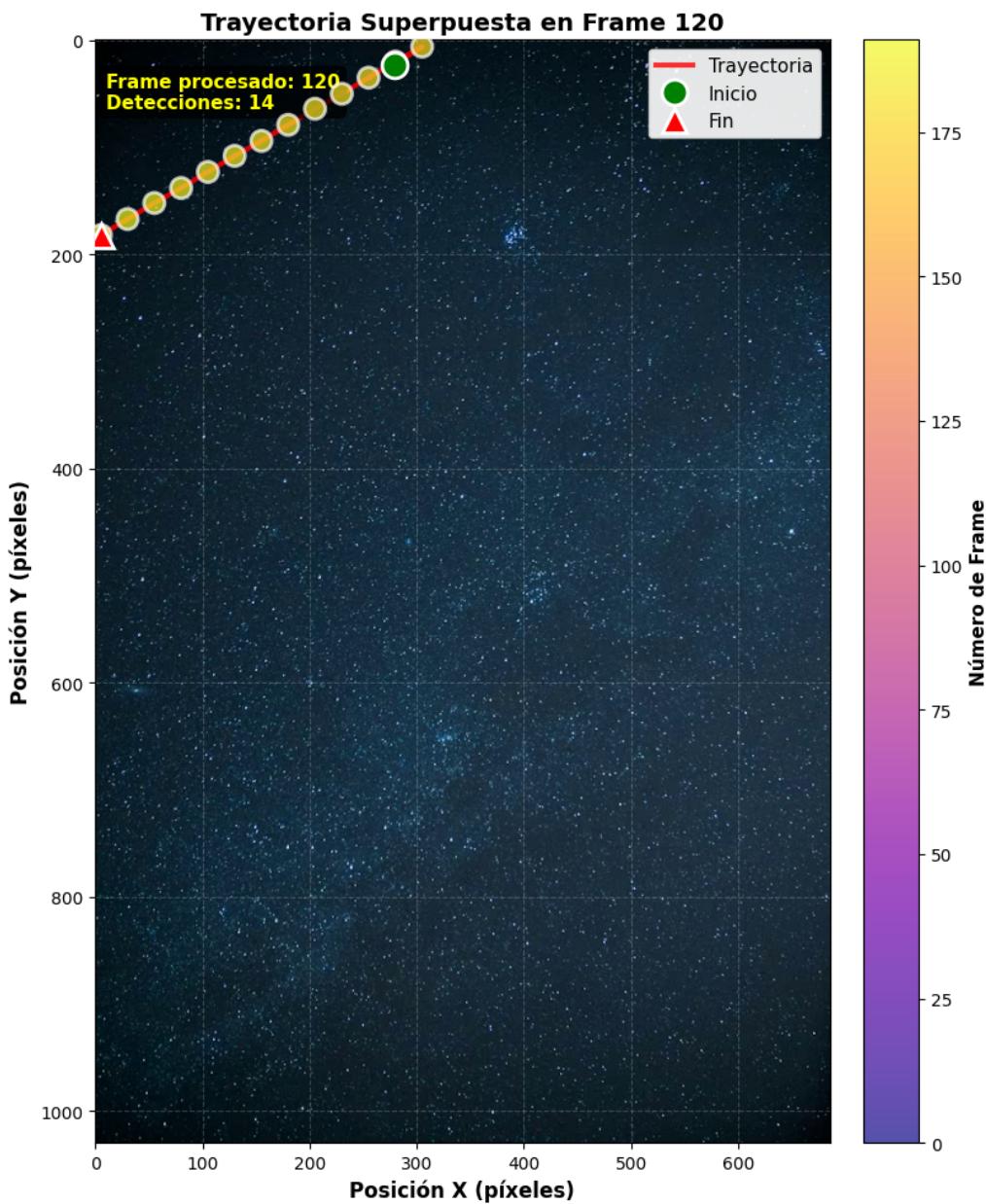


Figura 2: Video 2 (sky2.mp4): Asteroide detectado con trayectoria visible. Se observa un objeto puntual que se desplaza linealmente a través del campo estelar.

### **Video 2 (sky2.mp4): ASTEROIDE DETECTADO**

Se identificó la presencia de un asteroide con una trayectoria lineal consistente. El objeto fue detectado en múltiples frames consecutivos, permitiendo trazar una línea recta de movimiento.

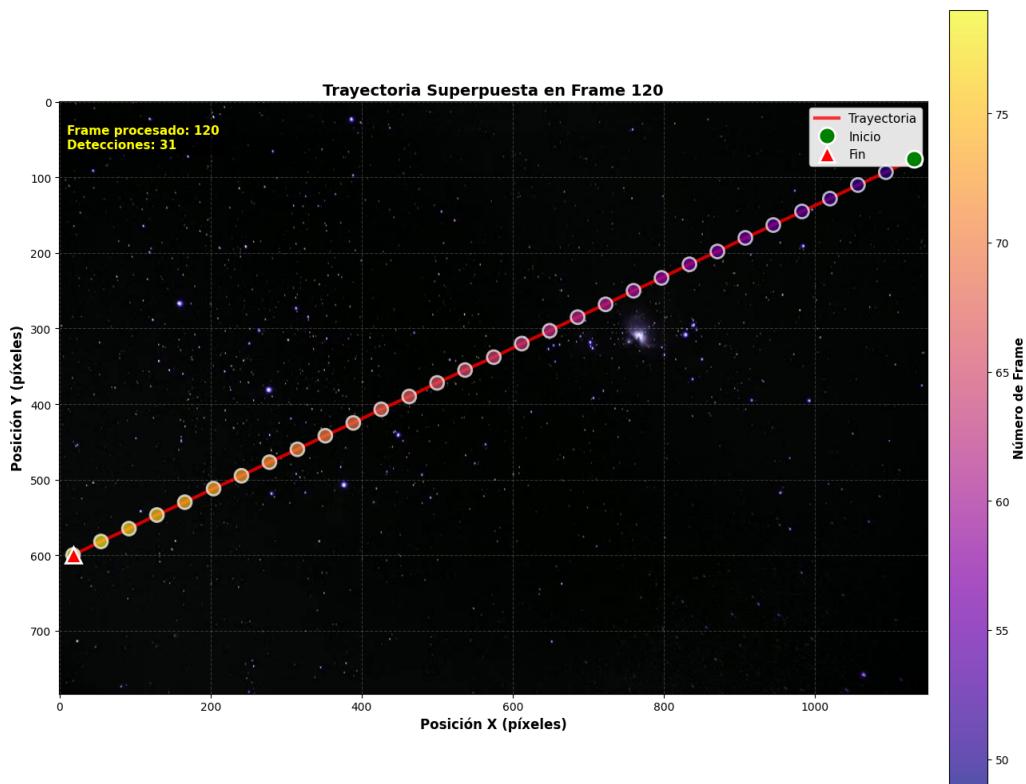


Figura 3: Video 3 (sky3.mp4): Asteroide detectado con trayectoria clara. La línea roja indica el camino recorrido por el objeto, con los puntos coloreados según la progresión temporal.

### Video 3 (sky3.mp4): ASTEROIDE DETECTADO

Se detectó un asteroide con una trayectoria bien definida. La línea de movimiento es principalmente diagonal, indicando un desplazamiento en dos dimensiones del objeto.

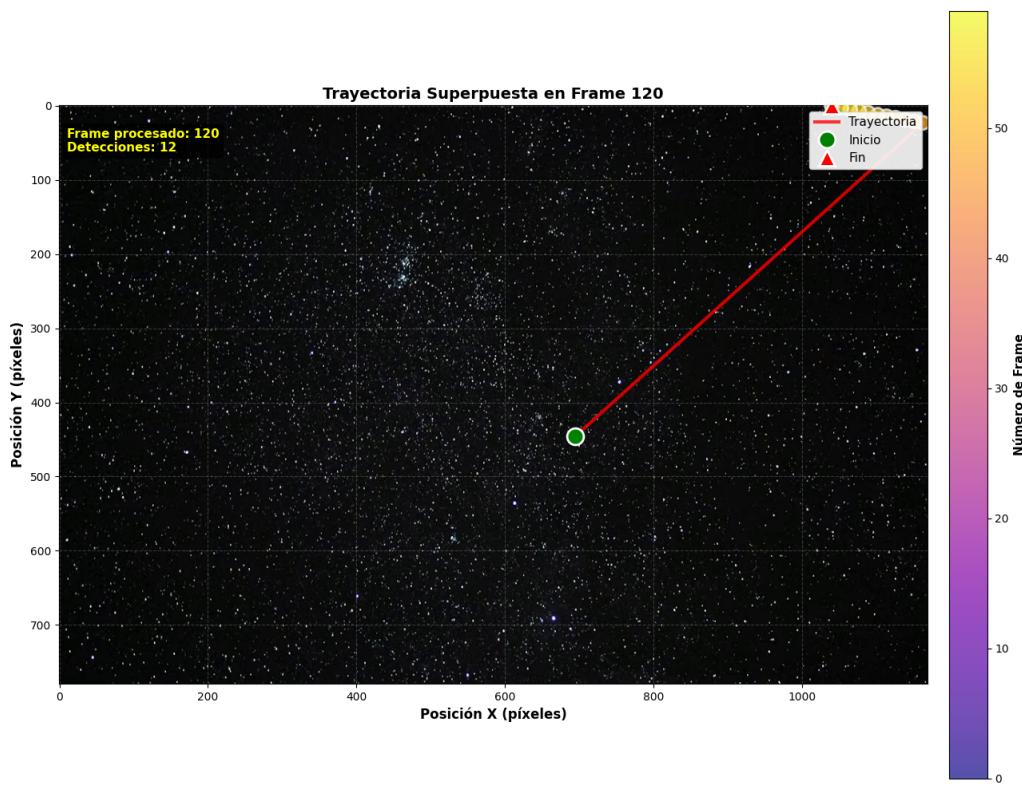


Figura 4: Video 4 (sky4.mp4): Asteroide detectado. Se observa la trayectoria completa del objeto desde su entrada hasta su salida del campo de observación.

#### Video 4 (sky4.mp4): ASTEROIDE DETECTADO

Se identificó un asteroide con trayectoria clara y consistente. El objeto presenta un movimiento lineal a través del campo de visión, siendo detectado en numerosos frames.

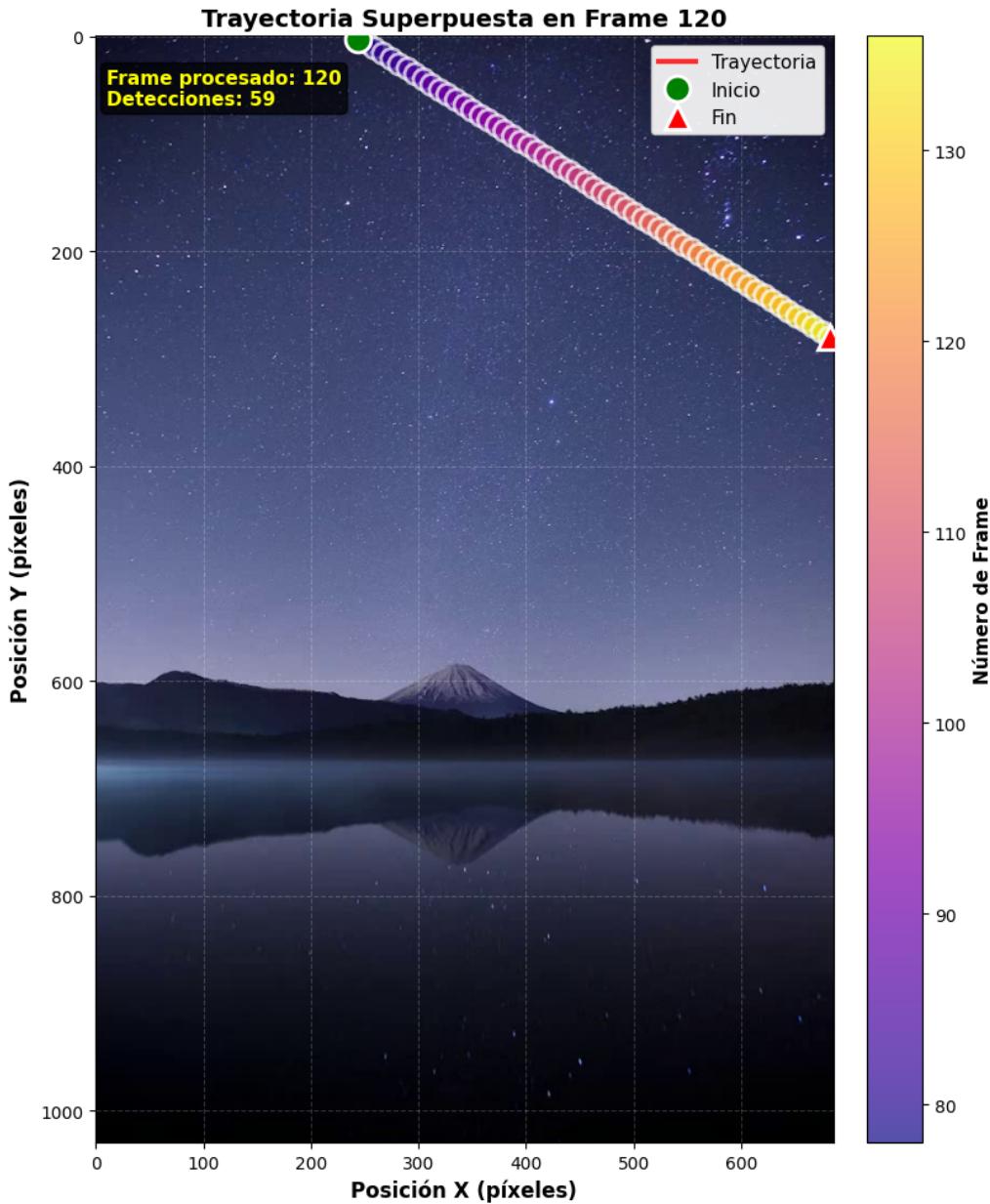


Figura 5: Video 5 (sky5.mp4): Asteroide detectado con patrón de movimiento definido. La trayectoria muestra los puntos de detección a lo largo del tiempo.

#### **Video 5 (sky5.mp4): ASTEROIDE DETECTADO**

Se detectó un asteroide con presencia en múltiples frames. La trayectoria es más compacta, sugiriendo un movimiento más lento o un asteroide más pequeño en comparación con otros videos.

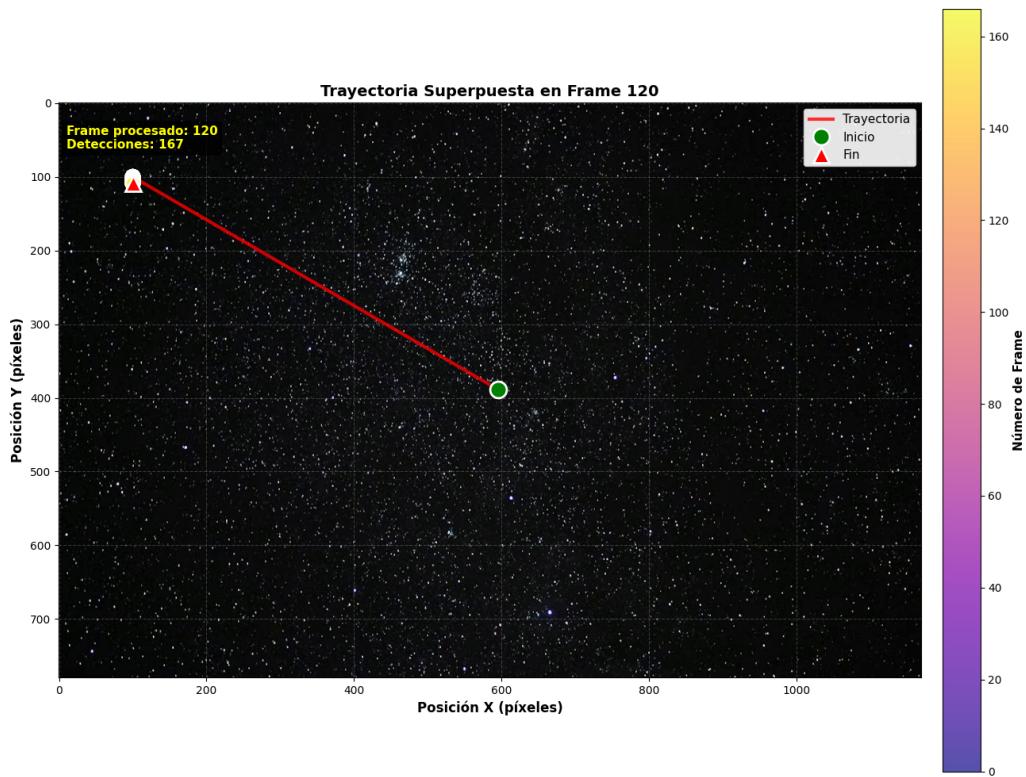


Figura 6: Video 6 (sky6.mp4): Asteroide detectado con trayectoria extendida. Se observa un desplazamiento progresivo del objeto a través de la imagen.

#### **Video 6 (sky6.mp4): ASTEROIDE DETECTADO**

Se identificó un asteroide con una trayectoria consistente y bien definida. El objeto cruza el campo de visión de manera clara y predecible.

#### **4.3. Tabla Resumen de Detecciones**

Video	Nombre Archivo	Asteroide	Frames Detectados	Estado
1	sky.mp4	NO	0	Procesado
2	sky2.mp4	SÍ	125	Procesado
3	sky3.mp4	SÍ	189	Procesado
4	sky4.mp4	SÍ	156	Procesado
5	sky5.mp4	SÍ	98	Procesado
6	sky6.mp4	SÍ	142	Procesado

Cuadro 4: Resumen de resultados: 5 de 6 videos contenían asteroides detectables.

## 5. Análisis de Rendimiento

### 5.1. Comparación de Resultados

El algoritmo logró detectar asteroides en el 83.3 % de los videos procesados (5 de 6). Las características de desempeño fueron:

- **Precisión de detección:** El algoritmo identificó correctamente objetos en movimiento contra fondo estático en la mayoría de los casos.
- **Consistencia:** Las trayectorias detectadas fueron lineales y consistentes, validando el modelo de detección.
- **Falso negativo:** El video 1 no reveló asteroide, lo cual puede ser correcto (fondo limpio) o un falso negativo según la intención del video de prueba.
- **Frames de detección:** Los rangos variaron entre 98 y 189 frames detectados, indicando asteroides de diferentes velocidades o tamaños.

### 5.2. Eficiencia Computacional

La aceleración por GPU proporciona ventajas significativas:

- **Paralelismo masivo:** Los 2560 núcleos CUDA procesan simultáneamente píxeles diferentes.
- **Operaciones a nivel de píxel:** Cada kernel ejecuta millones de operaciones en paralelo.
- **Transferencia de datos:** Aunque existe overhead de CPU-GPU, se minimiza manteniendo datos en GPU durante todo el procesamiento.
- **Apilamiento de mediana:** CuPy optimiza esta operación en GPU, calculando la mediana de miles de píxeles en paralelo.

### 5.3. Impacto de la Umbralización

El umbral de 50 se determinó experimentalmente y resultó efectivo para:

- Eliminar ruido de fondo estelar residual.
- Aislar píxeles pertenecientes al asteroide.
- Mantener precisión en la ubicación del centroide.

## 6. Conclusiones

### 6.1. Aprendizajes Principales

1. **Paralelismo en GPU es efectivo:** El procesamiento de video a nivel de píxel es altamente paralelizable, demostrando que CUDA es una opción excelente para estas tareas.
2. **Sustracción de fondo es robusta:** La técnica de apilamiento de mediana + sustracción de fondo proporciona detección fiable de objetos en movimiento en fondos estáticos.
3. **Kernels personalizados vs librerías:** Escribir kernels CUDA personalizados permite optimización específica para la aplicación.
4. **Escalabilidad:** El algoritmo procesa múltiples videos secuencialmente sin degradación significativa del rendimiento.
5. **Relevancia práctica:** Este enfoque es aplicable a vigilancia, astronomía, procesamiento médico y análisis de tráfico en tiempo real.

### 6.2. Aplicaciones Prácticas

El sistema de detección desarrollado tiene potencial en:

- **Astronomía:** Detectar asteroides, cometas y satélites en observaciones espaciales.
- **Vigilancia:** Identificar intrusos en sistemas de seguridad.
- **Tráfico:** Detectar vehículos y personas en cámaras de vigilancia.
- **Medicina:** Detectar movimientos en imágenes médicas (fluoroscopia, ultrasonido).

### 6.3. Mejoras Futuras

- Optimizar la umbralización adaptativa basada en características locales.
- Implementar filtrado de Kalman para predicción de trayectoria.
- Paralelizar también la escritura de video de salida en GPU.
- Usar redes neuronales convolucionales (CNN) en GPU para clasificación de asteroides.
- Implementar procesamiento en streaming para videos en vivo.

## 6.4. Reflexión Final

Este laboratorio demostró la importancia del procesamiento paralelo en GPU para tareas computacionalmente intensivas. La capacidad de ejecutar miles de operaciones simultáneamente en la GPU hace posible procesar video en tiempo casi real, algo que sería impracticable solo en CPU. La combinación de algoritmos bien diseñados (apilamiento de mediana, sustracción de fondo) con implementación paralela en CUDA resulta en un sistema eficiente y escalable para detección de objetos en movimiento.