

Implementación y Evaluación Comparativa de Detección AprilTag (GPU vs. CPU) en Jetson Orin Nano

Implementation and Comparative Evaluation of AprilTag Detection (GPU vs. CPU) on Jetson Orin Nano

Rodrigo Fernández Rojas¹, Luis Antonio Díaz Jiménez¹, Ana Marcela Herrera Navarro¹, Selene Ramírez Rosales¹, Ernesto Rubalcava Durán¹

¹Universidad Autónoma de Querétaro- Facultad de Informática Campus Juriquilla, Av. de las Ciencias s/n, Colonia Juriquilla, Querétaro, Qro. C.P. 76230 rfernandez28@alumnos.uaq.mx, luis.diaz@uaq.mx, mherrera@uaq.mx, selene.ramirez@uaq.mx, garrik@uaq.mx

Fecha de recepción: 1 de octubre de 2025

Fecha de aceptación: 12 de marzo de 2026

Resumen. En este trabajo se presenta una evaluación comparativa de rendimiento entre un pipeline de localización visual acelerado por GPU y uno basado en CPU, sobre una plataforma embebida Jetson Orin Nano. El sistema acelerado utiliza el framework Isaac ROS AprilTag, mientras que el sistema de referencia emplea paquetes estándar de ROS 2 (`image_proc`, `apriltag_ros`). Ambos sistemas procesaron la misma fuente de video, que es un conjunto de datos estándar r2b (`rosbags`). Se compararon los grafos de percepción completos, incluyendo el redimensionamiento de imagen y la detección de marcadores, se midieron métricas clave como el *throughput* (FPS), la latencia y la estabilidad (*jitter*). Los resultados cuantifican la ganancia de rendimiento obtenida mediante la aceleración por hardware, discutiendo las implicaciones para el diseño de sistemas de percepción en vehículos autónomos a escala.

Palabras Clave: AprilTag; Isaac ROS; Jetson Orin Nano Super; IMX219; Localización Visual.

Summary. This work presents a comparative performance evaluation between a GPU-accelerated visual localization pipeline and a CPU-based one, on an embedded Jetson Orin Nano platform. The accelerated system uses the Isaac ROS AprilTag framework, while the reference system employs standard ROS 2 packages (`image_proc`, `apriltag_ros`). Both systems use the same video source, a CSI camera (IMX219) with automatic settings, to ensure a fair comparison under realistic conditions. Validation was carried out reproducibly using the standard r2b (`rosbags`) dataset. The complete perception graphs were compared, including image resize and marker detection, measuring key metrics such as throughput (FPS), latency and stability. The results quantify the performance gain achieved through hardware acceleration, discussing the implications for the design of perception systems in scaled autonomous vehicles.

Keywords: AprilTag; Isaac ROS; Jetson Orin Nano; GPU Acceleration; Visual Location.

1 Introducción

Los sistemas de conducción autónoma dependen críticamente de su capacidad para determinar su posición y orientación (pose) en el entorno. Si bien el GPS [2] es fundamental, su ausencia en entornos interiores o urbanos densos exige soluciones alternativas. La visión por computadora, y en particular los marcadores fiduciales como los AprilTag, ofrece una solución robusta y computacionalmente eficiente para la estimación de pose 6-DoF [6].

Con el avance de plataformas embarcadas como la familia NVIDIA Jetson, es posible ejecutar algoritmos de percepción complejos directamente en el vehículo. El ecosistema Isaac ROS de NVIDIA proporciona implementaciones aceleradas por hardware para estos algoritmos. Sin embargo, para cuantificar el beneficio real de dicha aceleración, es crucial compararla con una implementación tradicional basada en CPU en el mismo hardware.

Este trabajo se enfoca en esa comparación directa: implementamos y evaluamos cuantitativamente el rendimiento de un pipeline de Isaac ROS AprilTag (GPU) frente a un pipeline equivalente construido con paquetes estándar de ROS 2 (CPU) en una plataforma Jetson Orin Nano. El objetivo es medir la ganancia en velocidad (*throughput*) y tiempo de respuesta (latencia) que ofrece la aceleración por hardware.

2 Panorama reducido del estado del arte

La localización basada en visión ha evolucionado desde sistemas pioneros como ARToolKit hasta soluciones más robustas como ArUco y AprilTag. Los desarrollos de Olson et al. en AprilTag lo han convertido en un estándar para la corrección de deriva en sistemas de odometría visual y SLAM [6].

- Sánchez Martín [7] utilizó ARToolKit para el reconocimiento de gestos, demostrando la viabilidad del seguimiento de marcadores para la interacción humano-computadora.
- Jara et al. [3] validaron la precisión de los marcadores ArUco en análisis de movimiento humano, comparándolos con sistemas de fotogrametría y encoders, y encontraron que el error introducido por los marcadores era mínimo.
- Campos Navarro [1] propuso un sistema de navegación para drones en interiores usando AprilTag para compensar la falta de señal GPS, logrando una localización y orientación exitosas.

Tecnologías Actuales Relevantes:

Detección AprilTag (v2/v3): Las versiones modernas del algoritmo, desarrolladas por Olson et al. [6], han mejorado la robustez y velocidad, convirtiéndose en un estándar para la corrección de deriva en sistemas de odometría visual y SLAM.

- Stacks Acelerados (Isaac ROS + NITROS): NVIDIA Isaac ROS ofrece nodos de ROS 2 optimizados que aprovechan los aceleradores de hardware de las plataformas Jetson. Utiliza NITROS (NVIDIA Isaac Transport for ROS), que gestiona la negociación de tipos de datos para minimizar o eliminar las copias de memoria entre la CPU y la GPU, acelerando todo el grafo de percepción [4].
- Cámaras CSI (Argus): El nodo `isaac_ros_argus_camera` interactúa directamente con el Procesador de Señal de Imagen (ISP) de la Jetson, permitiendo una captura de imágenes eficiente, con estampado de tiempo preciso y control de bajo nivel.
- Benchmarking Reproducible: El framework Isaac ROS Benchmark, basado en `ros2_benchmark`, permite ejecutar pruebas estandarizadas utilizando conjuntos de datos como `r2b`, garantizando que los resultados de rendimiento (FPS, latencia) sean consistentes y comparables entre diferentes plataformas o versiones de software [5].

3 Metodología

El proceso de evaluación se diseñó para comparar de manera justa ambos enfoques bajo condiciones idénticas.

3.1 Plataforma de Hardware y Software

- Hardware: Placa de desarrollo NVIDIA Jetson Orin Nano (8 GB).
- Software:
 - Sistema Operativo: Ubuntu 22.04 con JetPack 6.x.
 - Framework: ROS 2 Humble Hawksbill.
 - Librerías: Isaac ROS 3.x.

3.2 Grafo de Percepción

Se analizaron dos flujos de datos (pipelines) completos, desde la captura hasta la detección, compartiendo el mismo nodo de cámara para asegurar una fuente de datos idéntica.

1. Grafo Acelerado (GPU):

- **Flujo:** `DataLoaderNode (ros2_benchmark)` → `ResizeNode (isaac_ros_image_proc)` → `PlaybackNode (isaac_ros_benchmark)` → `AprilTagNode (isaac_ros_apriltag)`.

- **Descripción:** Este pipeline consume el rosbag y utiliza nodos de Isaac ROS optimizados que se ejecutan en la GPU. La comunicación entre ellos está acelerada por NITROS.
- 2. Grafo de Referencia (CPU):**
- **Flujo:** DataLoaderNode (ros2_benchmark) → ResizeNode (image_proc) → PlaybackNode (ros2_benchmark) → AprilTagNode(apriltag_ros)
 - **Descripción:** Este pipeline también consume el rosbag y utiliza paquetes estándar de ROS 2. El nodo ResizeNode realiza el redimensionamiento en la CPU, y AprilTagNode realiza la detección de marcadores, también en la CPU.

3.3 Marco de Evaluación (Benchmarking)

Se utilizó la herramienta Isaac ROS Benchmark para orquestar las pruebas y medir el rendimiento.

- **Dataset de Entrada:** Se empleó el dataset r2b de NVIDIA (formato rosbag) para ambos *pipelines*, específicamente la secuencia r2b_storage [8].
- **Pruebas Ejecutadas:** Se ejecutó una prueba de "grafo completo" para cada uno de los *pipelines* descritos anteriormente, midiendo el rendimiento desde la salida del nodo de redimensionamiento hasta la publicación de la pose del AprilTag.

3.4 Métricas de Rendimiento

Las métricas clave recopiladas por el framework de benchmark son:

- **Throughput (Rendimiento):** Medido en fotogramas por segundo (FPS), representa cuántas imágenes puede procesar el sistema por unidad de tiempo.
- **Latencia Extremo a Extremo (ms):** Medida en milisegundos (ms), representa el tiempo que transcurre desde que un fotograma entra al pipeline hasta que el resultado final es publicado.
- **Jitter (ms):** Variación en la latencia entre fotogramas consecutivos, que mide la estabilidad del sistema.

3.5 Definiciones Formales de las Métricas

Throughput (Rendimiento):

$$FPS = \frac{N_{framesprocesados}}{t_{fin} - t_{ini}}$$

Donde:

- $N_{framesprocesados}$ es el número total de fotogramas procesados.
- T_{ini} es el tiempo de inicio de la prueba.
- T_{fin} es el tiempo de finalización de la prueba.

Capítulo 1 Latencia de extremo a extremo (li):

$$Pi = ti^{out} - ti^{in}$$

Donde:

- ti^{out} es la marca de tiempo de la publicación del fotograma i a la salida del último nodo del pipeline medido.
- ti^{in} es la marca de tiempo de la llegada del fotograma i al primer nodo del pipeline medido.

Capítulo 2 Jitter (ms):

$$J_i = |L_i - L_{i-1}|$$

Donde:

- J_i es el jitter en el fotograma i.
- L_i es la latencia del fotograma actual.
- L_{i-1} es la latencia del fotograma anterior.

4 Resultados y discusión

4.1 Métricas de Rendimiento.

Los resultados cuantitativos de las pruebas comparativas se presentan en la Tabla 1.

Tabla 1. Comparativa de Rendimiento: Pipeline GPU vs. CPU.

Métrica	Pipeline acelerado (GPU)	Pipeline de referencia (CPU)	Ganancia (GPU vs CPU)
Throughput promedio (FPS)	83.95 FPS	19.37 FPS	4.33x más rápido
Latencia Promedio (ms)	11.02 ms	49.59 ms	77.8% menor (más rápido)
Jitter Promedio (ms)	0.74 ms	7.80 ms	90.5% más estable

4.2 Analisis Comparativo

Los resultados demuestran una mejora de rendimiento drástica al utilizar el pipeline acelerado por GPU.

- **Throughput:** El grafo de Isaac ROS (GPU) procesó imágenes a una tasa aproximadamente **4.2 veces superior** a la del grafo basado en CPU. Mientras que la solución de CPU apenas podría seguir el ritmo de una cámara estándar de 20 FPS, la solución de GPU puede manejar flujos de video de alta velocidad, dejando además recursos computacionales libres para otras tareas.
- **Latencia y Jitter:** La variación de la latencia (jitter) se redujo drásticamente. El jitter promedio en el pipeline de GPU fue un **90.5% menor**, y el jitter máximo se redujo en un **60.9%**. Esto es crítico para aplicaciones de robótica en tiempo real, como el control de un vehículo, donde un tiempo de respuesta más consistente y predecible permite reacciones más rápidas y seguras ante los cambios en el entorno.

La diferencia de rendimiento subraya el impacto de descargar el procesamiento de visión de la CPU a la GPU y de optimizar la transferencia de datos con tecnologías como NITROS. El costo computacional de la detección de marcadores es significativo para la CPU, convirtiéndose en un cuello de botella, mientras que la arquitectura paralela de la GPU está diseñada para manejar este tipo de cargas de trabajo de manera eficiente.

4.3 Limitaciones y Observaciones

- **Limitaciones del Dataset (r2b):** El dataset r2b está diseñado para medir el rendimiento computacional. No incluye una "verdad terreno" (ground truth) de la pose de la cámara, por lo que no es posible calcular métricas de exactitud, como el Error Cuadrático Medio (RMSE) de la estimación de pose. Este estudio se centra exclusivamente en el rendimiento.

5 Conclusiones y trabajos futuros de investigación

Este trabajo validó cuantitativamente el beneficio de utilizar un pipeline de detección de AprilTag acelerado por hardware (GPU) en una plataforma Jetson Orin Nano. Los resultados muestran una mejora de rendimiento de más de **4 veces en throughput** y una reducción de la variación de latencia (jitter) de hasta un **90%**, demostrando la viabilidad y ventaja de la solución de Isaac ROS para aplicaciones de robótica de alto rendimiento.

La comparativa directa proporciona una base sólida para que los desarrolladores tomen decisiones informadas sobre la arquitectura de sus sistemas de percepción.

Como trabajo futuro, se proponen las siguientes líneas:

- **Análisis del Consumo Energético:** Comparar el consumo de energía para evaluar la eficiencia energética (rendimiento por vatio).
- **Evaluación de Exactitud:** Incorporar datasets con ground truth de pose para evaluar si existe alguna diferencia en la precisión de la estimación.

- **Comparación con otros Aceleradores:** Realizar un análisis comparativo utilizando otros backends disponibles en la Jetson, como el PVA (Programmable Vision Accelerator).

References

- [1] F. J. Campos Navarro, “Sistema visual con referencia de AprilTags para navegación de un drone,” Informe técnico, CIATEQ, A.C., 2021. [En línea]. Disponible: <https://ciateq.repositorioinstitucional.mx/jspui/bitstream/1020/450/1/Sistema%20visual%20con%20referencia%20de%20apriltags.pdf>
- [2] E. Huerta, A. Mangiaterra, y G. Noguera, GPS: Posicionamiento satelital. UNR Editora, 2005.
- [3] N. Jara, J. L. Abad, J. Navarro, y Á. Page, “Validación de los marcadores ArUco para el análisis de movimientos humanos,” en 11º Simposio CEA de Bioingeniería, 2019.
- [4] NVIDIA, “Isaac ROS AprilTag Documentation,” Isaac ROS Documentation. [En línea]. Disponible: https://nvidia-isaac-ros.github.io/repositories_and_packages/isaac_ros_apriltag/index.html
- [5] NVIDIA, “Isaac ROS Benchmark Documentation,” Isaac ROS Documentation. [En línea]. Disponible: <https://nvidia-isaac-ros.github.io/concepts/benchmarking/index.html>
- [6] E. Olson, “AprilTag: A robust and flexible visual fiducial system,” en Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2011.
- [7] M. Sánchez Martín, “Reconocimiento de patrones de movimiento con ARToolKit,” Trabajo Fin de Grado, Universidad Politécnica de Madrid, 2016.
- [8] NVIDIA, “NVIDIA Isaac ROS R2B Dataset,” NVIDIA NGC Catalog, 2023. [En línea]. Disponible: <https://catalog.ngc.nvidia.com/orgs/nvidia/teams/isaac/resources/r2bdataset2023?version=3>