Revista ELECTRO, Vol. 46, 2024, pp. 369 - 373
https://itchihuahua.mx/revista_electro
ISSN Electrónico: 3061-774X

SISTEMA PARA DETECCIÓN DE OBJETOS MEDIANTE VISION ARTIFICIAL IMPLEMENTADO EN ARQUITECTURA ARM PARA APLICACIONES DE NAVEGACIÓN AUTÓNOMA

Beltrán Valenzuela Nancy Judith[1], De Mateo González Jesús Rodrigo[1], Alcántara Llanas Ivonne Tatiana[1], Rodríguez Rangel Héctor[2], González Huitrón Víctor Alejandro[1]

[1]Tecnológico Nacional de México/Instituto Tecnológico de Querétaro, Av Tecnológico S/N, Centro Histórico, Centro, 76000 Santiago de Querétaro, Qro. Departamento de Sistemas y computación
nancy.bv@queretaro.tecnm.mx, m23140009@queretaro.tecnm.mx, ivonne.al@queretaro.tecnm.mx, victor.gh@queretaro.tecnm.mx

[2]Tecnológico Nacional de México/Instituto Tecnológico de Culiacán, Juan de Dios Batiz No. 310 pte, Guadalupe, 80220 Culiacán Rosales, Sin. Departamento de Posgrado e Investigación
hector.rr@culiacan.tecnm.mx

RESUMEN.

En la industria automotriz, las tecnologías de visión artificial y aprendizaje profundo han revolucionado la seguridad y la eficiencia de los vehículos, fomentando el desarrollo de sistemas de navegación autónoma. Este artículo presenta un sistema de detección de objetos implementado en una arquitectura ARM, utilizando una cámara RGB frontal para capturar imágenes del entorno. Las imágenes se serializan mediante Google Protocol Buffers y se transmiten a un módulo de cómputo usando eCAL (Enhanced Communication Abstraction Layer) como API de Protocol Buffers. El procesamiento de las imágenes se realiza mediante el algoritmo YOLO (You Only Look Once), conocido por su precisión y rapidez en la detección de objetos. Los resultados se despliegan en un monitor, permitiendo una visualización de los objetos detectados. Esta implementación demuestra la viabilidad de utilizar arquitecturas ARM para aplicaciones de visión artificial en sistemas autónomos, resaltando su eficiencia energética y capacidad de procesamiento.

Palabras Clave: Navegación Autónoma, Aprendizaje Profundo, Procesamiento de Imágenes

ABSTRACT.

In the automotive industry, computer vision and deep learning technologies have revolutionized vehicle safety and efficiency, fostering the development of autonomous navigation systems. This paper presents an object detection system implemented on an ARM architecture, using a front-facing RGB camera to capture images of the environment. The images are serialized using Google Protocol Buffers and transmitted to a computation module using eCAL (Enhanced Communication Abstraction Layer) as Protocol Buffer API. Image processing is performed using the YOLO (You Only Look Once) algorithm, known for its accuracy and speed in detecting objects. The results are displayed on a monitor, allowing a visualization of the detected objects. This implementation demonstrates the feasibility of using ARM architectures for machine vision applications in autonomous systems, highlighting its energy efficiency and processing capacity.

Keywords: Autonomous Driving, Deep Learning, Image Processing

1. INTRODUCCIÓN

La industria automotriz ha experimentado una transformación derivada de la integración e innovación de tecnologías avanzadas como la visión artificial y el aprendizaje profundo. Estas tecnologías no solo han mejorado la seguridad y la eficiencia en los vehículos, sino que también han sentado las bases para la implementación de sistemas de navegación autónoma. La visión artificial permite a los vehículos percibir y comprender su entorno, mientras que el aprendizaje profundo proporciona la capacidad necesaria para analizar grandes volúmenes de datos visuales y tomar decisiones incluso en tiempo real.

En el ámbito de la seguridad vehicular, los sistemas avanzados de asistencia al conductor (ADAS, por sus siglas en inglés) utilizan la visión artificial para funciones como el reconocimiento de señales de tráfico, la detección de peatones y ciclistas, así como advertencias de posible colisión. Estudios recientes han demostrado que la implementación de ADAS puede reducir significativamente la cantidad de accidentes de tráfico y mejorar la seguridad en el camino [1].

Por otro lado, los algoritmos de aprendizaje profundo, en particular las redes neuronales convolucionales (CNN, por sus siglas en inglés), han revolucionado la precisión y la velocidad de los sistemas de detección de objetos. Por ejemplo, en el modelo YOLO (*You Only Look Once*) que muestra ser muy eficaz para la identificación y clasificación de objetos con capacidad de ofrecer implementaciones de respuestas en tiempo real, lo cual es crucial para la navegación autónoma [2], esto ha aumentado la capacidad de procesar imágenes de alta resolución y extraer características complejas que permite a los vehículos autónomos operar de manera segura y eficiente en entornos dinámicos y complejos.

La industria automotriz también se beneficia de la visión artificial y el aprendizaje profundo para mejorar la experiencia del usuario. Adicionalmente, los sistemas de monitoreo del conductor, que emplean estas tecnologías, pueden detectar signos de fatiga o distracción, y alertar al conductor o tomar medidas correctivas automáticas para prevenir accidentes [3].

Además, estas tecnologías se utilizan en el desarrollo de interfaces hombre-máquina (HMI) avanzadas, que facilitan una interacción más intuitiva y segura entre el conductor y el vehículo.

En la optimización de las operaciones logísticas y de fabricación dentro de la industria automotriz, también se aplican la visión artificial y el aprendizaje profundo. La inspección automatizada de piezas y el control de calidad, son áreas donde estas tecnologías han demostrado ser extremadamente efectivas, mejorando la precisión y reduciendo los costos operativos [4].

El sistema propuesto utiliza una cámara RGB frontal para capturar imágenes del entorno, las cuales son serializadas mediante Google *Protocol Buffers* para su eficiente transmisión. Una vez recibidas por el módulo de cómputo, las imágenes son procesadas utilizando el algoritmo de detección de objetos YOLO (You Only Look Once). Este enfoque permite detectar en tiempo real con alta precisión y baja latencia, lo cual es esencial para las aplicaciones de navegación autónoma.

En este trabajo, se propone un sistema modular capaz de procesar imágenes, mediante aprendizaje profundo e incluso grabar sesiones para un prototipado eficiente y escalable. La implementación demuestra la viabilidad de utilizar arquitecturas ARM, conocidas por su eficiencia energética y su potencia de procesamiento, para aplicaciones de visión artificial en sistemas autónomos y embebidos. Los resultados obtenidos sugieren que esta combinación de tecnologías puede ser una solución efectiva y económica para mejorar la capacidad de percepción y diseñar prototipos escalables para sistemas de navegación autónoma.

2. DISEÑO DEL SISTEMA

El sistema consiste en 3 etapas, como se muestra en la Fig. 1, en la etapa de adquisición se captura cada imagen a procesar con una cámara RGB, la cual es enviada mediante la API eCAL. Como segunda etapa, se decodifica cada imagen recibida por el módulo de detección y se ejecuta el sistema de detección mediante YOLO para enviar el resultado de manera codificada y serializada, por último, el bloque de visualización decodifica y proporciona una guía visual. A continuación, se describirá con mayor detalle cada etapa de la propuesta.

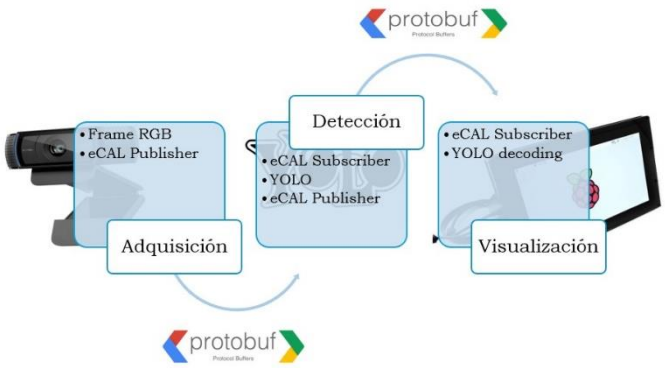

Fig. 1 Diagrama a bloques de la propuesta.

2.1. Adquisición

Por medio de una cámara RGB se adquiere una imagen, la cual será codificada según la aplicación y ancho de banda necesarios, presentando 3 opciones:

- *Sin Compresión:* Los datos solamente se serializan, mayor ancho de banda, menor costo computacional.
- *Compresión LZ4:* Algoritmo de compresión sin pérdida que se utiliza para reducir el tamaño de datos, incluidas las imágenes. Funciona detectando y eliminando redundancias dentro de los datos. LZ4 busca secuencias repetitivas y las reemplaza con referencias a una copia anterior de la misma secuencia, se mantiene integridad de la imagen y reduce uso de ancho de banda, aumenta costo computacional.
- *Compresión JPEG2000:* Compresión de imágenes con pérdida. Utiliza la transformada wavelet. El proceso implica dividir la imagen en sub-bandas de diferentes resoluciones, lo que facilita la compresión en múltiples niveles de calidad. Costo computacional más alto, menor uso de ancho de banda.

Seleccionada previamente una compresión, se serializa la información y se forma el mensaje mediante *Protocol Buffers*. Posteriormente se crea la instancia para el envío de información (eCAL publisher).

2.2. Detección

En el núcleo de este módulo se encuentra YOLO (You Only Look Once), un avanzado algoritmo de detección de objetos que se destaca por su capacidad para realizar detección en tiempo real. YOLO es una técnica de detección de objetos basada en redes neuronales profundas, específicamente en redes neuronales convolucionales (CNN, por sus siglas en inglés). Su principal innovación radica en cómo aborda la tarea de identificar y localizar objetos dentro de una imagen.

El funcionamiento de YOLO se basa en una estrategia integral que divide una imagen en una cuadrícula de celdas. Cada celda de esta cuadrícula es responsable de detectar los objetos que se encuentran en su área específica. Este proceso se realiza de manera simultánea a través de una única red neuronal convolucional, lo cual contrasta con enfoques más tradicionales que aplican múltiples etapas para la detección y localización de objetos.

La red neuronal convolucional que emplea YOLO es un modelo complejo que ha sido entrenado previamente utilizando grandes conjuntos de datos con anotaciones detalladas sobre las posiciones y categorías de los objetos. Durante el entrenamiento, el modelo aprende a reconocer patrones y características visuales que son indicativos de diferentes clases de objetos. Esta capacidad para extraer y entender características complejas de las imágenes es una de las principales ventajas de las redes

neuronales profundas en comparación con métodos más tradicionales.

Cuando una imagen es procesada por YOLO, la red convolucional realiza varias tareas simultáneamente. Los pasos principales se ilustran en la Fig. 2. Primero, divide la imagen en una cuadrícula de celdas, cada una de las cuales realiza una predicción sobre la presencia de objetos. Dentro de cada celda, el algoritmo genera un número fijo de cuadros delimitadores (bounding boxes), cada uno con una probabilidad asociada de contener un objeto. Además de la probabilidad de presencia, cada cuadro delimitador incluye información sobre la ubicación del objeto (generalmente representada por coordenadas en relación con la celda) y su categoría o clase.

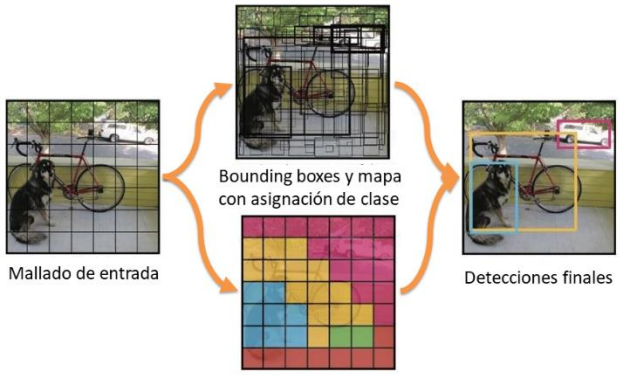

Fig. 2. Diagrama de flujo para YOLO

Este enfoque de predicción simultánea para toda la imagen permite que YOLO realice detección de objetos con capacidad de tiempo real y alta precisión.

La relación entre YOLO y el aprendizaje profundo es fundamental, ya que el algoritmo se basa en la capacidad de las redes neuronales convolucionales para aprender representaciones jerárquicas de datos visuales. Las CNN son capaces de capturar características de bajo nivel como bordes y texturas en las primeras capas, y características de alto nivel como formas y objetos en las capas más profundas. Esta capacidad de abstracción permite a YOLO identificar objetos y sus ubicaciones con una alta precisión.

El módulo de detección funcionará solo bajo demanda, por cada imagen recibida deberá ejecutar YOLO en su versión 8 [5]. Operará con 2 instancias de comunicación creadas para el uso de eCAL: una instancia *Subscriber* que se encargará de decodificar la imagen recibida y una instancia *Publisher* para el envío de información al módulo siguiente que contenga las detecciones realizadas.

2.3. Visualización

Este módulo recibirá las detecciones e imagen para formar una visualización que permita monitorear el proceso y brindar contexto sobre los resultados del módulo de detección. Para ello es necesaria la decodificación de los resultados obtenidos de YOLO, que consiste en dibujar los marcos por detección y asignarle un color único por clase.

3. IMPLEMENTACIÓN

En este trabajo se definió el uso de la arquitectura ARM, la cual es un conjunto de arquitecturas de procesador basadas en un diseño RISC (*Reduced Instruction Set Computer*), conocido por su eficiencia energética y alto rendimiento. Es ampliamente utilizada en dispositivos móviles, embebidos y otros sistemas donde la eficiencia energética es crucial.

3.1. Software

El diseño propuesto se desarrolló en lenguaje Python por su adaptabilidad con diversos sistemas operativos y arquitecturas, siendo compatibles las librerías y dependencias con la arquitectura ARM, además de mostrar facilidad para una posible migración a lenguaje C o C++.

Para el procesamiento de imágenes y aprendizaje profundo, se utilizaron OpenCV y Ultralytics [6] respectivamente, de esta forma se logra una manipulación de arreglos dinámicos respecto al uso de memoria, así como modelos previamente entrenados con el dataset COCO [7] para tener la capacidad de detectar hasta 80 clases distintas.

Como middleware se seleccionó eCAL, que es un software de código abierto diseñado para facilitar la comunicación entre componentes en sistemas distribuidos, especialmente en aplicaciones de robótica e industrial que proporciona una capa de abstracción que simplifica el intercambio de datos entre diferentes módulos de software [8], eCAL establece la funcionalidad necesaria para el envío y recepción de datos entre módulos de software, con independencia tecnológica como se ejemplifica en la Fig. 3.

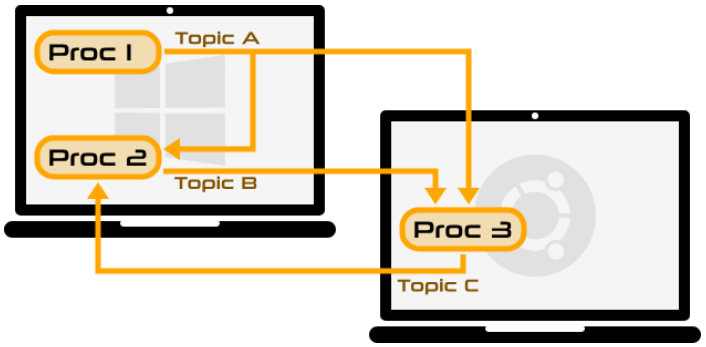

Fig. 3. Ejemplo de flujo con middleware eCAL

Para establecer la codificación y decodificación de los mensajes, se utiliza *Protocol Buffers*, siendo un lenguaje de serialización de datos desarrollado por Google, que permite definir estructuras de datos y serializarlas de manera eficiente para el intercambio

entre aplicaciones. Además, es utilizado para comunicaciones de red y de almacenamiento de datos [9]. En el código 1 se muestra la estructura de datos como está definida para enviar y recibir datos para reconstruir la imagen.

En la Fig. 4 se presenta el flujo o proceso para la integración de un archivo proto. Un mensaje o archivo proto contiene las variables a utilizar y el tipo de dato, estos códigos compilados mediante *Protocol Buffers* generan un nuevo archivo asociado a un lenguaje de programación, mientras que el numeral indica la posición que llevan los datos al serializarlos, mientras en la Fig. 4 se muestra el componente de monitoreo de eCAL, donde podemos observar la estructura de un mensaje enviado.

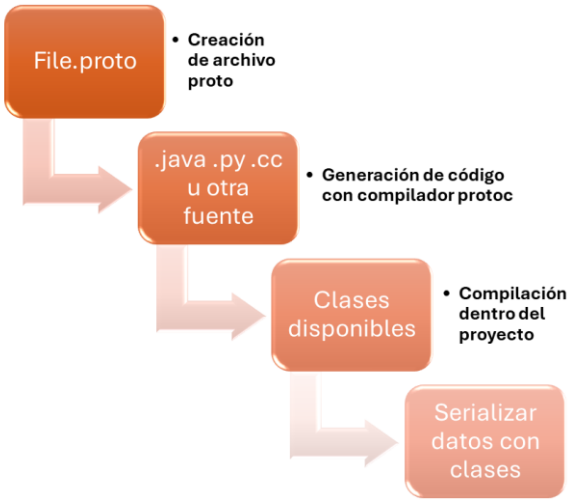

Fig. 4. Diagrama del proceso para la integración de un mensaje a partir de un archivo proto

Usando el archivo proto compilado para el lenguaje de programación objetivo, es posible utilizarlo como una clase en la cual las propiedades son los elementos del mensaje y los métodos funciones para serializar y codificar o decodificar el mensaje, mientras que en eCAL podemos crear un publisher y un subscriber para enviar y recibir el mensaje respectivamente.

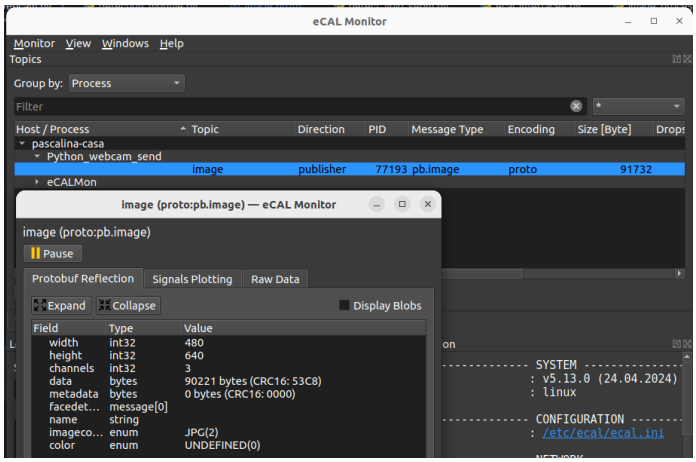

Fig. 5 eCAL monitor enviando imágenes mediante un mensaje proto predefinido

3.2. Hardware

Se seleccionó la microcomputadora Raspberry Pi 4 para la implementación debido a su arquitectura ARM y capacidad de cómputo para funcionar como uno de los módulos propuestos, especialmente la adquisición de datos, donde su tamaño y bajo consumo la hacen ideal para prototipado. En la Fig. 6 se muestra la conexión durante una prueba de manejo realizada.

Fig. 6. Captura de video con vehículo en movimiento.

eCAL tiene funcionalidad para el almacenamiento de datos, esto es que los mensajes enviados pueden ser persistidos para futuras pruebas, facilitando el proceso de prototipado. Para esta propuesta se realizaron 4 recorridos, almacenando las imágenes con compresión JPEG2000. Cada grabación tiene una duración que va desde los 45 segundos hasta los 2 minutos. El almacenamiento se realiza bajo archivos H5. En la Fig. 7 se muestra el resultado de utilizar YOLO con segmentación semántica.

Fig. 7. Visualización de la propuesta

En la Tabla 1 se muestran los tiempos de procesamiento obtenidos para 3 diferentes implementaciones. Cabe resaltar que el costo computacional es muy elevado para un solo dispositivo Raspberry Pi 4, sin embargo, es prometedor los alcances que pueden obtenerse con equipos de mayor capacidad manteniendo la arquitectura ARM como objetivo de implementación

TABLA I Tiempos de procesamiento para detección de objetos por frame

Equipo	Tiempo por frame (segundos)	Arquitectura CPU
Raspberry Pi 4	3.2	ARM
Intel i7-7700	0.114	X86 (64 bits)

4. CONCLUSIONES

En este trabajo, se han abordado tres aspectos fundamentales para la implementación en hardware específico. Primero, se ha implementado un sistema de detección de objetos que aprovecha las capacidades de hardware especializado, lo que permite una operación eficiente y rápida. Segundo, se ha propuesto una metodología que facilita una implementación sencilla y de bajo costo, accesible incluso para proyectos con recursos limitados. Tercero, se ha diseñado la metodología con capacidad escalable, que puede adaptarse a diferentes niveles de complejidad y a diversas necesidades de aplicación.

La implementación en hardware especializado además de mejorar la velocidad de procesamiento, también reduce la latencia y el consumo energético, lo cual es crucial para aplicaciones en tiempo real.

La propuesta metodológica presentada en este trabajo se centra en la simplicidad y el bajo costo. Se ha diseñado un enfoque que utiliza herramientas y recursos accesibles. El proyecto puede ser replicado para arquitecturas ARM y/o x86 usando el repositorio en [10].

Como trabajo a futuro, un área prometedora de desarrollo es la integración de los sistemas de detección de objetos con módulos de control para la toma de decisiones automatizada. Esta integración permitirá que la información obtenida a partir de la detección de objetos se envíe a un módulo de control, con capacidad de analizar los datos y tomar decisiones basadas en algoritmos predefinidos.

Respecto al procesamiento, en posteriores desarrollos se plantea el uso de GPU embebido y bajo consumo o bien una microcomputadora ARM de mayores prestaciones para mejorar los tiempos de ejecución en el módulo de detección, por ejemplo, con una implementación con Raspberry Pi 5. Además de evaluar la cuantización del modelo para implementaciones con menor costo computacional.

5. REFERENCIAS

[1] W. M. D. Chia, S. L. Keoh, C. Goh and C. Johnson, "Risk Assessment Methodologies for Autonomous Driving: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 10, pp. 16923-16939, Oct. 2022, doi: 10.1109/TITS.2022.3163747.

[2] J. S. Murthy, G. M. Siddesh, W.-C. Lai, B. D. Parameshachari, S. N. Patil, and K. L. Hemalatha, ‘ObjectDetect: A Real-Time Object Detection Framework for Advanced Driver Assistant Systems Using YOLOv5’, Wireless Communications and Mobile Computing, vol. 2022, no. 1, p. 9444360, 2022.

[3] Y. Albadawi, M. Takruri, and M. Awad, ‘A Review of Recent Developments in Driver Drowsiness Detection Systems’, Sensors, vol. 22, no. 5, 2022.

[4] Chen, D. Jin, H. He, F. Yang and J. Yang, "Deep Learning Based Online Nondestructive Defect Detection for Self-Piercing Riveted Joints in Automotive Body Manufacturing," in IEEE Transactions on Industrial Informatics, vol. 19, no. 8, pp. 9134-9144, Aug. 2023, doi: 10.1109/TII.2022.3226246

[5] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, ‘A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS’, Machine Learning and Knowledge Extraction, vol. 5, no. 4, pp. 1680–1716, 2023.

[6] M. Sohan, T. Sai Ram, and C. V. Rami Reddy, ‘A Review on YOLOv8 and Its Advancements’, in Data Intelligence and Cognitive Informatics, 2024, pp. 529–545.

[7] T.-Y. Lin et al., ‘Microsoft COCO: Common Objects in Context’, in Computer Vision -- ECCV 2014, 2014, pp. 740–755.

[8] A. Loeffler, R. Zergiebel, J. Wache and M. Mejdoub, "Advances in Automotive Radar for 2023," 2023 24th International Radar Symposium (IRS), Berlin, Germany, 2023, pp. 1-8, doi: 10.23919/IRS57608.2023.10172436.

[9] C. Currier, ‘Protocol Buffers’, in Mobile Forensics -- The File Format Handbook: Common File Formats and File Systems Used in Mobile Devices, C. Hummert and D. Pawlaszczyk, Eds. Cham: Springer International Publishing, 2022, pp. 223–260.

[10] V. A. Gonzalez-Huitron, N. J. Beltrán Valenzuela, J. R. De Mateo Gonzalez, and I. T. Alcantara Llanas, YOLO Object detection employing ARM and eCAL. https://github.com/alx3416/Soa-2024-1.