Sistema de asistencia visual basado en ESP32-CAM

José Eleazar Peralta López[1], José Adrián Rojas Jaime[1], Víctor Hugo Cárdenas García[1]
[1] Instituto Tecnológico Superior de Salvatierra. joperalta@itess.edu.mx, at22110381@itess.edu.mx, at22110288@itess.edu.mx

Resumen

La inclusión de sistemas de visión artificial en autos y modelos está presente hoy en día con investigaciones y proyectos, pero siempre suponen un reto ya que son sistemas muy robustos y complejos, basados en arquitecturas muy complejas teniendo en cuenta que son difíciles de conseguir ya que vienen de proveedores muy específicos. El presente proyecto describe el desarrollo de un sistema de asistencia visual con una cámara ESP32-CAM. El sistema integra la captura y transmisión de video mediante una red WiFi local con el procesamiento de imágenes en Python utilizando Opencv. La imagen recibida es analizada mediante segmentación en el espacio de color HSV para distinguir la superficie del piso de posibles obstáculos. Para mejorar la robustez se implementaron una región de interés (ROI), filtros morfológicos y análisis de contornos.

Abstract
The inclusion of computer vision systems in vehicles and vehicle prototypes is currently supported by various research studies and development projects. However, these systems always present a significant challenge, as they are highly robust and complex, network, with image processing performed in Python using OpenCV. The received image is analyzed through segmentation in the HSV color space to distinguish the floor surface from potential obstacles. To improve system robustness, a Region of Interest (ROI) is implemented, along with morphological filtering and contour analysis.

Palabras clave ESP32-CAM, visión artificial, OpenCV, detección de obstáculos, HSV, ROI.

Introducción

Hoy en día la industria 4.0 requiere sistemas que ayuden y faciliten la conducción del habitáculo previniendo accidentes y riesgos. La literatura reciente [1-5] ofrece análisis exhaustivos del rendimiento de las plataformas embebidas para tareas de visión artificial en entornos IoT, Aunque hay una falta de metodología estandarizada para integrar algoritmos de aprendizaje profundo en sistemas embebidos sin depender de fabricantes específicos, se han propuesto entornos para generalizar este proceso.

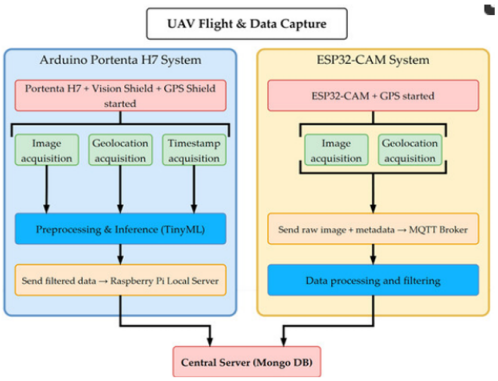


Fig. 1 the figure shows the workflow diagram of the two systems. On the left is the main system based on Arduino Portenta H7, and on the right is the alternative system based on ESP32-CAM. [2]

En este proyecto se planteó la integración de una ESP32-CAM en un modelo de vehículo móvil para proporcionar percepción visual del entorno La propuesta evolucionó a partir de la necesidad de utilizar los recursos disponibles de la ESP32 CAM de una manera práctica. En lugar de realizar todo el procesamiento de visión directamente en el dispositivo, se optó por transmitir el video hacia Python y realizar allí el procesamiento mediante OpenCV. Esta arquitectura separa las funciones de adquisición y comunicación de las funciones de procesamiento de imágenes y control. Referente a los problemas con la utilización de ESP32-CAM mencionados en [3] nosotros encontramos desafíos similares, que fueron la superficie utilizada para las pruebas estaba formada por mosaicos. Aunque el piso presentaba características de color que permitían diferenciarlo de obstáculos más oscuros, las líneas divisoras entre mosaicos producían cambios de tonalidad que podían interpretarse como obstáculos. A esto se sumaron variaciones de iluminación que afectaban los valores observados por la cámara.

El objetivo general fue desarrollar un sistema de asistencia visual que permitiera identificar posibles obstáculos que interfirieran con su camino. Como objetivos específicos se planteó establecer la comunicación entre la ESP32-CAM y Python, obtener el flujo de video, determinar los valores de color apropiados para la superficie de operación, generar una máscara de segmentación, reducir falsos positivos, detectar regiones correspondientes a obstáculos.

Metodología

A. Arquitectura general:



Fig. 2 flujo de trabajo

B. Hardware:

La unidad principal es una ESP32-CAM instalada sobre un vehículo móvil. En el código proporcionado se utilizan LEFT_M0 = 13, LEFT_M1 = 12, RIGHT_M0 = 14 y RIGHT_M1 = 15 para las señales de control de motores, además de una salida para la iluminación. . The figure shows the workflow diagram of the two systems. On the left is the main system based on Arduino Portenta H7, and on the right is the alternative system based on ESP32-CAM

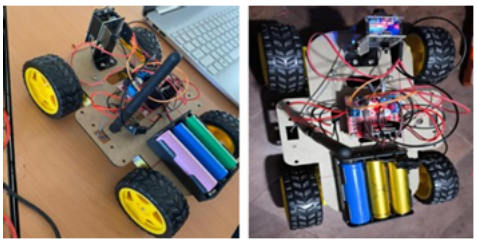


Fig. 3 Modelo con la ESP32-CAM

C. servidores WEB, comunicación WIFI y HSV:

La ESP32-CAM funciona como servidor web dentro de una red WiFi local. El servidor registra rutas o endpoints y asocia cada ruta con una función. Por ejemplo, una solicitud GET a /go ejecuta go_handler(), que

llama a robot_fwd(). La URL funciona como dirección del recurso; la lógica está implementada en el handler de la ESP32-CAM. Transmisión y lectura del video La ESP32-CAM transmite imágenes JPEG mediante el endpoint /stream usando multipart/x-mixed-replace (MJPEG). Python accede al flujo, obtiene los datos de cada imagen y utiliza OpenCV para reconstruir y decodificar los fotogramas. Esto permite que la ESP32-CAM se concentre en adquisición y transmisión, mientras Python realiza el procesamiento. Obtención de valores HSV Se desarrolló una herramienta para seleccionar píxeles sobre la imagen y observar valores RGB y HSV. Se realizaron múltiples selecciones sobre el piso para conocer la variación de sus componentes de color bajo las condiciones reales de iluminación. Se observaron, entre otros, valores HSV cercanos a H=35, S=9–14 y V=142–166. Segmentación del piso mediante HSV Se utilizó el espacio HSV para separar tonalidad, saturación y brillo. Los parámetros finales documentados fueron: lower_floor = [20, 0, 80] upper_floor = [65, 60, 255] El rango se amplió para tolerar variaciones de iluminación y reducir falsos obstáculos generados por el piso.



Fig. 4 Comunicación, transmisión y obtención de valores HSV.

D. región de interés:

La ROI limita el análisis a la zona relevante de la imagen. El parámetro roi_start define desde qué posición vertical comienza la zona analizada. El valor final documentado es roi_start = 60. Esta delimitación reduce procesamiento y falsos positivos. Filtrado morfológico Después de obtener la máscara se aplican operaciones morfológicas para reducir ruido, regiones aisladas y discontinuidades. El código documentado utiliza un kernel de 5 × 5 píxeles.

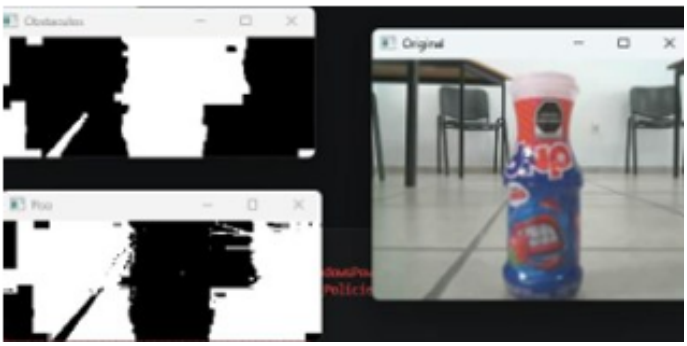


Fig. 5 Región de interés y filtrado morfológico.

Resultados

El resultado final fue un sistema funcional que permite recibir el video de la ESP32-CAM en Python, procesarlo mediante OpenCV y visualizar la escena con información de detección. La combinación de HSV, ROI y filtrado morfológico redujo los falsos positivos asociados con mosaicos e iluminación.

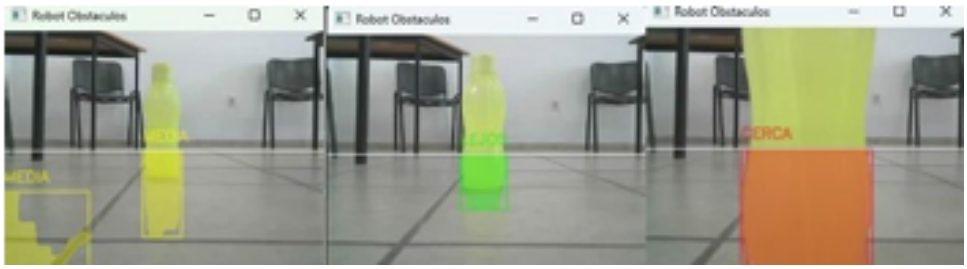


Fig. 6 resultados visuales.

Conclusiones y trabajos futuros:

El proyecto permitió integrar ESP32-CAM, comunicación WiFi, servidor web, Python y OpenCV en un vehículo móvil con asistencia visual, la arquitectura separa adquisición/transmisión y procesamiento de visión. El uso de HSV permitió adaptar la segmentación a la superficie de prueba. Las condiciones reales demostraron que iluminación y textura pueden producir falsos positivos, por lo que fue necesario complementar la segmentación con ROI, operaciones morfológicas y análisis de contornos. Como trabajo futuro se plantea mejorar la robustez ante iluminación variable, incorporar estimación de distancia y explorar métodos de detección más avanzados.

Referencia

Contreras Pérez, A. S., & Machado Olivero, A. F. (2025). *Sistema de semaforización inteligente para el tránsito vehicular seguro sobre la Carrera 15* [Trabajo de grado de pregrado, Universidad Católica de Colombia]. Repositorio Institucional de la Universidad Católica de Colombia.

García-González, J., Gómez Muñoz, C. Q., Gachet Páez, D., & Sánchez-Soriano, J. (2025). Automated road data collection systems using UAVs: Comparative evaluation of architectures based on Arduino Portenta H7 and ESP32-CAM. *Electronics*, *14*(21), 4165. https://doi.org/10.3390/electronics14214165

Piedrahita Sánchez, J. C., & Herrera Zapata, J. A. (2025). *Diseño de un prototipo para control del tráfico vehicular en el campus universitario de la UCEVA* [Trabajo de grado de pregrado, Unidad Central del Valle del Cauca].

Sandoval Herrera, J. D. (2024). *Diseño de una red neuronal para evitar colisiones de vehículos prototipo* [Trabajo de titulación de grado, Universidad Politécnica Salesiana]. Repositorio UPS.

Villacreses Suárez, N. J. (2025). *Prototipo de semáforo inteligente para promover las buenas prácticas de educación vial a motociclistas del cantón Jipijapa* [Trabajo de titulación de grado, Universidad Estatal del Sur de Manabí]. Repositorio UNESUM.