CEA Comité Español de Automática
XLVII JORNADAS DE AUTOMÁTICA CÓRDOBA 2026
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Diseño e implementación de un sistema de detección y localización de objetos en tiempo real mediante YOLOv8n y ROS2

Asier Reig-Lozano[a,*], David Martínez-Pascual[a], Raúl Martín-Batanero[a], Daniel Rodríguez-López[a], Luís D. Lledó[a], Nicolás García-Aracil[a]

[a] *Grupo de Robótica e Inteligencia Artificial, Instituto de Bioingeniería de Elche, Universidad Miguel Hernández, Avenida de la Universidad, s/n. 03202 Elche, Alicante, España*

To cite this article: Reig-Lozano, A., Martínez-Pascual, D., Martín-Batanero, R., Rodríguez-López, D., Lledó, L. D., García-Aracil, N. 2026. Design and implementation of a real-time object detection and localization system using YOLOv8n and ROS2. Jornadas de Automática, 47. https://doi.org/10.17979/ja-cea.2026.47.13754

Resumen

Este artículo presenta el diseño e implementación de un sistema de percepción visual en tiempo real para robótica de asistencia bimanual. Mediante el reentrenamiento de un modelo YOLOv8-nano con un conjunto de datos propio, etiquetado manualmente, y su posterior integración en el ecosistema ROS2 utilizando una cámara de profundidad RGB-D, se logra la detección y localización tridimensional de objetos cotidianos. El sistema se encapsula en contenedores Docker para garantizar su portabilidad y reproducibilidad en diferentes plataformas de hardware. Los resultados experimentales demuestran un rendimiento óptimo en términos de precisión computacional y estimación espacial. Se valida así una arquitectura modular capaz de equilibrar la velocidad de inferencia y la robustez del modelo ante variaciones del entorno mediante técnicas de aumento de datos, sentando las bases operativas para el control tridimensional de exoesqueletos.

Palabras clave: Tecnología robótica, Percepción y detección, Validación de modelos, Aprendizaje automático, Procesamiento de imágenes, Algoritmos en tiempo real.

Design and implementation of a real-time object detection and localization system using Yolov8n and Ros2

Abstract

This paper presents the design and implementation of a real-time visual perception system for bimanual assistive robotics. By retraining a YOLOv8-nano model with a custom, manually labeled dataset and its subsequent integration into the ROS2 ecosystem using an RGB-D depth camera, three-dimensional detection and localization of daily objects are achieved. The system is encapsulated in Docker containers to guarantee portability and reproducibility across different hardware platforms. Experimental results demonstrate optimal performance in terms of computational accuracy and spatial estimation. This validates a modular architecture capable of balancing inference speed and model robustness against environmental variations through data augmentation techniques, laying the operational foundations for the three-dimensional control of upper-limb exoskeletons.

Keywords: Robotics technology, Perception and sensing, Model validation, Machine Learning, Image processing, Real-time algorithms.

1. Introducción

El envejecimiento demográfico y el aumento progresivo de la esperanza de vida en los países desarrollados han traído consigo un incremento en la incidencia de patologías crónicas que afectan severamente la autonomía personal (Izzo et al., 2018),(Pantoni et al., 2009). Entre las causas más comunes de discapacidad motora y cognitiva en la edad adulta se encuentran los Accidentes Cerebrovasculares (ACVs), así como las

*Autor para correspondencia: asier.reig@goumh.umh.es

lesiones de la médula espinal o del sistema nervioso central (Salomon et al., 2012). Estas condiciones provocan un deterioro notable en la calidad de vida de los pacientes al limitar su capacidad para desenvolverse con independencia (Fares et al., 2021).

En este escenario, la robótica de asistencia ha emergido como una alternativa tecnológica fundamental para apoyar a personas con déficits motores durante la realización de las actividades de la vida diaria (AVDs) (Park et al., 2020). Dentro de la literatura científica, las soluciones más extendidas se basan en el uso de manipuladores externos y, especialmente, de exoesqueletos (Frisoli et al., 2022). Estos dispositivos no solo permiten mejorar la autonomía del usuario, sino que han demostrado resultados prometedores tanto en entornos industriales como en el ámbito de la rehabilitación.

Bajo esta premisa, la investigación actual busca impulsar el desarrollo de interfaces de usuario avanzadas para el control de dispositivos de asistencia (Irles et al., 2024). Esta línea de trabajo da continuidad a iniciativas previas orientadas al diseño de arquitecturas multimodales, modulares y adaptativas que se ajustan a las necesidades individuales y a la diversidad funcional de cada usuario (Crea et al., 2018),(Catalán et al., 2023).

En este contexto, la optimización y el guiado de un exoesqueleto bimanual de miembro superior requieren de esquemas de control compartido hombre-máquina que integren la información del entorno con las intenciones del usuario. Para alcanzar este nivel de interacción bimanual, se vuelve indispensable incorporar un módulo avanzado de visión artificial. Esto actúa como el núcleo perceptivo del sistema, permitiendo reconocer actividades, procesar mapas de profundidad e interpretar el entorno tridimensional en tiempo real. De este modo, la información visual se traduce de forma precisa y robusta en coordenadas espaciales que guían las trayectorias del manipulador robótico, cerrando la brecha entre la percepción del entorno y la asistencia física directa al paciente.

Figura 1: Sistema robótico bimanual de tipo exoesqueleto con detección de objetos mediante cámara RGB-D.

Ante la necesidad de superar las limitaciones de los métodos actuales en estos escenarios controlados, el presente artículo plantea como objetivo general desarrollar un sistema de percepción visual en tiempo real capaz de detectar y localizar objetos tridimensionalmente mediante técnicas modernas de visión y el reentrenamiento (*fine-tuning*) de un modelo de aprendizaje profundo integrado con un sensor de profundidad. Específicamente, el trabajo contempla: 1) recolectar y etiquetar un dataset propio con Label Studio; 2) reentrenar y evaluar un modelo base de la familia YOLO; 3) encapsular el entorno en contenedores Docker para garantizar un despliegue reproducible; y 4) desarrollar un nodo en ROS2 acoplado a una cámara RGB-D para estimar la posición espacial (3D) de los objetos para su futura manipulación robótica.

2. Materiales y métodos

2.1. *Preparación y etiquetado del dataset con Label Studio*

Para poder desarrollar un modelo de detección de objetos, es necesario nutrirlo previamente de un conjunto de datos visuales tales como imágenes para que el algoritmo pueda detectar patrones comunes entre ellas, con el fin de reconocer clases u objetos. Por defecto, YOLOv8n está preentrenado con el dataset COCO, por lo que es capaz de detectar objetos comunes, pero para adaptarlo a nuestras clases, fue necesario crear un conjunto de datos propio. Para ello, se capturaron imágenes de los diferentes objetos utilizando una cámara Logitech C930e 1080p en diferentes ángulos y condiciones de iluminación, con el fin de representar diferentes escenarios. En total, se recolectaron 164 imágenes, que posteriormente fueron etiquetadas para identificar los objetos *taza, botella y cuchara*. Para conseguir enriquecer el conjunto de entrenamiento y evitar un sobreajuste, se aplicaron técnicas de aumentación de datos con el objetivo de ayudar a que el modelo generalice mejor.

Para recolectar datos y clasificarlos, se utilizó Label Studio, una plataforma de etiquetado de datos para validar modelos de inteligencia artificial, la cual permite etiquetar datos como audio, texto, imágenes, vídeos y series temporales de manera sencilla con una interfaz de usuario simple y directa, y exportar a varios tipos de modelo (Tkachenko et al., 2020). Para cada una de las imágenes, se etiquetaron manualmente todas las instancias de los objetos de interés, delimitando su posición mediante cajas delimitadoras (bounding boxes) y se le asignó una etiqueta con su clase. Una vez finalizado este proceso y con todo el conjunto de imágenes etiquetado, el dataset fue exportado en formato YOLO, el cual contiene un archivo de texto por imagen con la clase y las coordenadas normalizadas de cada objeto etiquetado.

2.2. *Entrenamiento del modelo de detección de objetos*

El entrenamiento del modelo constituye una de las fases más relevantes ya que de él depende directamente la capacidad del sistema para detectar y localizar correctamente los objetos de interés en tiempo real. Se optó por entrenar un modelo de detección de objetos basado en YOLOv8n, utilizando el conjunto de datos propio y aplicando diversas técnicas destinadas a mejorar la generalización del modelo.

Se seleccionó la arquitectura YOLOv8n (nano) debido a su óptimo equilibrio entre velocidad de inferencia y precisión

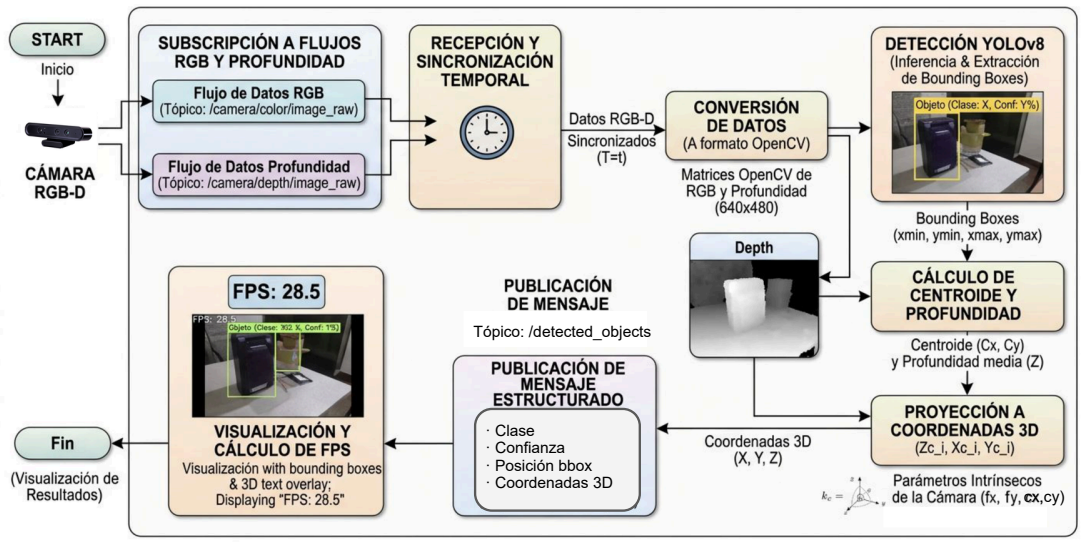

Figura 2: Estructura del nodo ROS2, desde la recepción de la imagen con la cámara hasta la detección de los objetos.

en entornos de tiempo real (Jocher et al., 2023). El conjunto de datos se dividió en subconjuntos de entrenamiento (70 %), validación (15 %) y prueba (15 %) para prevenir el sobreajuste. El proceso de ajuste (*fine-tuning*) se inició a partir de los pesos preentrenados del dataset COCO, lo que optimiza la convergencia al trabajar con muestras reducidas. El entrenamiento se ejecutó en GPU mediante CUDA durante 100 épocas, utilizando imágenes redimensionadas a 640 píxeles y un tamaño de lote (*batch size*) de 16.

Para mejorar la robustez del sistema frente a la variabilidad del entorno asistencial, se implementaron técnicas de aumentación de datos (*data augmentation*) en tiempo de ejecución. Estas incluyeron modificaciones en la escala, traslaciones, variaciones de color y brillo, así como técnicas avanzadas de mezcla de imágenes (*Mosaic* y *Random Erasing*). Esta estrategia permitió diversificar el dataset sin coste de etiquetado manual, con el fin de mejorar la capacidad de generalización del modelo ante oclusiones parciales y cambios de iluminación.

2.3. *Cámara de profundidad e integración con ROS2*

El siguiente paso es integrar el modelo en el entorno ROS2, haciendo uso de una cámara con sensor de profundidad. El objetivo es utilizar las ventajas que nos ofrece esta cámara para conocer la posición tridimensional respecto a la cámara de cada objeto detectado, para así poder alcanzarlo mediante el control de los brazos robóticos instalados en la silla.

Para la obtención de información espacial de los objetos detectados se empleó una cámara Orbbec Astra S, la cual integra un sensor RGB y un sensor de profundidad. Esta configuración permite capturar simultáneamente imágenes en color y mapas de profundidad, permitiendo estimar la posición tridimensional de los objetos en el espacio.

Para poder hacer uso de la cámara y su información, se desarrolló un nodo en ROS2 Humble (Figura 2) encargado de la detección de objetos y la estimación de su posición tridimensional. El nodo de detección fue implementado dentro de un paquete (*camera_detection*) de ROS2 desarrollado en Python. Al crear el paquete mediante las herramientas de ROS2 se genera automáticamente una estructura básica que incluye varios archivos de configuración necesarios para su compilación y ejecución dentro del ecosistema ROS.

El nodo desarrollado en ROS2 actúa como *subscriber* de los *topics* de la cámara vistos en la Figura 2, ya que la imagen RGB se utiliza como entrada para el modelo de detección YOLOv8n entrenado previamente mientras que la imagen de profundidad junto con la matriz intrínseca de la cámara, se emplea para calcular la distancia al objeto detectado.

Uno de los principales retos en sistemas RGB-D es la correcta alineación temporal entre la imagen RGB, la imagen de profundidad y la información de la cámara. Dado que cada *topic* puede publicarse con ligeras diferencias temporales, se implementó un mecanismo de sincronización en ROS2 empleando filtros temporales. En concreto, se hizo uso de la clase *ApproximateTimeSynchronizer* de ROS2 (perteneciente al paquete *message_filters*) para reunir mensajes que llegan desde diferentes tópicos y agruparlos si sus marcas de tiempo son cercanas entre sí.

Los mensajes de ROS2 se convierten a formato OpenCV, biblioteca que permite el procesamiento de imágenes en tiempo real, mediante la librería *CvBridge*. De esta manera, obtenemos la imagen RGB en formato «bgr8», mientras que la profundidad en «passthrough». Todo ello permite operar sobre matrices NumPy para la inferencia con YOLO y el acceso directo a valores de profundidad por píxel.

El modelo YOLOv8n previamente entrenado se carga durante la inicialización del nodo en ROS2. De este modo, el modelo permanece en memoria durante toda la ejecución, evitando penalizaciones por recarga en cada frame. Para cada imagen RGB recibida y sincronizada:

- Se ejecuta la inferencia del modelo.
- Se obtienen las coordenadas de las bounding boxes detectadas.
- Se extrae la clase inferida y la confianza asociada.
- Se calcula el centro de cada bounding box.

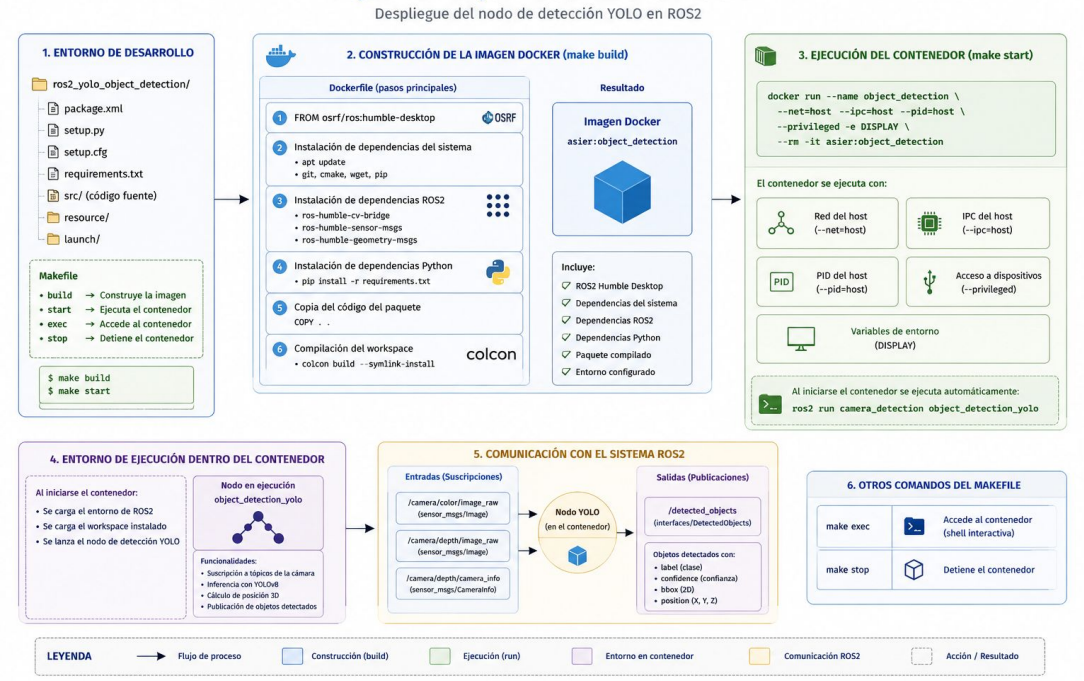

Figura 3: Flujo de trabajo del contenedor Docker desarrollado.

Este centro se utiliza como punto representativo del objeto para la obtención de la profundidad.

El centro geométrico se calcula, para cada bounding box detectada, de la siguiente manera:

$$u = \frac{x_{min} + x_{max}}{2} \tag{1}$$

$$v = \frac{y_{min} + y_{max}}{2} \tag{2}$$

La profundidad Z se obtiene directamente del mapa de profundidad de milímetros a metros:

$$Z = \frac{depth(u, v)}{1000} \tag{3}$$

Para poder proyectar a coordenadas tridimensionales, hay que hacer uso de la matriz intrínseca K, la cual podemos obtener directamente de unos de los topics de la cámara Orbbec Astra:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 570{,}34 & 0 & 319{,}5 \\ 0 & 570{,}34 & 239{,}5 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Para posteriormente obtener las coordenadas tridimensionales mediante la siguiente ecuación:

$$P_{3D} = K^{-1} \cdot \begin{bmatrix} u \cdot Z \\ v \cdot Z \\ Z \end{bmatrix} \tag{5}$$

De esta forma se obtiene la posición del objeto en el sistema de referencia de la cámara. Este procedimiento permite transformar detecciones bidimensionales en información espacial tridimensional, habilitando su posible integración en sistemas robóticos.

Además, cabe mencionar que se implementó un cálculo de frames por segundo (FPS), cuyo resultado se superpone visualmente sobre la imagen, siendo el promedio 30 FPS, lo que permite monitorizar el rendimiento en tiempo real y evaluar el impacto de la inferencia e influencia de la GPU (Figura 4).

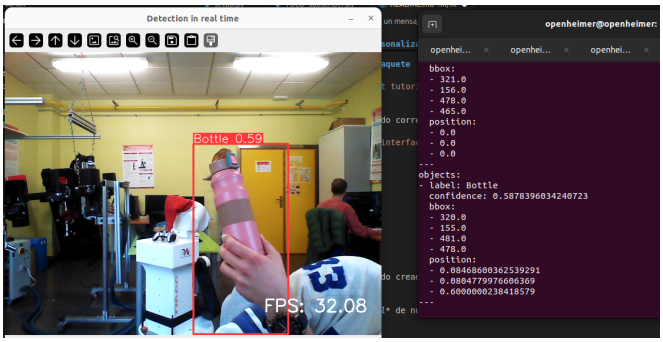

Figura 4: Detección en tiempo real con cámara Orbbec Astra.

2.4. *Contenerización del nodo de ROS2*

Para facilitar el despliegue del sistema desarrollado y garantizar su ejecución en distintos entornos sin dependencias externas, se procedió a la contenerización mediante *Docker*, encapsulando el sistema completo, incluyendo el nodo de ROS2, sus dependencias y el proceso de inferencia en tiempo real (Figura 3).

La contenerización del sistema se realizó mediante la creación de una imagen Docker personalizada. Dicha imagen incluye todos los elementos necesarios para la instalación de ROS2 Humble junto con herramientas de desarrollo y visualización. Para la construcción de la imagen se utilizó un *Dockerfile*, en el que se definieron los pasos necesarios para configurar el entorno de ejecución.

Además, para facilitar la ejecución del contenedor, se desarrolló un *Makefile* que automatiza las tareas principales del ciclo de vida del sistema. Mediante este archivo se simplifica

la creación de la imagen Docker y el lanzamiento del contenedor con acceso a ROS2, a la GPU, a la cámara y al sistema de visión. Este enfoque permite ejecutar el sistema completo mediante un único comando, reduciendo errores y mejorando la reproducibilidad del entorno.

2.5. *Métodos de evaluación del rendimiento del sistema*

Para poder evaluar la validez del modelo de detección de objetos, se seleccionaron métricas estándar basadas en la matriz de confusión: Precisión (P), Exhaustividad (*Recall*, R) y la Precisión Media Promedio (*Mean Average Precision*, mAP). La Precisión mide la fracción de detecciones correctas entre todas las predicciones realizadas, expresada como:

$$P = \frac{TP}{TP + FP} \tag{6}$$

donde TP representa los verdaderos positivos y FP los falsos positivos. Por su parte, el *Recall* cuantifica la capacidad del modelo para identificar todos los objetos reales presentes en la escena:

$$R = \frac{TP}{TP + FN} \tag{7}$$

donde FN denota los falsos negativos.

Como indicador global del rendimiento del clasificador, se emplea el mAP, calculado mediante el área bajo la curva Precisión-Recall promediada para las N clases del sistema:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} \int_0^1 P(R)\, dR \tag{8}$$

Específicamente, se analizan el $mAP@0{,}5$, que evalúa la concordancia con un umbral de Intersección sobre la Unión (IoU) del 50 %, y el $mAP@0{,}5 : 0{,}95$, que promedia el rendimiento en umbrales crecientes de confinamiento espacial (desde el 50 % hasta el 95 %). Finalmente, la eficiencia temporal del nodo se determina mediante la tasa de frames por segundo (FPS), garantizando la viabilidad del sistema en el bucle de control en tiempo real.

3. Resultados y discusión

3.1. *Evaluación del entrenamiento del modelo*

Una vez finalizado el proceso de entrenamiento del modelo YOLOv8n, se analizaron las métricas generadas automáticamente por el framework con el objetivo de evaluar tanto la capacidad de aprendizaje como el rendimiento del modelo sobre datos no vistos previamente. Durante el entrenamiento se monitorizaron las funciones de pérdida asociadas a: localización de bounding boxes (*box_loss*), clasificación (*cls_loss*) y distribución focal (*dff_loss*).

Las curvas obtenidas exhibieron una convergencia decreciente y estable a lo largo de las 100 épocas (Figura 5.a). La correspondencia entre las curvas de entrenamiento y validación descarta un sobreajuste severo. Los valores elevados de precisión y recall validan la idoneidad de las cajas generadas. El mAP@0.5 demuestra un desempeño sólido en el reconocimiento general de las tres clases, dejando un pequeño margen de optimización en el mAP@0.5:0.95 en lo respectivo al ajuste milimétrico de los contornos.

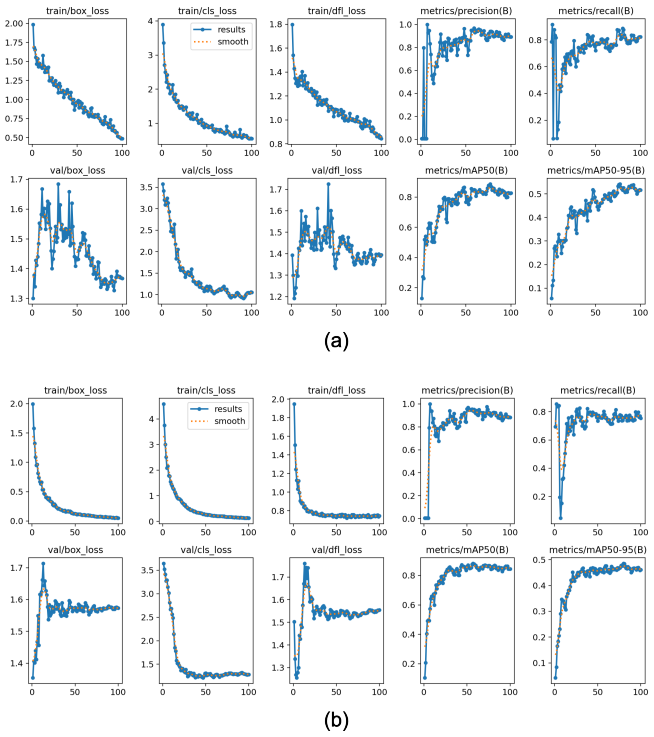

Figura 5: (a) Resultados entrenamiento del modelo YOLOv8n empleando técnicas de aumentación de datos; (b) Resultados entrenamiento del modelo YOLOv8n sin técnicas de aumentación de datos

Además, con el objetivo de analizar la infuencia de las técnicas de aumentación de datos se realizó un entrenamiento sin aplicar estas técnicas (Figura 5.b). A partir de los resultados obtenidos puede observarse que el uso de data augmentation no produce diferencias significativas en la precisión general del modelo, manteniéndose valores similares de precision y mAP@0.5. Sin embargo, sí se aprecia una mejora en las métricas de recall y mAP@0.5:0.95. Esto indica que el modelo entrenado con augmentation presenta una mayor capacidad de generalización y una localización más precisa de los objetos detectados. Además, las curvas de validación del entrenamiento con augmentation muestran un comportamiento más progresivo y estable, reduciendo el riesgo de sobreajuste sobre el conjunto de entrenamiento. Por tanto, el uso de técnicas de aumento de datos contribuye positivamente a la robustez del modelo frente a variaciones en las imágenes de entrada y mejora su comportamiento en escenarios no vistos durante el entrenamiento.

Para evaluar la precisión por clase del modelo reentrenado, se analizó la matriz de confusión normalizada (Figura 6). Los resultados cuantitativos muestran un desempeño sobresaliente en la clase *Bottle* con un tasa de verdadero positivo del 94 %, seguida de *Spoon* con un 86 % y *Cup* con un 82 %. Las discrepancias menores del modelo se concentran principalmente en la clasificación errónea de falsos negativos hacia el fondo (*background*), alcanzando un 18 % en las tazas y un 14 % en las cucharas. Este comportamiento es aceptable en el entorno asistencial propuesto, ya que el sistema prioriza omitir detecciones dudosas antes que generar falsos positivos interclase, garantizando la seguridad en la posterior manipulación robótica.

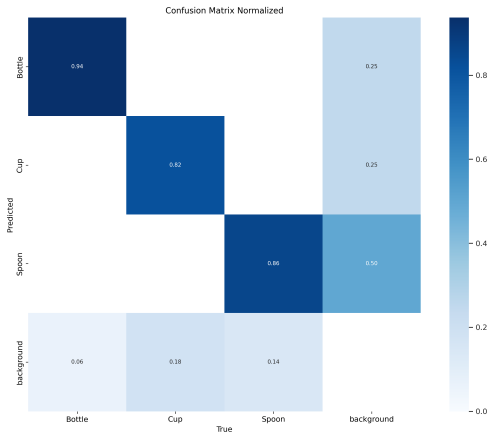

Figura 6: Matriz de confusión normalizada del modelo YOLOv8n.

3.2. Evaluación del sistema de percepción en tiempo real

Una vez entrenado el modelo, este fue integrado en el sistema de percepción desarrollado sobre ROS2 junto con la cámara RGB-D Orbbec Astra. El sistema completo fue capaz de:

1. Detectar objetos en tiempo real.
2. Obtener su profundidad.
3. Calcular coordenadas tridimensionales.
4. Publicar información estructurada mediante mensajes ROS2 personalizados.

Durante las pruebas realizadas, el sistema mostró un funcionamiento estable, manteniendo una detección continua de los objetos presentes en la escena. El cálculo de FPS integrado dentro del nodo permitió monitorizar el rendimiento en tiempo real durante la inferencia. La utilización de YOLOv8n, una variante ligera del modelo YOLOv8, permitió mantener un equilibrio adecuado entre precisión y velocidad de ejecución. Asimismo, la sincronización entre imágenes RGB y mapas de profundidad permitió obtener coordenadas tridimensionales coherentes para los objetos detectados.

La utilización de Docker tanto para el modelo de detección como para el nodo ROS2 permitió además simplificar considerablemente el despliegue del sistema completo, garantizando la reproducibilidad del entorno de ejecución y facilitando su portabilidad entre distintos equipos.

4. Conclusiones

El presente trabajo ha demostrado la viabilidad de un sistema de percepción visual en tiempo real que asocia de forma eficaz el reconocimiento de objetos por aprendizaje profundo con la localización espacial tridimensional. El modelo YOLOv8n, reentrenado con un dataset personalizado generado mediante Label Studio, exhibe aceptables niveles de robustez gracias al uso de técnicas de aumento de datos (*data augmentation*), las cuales mejoraron el recall y la precisión espacial frente a oclusiones o cambios de iluminación.

Por otra parte, la integración modular en el ecosistema ROS2 Humble mediante una cámara RGB-D Orbbec Astra resolvió con éxito la sincronización temporal entre flujos de datos, garantizando una proyección espacial coherente a coordenadas métricas 3D. Finalmente, el uso de contenedores Docker mejoró la portabilidad y la reproducibilidad del entorno de ejecución, sentando las bases operativas para incorporar este módulo en esquemas de control bimanual de exoesqueletos de miembro superior.

Como líneas futuras de trabajo, se contempla la expansión del conjunto de datos para incluir una mayor diversidad de objetos, así como la ampliación de escenarios para mejorar la robustez del sistema. Además, se plantea el entrenamiento de modelos de mayor capacidad para lograr obtener mejores resultados en la detección de los objetos. Asimismo, se prevé optimizar el rendimiento computacional migrando el modelo a arquitecturas de hardware embebido dedicadas, tales como plataformas NVIDIA Jetson. Por último, se pretende mejorar la estimación espacial utilizando segmentación o filtrado de profundidad.

Agradecimientos

Este trabajo ha sido financiado parcialmente por Agencia Estatal de Investigación, la Unión Europea-Next Generation EU y Generalitat Valenciana a través de los proyectos CIPROM/2022/12 y PLEC2022-009424.

Referencias

Catalán, J. M., Trigili, E., Nann, M., Blanco-Ivorra, A., Lauretti, C., Cordella, F., Ivorra, E., Armstrong, E., Crea, S., Alcañiz, M., et al., 2023. Hybrid brain/neural interface and autonomous vision-guided whole-arm exoskeleton control to perform activities of daily living (adls). Journal of NeuroEngineering and Rehabilitation 20 (1), 61.

Crea, S., Nann, M., Trigili, E., Cordella, F., Baldoni, A., Badesa, F. J., Catalán, J. M., Zollo, L., Vitiello, N., Aracil, N. G., et al., 2018. Feasibility and safety of shared eeg/eog and vision-guided autonomous whole-arm exoskeleton control to perform activities of daily living. Scientific reports 8 (1), 10823.

Fares, N., Sherratt, R. S., Elhajj, I. H., 2021. Directing and orienting ict healthcare solutions to address the needs of the aging population. In: Healthcare. Vol. 9. MDPI, p. 147.

Frisoli, A., Barsotti, M., Sotgiu, E., Lamola, G., Procopio, C., Chisari, C., 2022. A randomized clinical control study on the efficacy of three-dimensional upper limb robotic exoskeleton training in chronic stroke. Journal of neuroengineering and rehabilitation 19 (1), 14.

Irles, C. F., Ruiz, F. J. M., Ivorra, A. B., Cerdán, E. B., Orts, J. M. C., Aracil, N. G., 2024. Diseño mecánico de un exoesqueleto bimanual para la asistencia en actividades de la vida diaria. Jornadas de Automática (45).

Izzo, C., Carrizzo, A., Alfano, A., Virtuoso, N., Capunzo, M., Calabrese, M., De Simone, E., Sciarretta, S., Frati, G., Oliveti, M., et al., 2018. The impact of aging on cardio and cerebrovascular diseases. International journal of molecular sciences 19 (2), 481.

Jocher, G., Chaurasia, A., Qiu, J., 2023. Ultralytics yolov8.
URL: `https://github.com/ultralytics/ultralytics`

Pantoni, L., Poggesi, A., Inzitari, D., 2009. Cognitive decline and dementia related to cerebrovascular diseases: some evidence and concepts. Cerebrovascular Diseases 27 (Suppl. 1), 191–196.

Park, D., Hoshi, Y., Mahajan, H. P., Kim, H. K., Erickson, Z., Rogers, W. A., Kemp, C. C., 2020. Active robot-assisted feeding with a general-purpose mobile manipulator: Design, evaluation, and lessons learned. Robotics and Autonomous Systems 124, 103344.

Salomon, J. A., Vos, T., Hogan, D. R., Gagnon, M., Naghavi, M., Mokdad, A., Begum, N., Shah, R., Karyana, M., Kosen, S., et al., 2012. Common values in assessing health outcomes from disease and injury: disability weights measurement study for the global burden of disease study 2010. The Lancet 380 (9859), 2129–2143.

Tkachenko, M., Malyuk, M., Holmanyuk, A., Liubimov, N., 2020. Label Studio: Data labeling software. Open source software available from https://github.com/HumanSignal/label-studio.
URL: `https://github.com/HumanSignal/label-studio`