Sistema de optimización de parámetros para la detección de objetos en tiempo real

Daniela López De Luise [1,2,4], Jin Sung Park[4], Silvia Rosana Hoferek[1,3], Lautaro Nicolás Ávila[1], Micaela Antonella Benítez[1] , Félix Raul Bordón Sbardella[1], Gastón Emmanuel Machado[1], Aramis Oscar Mencia[1], Anahí Ailén Ríos[1], Emiliano Luis Ríos[1], Nahuel Edgardo Riveros[1]

hofereksilviarosana_for@ucp.edu.ar

[1] Facultad de Ingeniería, Tecnología y Arquitectura, Instituto de Investigaciones Científicas (IDIC)- Universidad de la Cuenca del Plata

[2] CAETI – Universidad Abierta Interamericana – Facultad de Tecnología Informática

[3] Universidad Siglo 21, Decanato de Ciencias Aplicadas, Argentina

[4]CI2S Labs, Ciudad de Buenos Aires, Argentina

Resumen

El objetivo de este artículo es describir los hallazgos sobre el prototipo para asistir a personas no videntes. La metodología principal aplicada para construir un sistema inteligente compuesto de modelos de machine learning que detecte y reconozca múltiples objetos consiste en la aplicación de los siguientes pasos: “Generación sistemática y protocolizada de datos a partir de material audiovisual” o en otras palabras la creación de videos y obtención de imágenes; “Pre Procesamiento” del material obtenido; y “Entrenamiento del modelo neuronal con las imágenes”.

El alcance de las actividades realizadas incluye evaluación de eficiencia, compilación de datos sobre video, segmentación de imágenes, minería y procesamiento de datos, y etiquetación. Este trabajo también evalúa y describe ciertas técnicas y procedimientos utilizados para crear modelos con un grado alto de eficiencia en detección de patrones. Los algoritmos planteados tienen carácter liviano y veloz, este requerimiento es necesario para ser utilizados en celulares estándares usados por los videntes para proveer información significativa y además de poder realizar pequeños análisis de los resultados. La idea de este estudio es sobre las metodologías, estrategias empleadas para desarrollar un modelo eficiente para reconocimiento de patrones.

Palabras clave Asistencia al invidente, Procesamiento de video, Detección de objetos, Minería de datos, modelo altamente eficiente.

Abstract The objective of this article is to describe the findings on the prototype to assist visually impaired individuals. The primary methodology applied to construct an intelligent system composed of machine learning models that detect and recognize multiple objects involves the following steps: "Systematic and protocolized data generation from audiovisual material," or in other words, the creation of videos and obtaining images; "Pre-Processing" of the obtained material; and "Training the neural model with the images."The scope of the activities includes efficiency evaluation, video data compilation, image segmentation, data mining and processing, and labeling. This work also evaluates and describes certain techniques and procedures used to create models with a high degree of efficiency in pattern detection. The proposed algorithms are lightweight and fast, a necessary requirement for use on standard mobile phones used by the sighted to provide meaningful information and perform small analyses of the results. The idea of this study is about the methodologies and strategies employed to develop an efficient model for pattern recognition.

Keywords Assistance for the visually impaired, Video processing, Object detection, Data mining, Highly efficient model.

Introducción

El trabajo presentado es parte del proyecto HOLOTECH [03] tanto la problemática como el objetivo final a solucionar es desarrollar una herramienta auxiliar para incrementar la autonomía de las personas con discapacidad visual.

Según las investigaciones realizadas personas que experimentan discapacidad visual [01][02] a menudo se encuentran con obstáculos al intentar desplazarse en entornos pocos familiares, particularmente en espacios públicos y al aire libre. Para abordar esta problemática, y poder desplazarse en distintos lugares actualmente precisan el uso de herramienta como el bastón, el apoyo de otras personas o animales entrenados. Actualmente el

bastón es uno de los recursos limitados, pero más populares. Sin embargo estas soluciones, salvo el caso de la ayuda de otras personas, no aseguran que el invidente esté a salvo ante algunos obstáculos como pozos o puertas. Por lo tanto, y para evitar estos inconvenientes, se plantea otra solución para permitir tener una alternativa más precisa para advertir sobre estos impedimentos en el camino. Uno de los requerimientos necesarios es que el sistema pueda reconocer los obstáculos para tomar decisiones al respecto, siendo así la incorporación de *machine learning* para la detección. No obstante, los obstáculos a detectar no son frecuentes y es necesario generar nuevos modelos.

Este trabajo está basado en el estudio comparativo de la optimización de *performance* en la detección instantánea con el apoyo de redes entrenadas adicionales, para crear nuevos modelos que funcionen igual o mejor de las que hay disponibles actualmente y permitan además detectar otros objetos que no están presente en los pre entrenados.

Etapa I. Investigación y Proyectos Previos

La propuesta introducida en este proyecto HOLOTECH [03] es lograr notificar a la persona no vidente mediante un mecanismo de alarmas generado por un módulo inteligente que toma de decisiones basado información evaluada automáticamente para detectar la presencia de potenciales riesgos específicos del entorno. La característica de esta herramienta es aprovechar tecnologías inteligentes y utilizar componentes accesibles al usuario. El dispositivo base es un teléfono celular gama media, eventualmente complementado con accesorios adicionales para mejorar la detección y predicción de eventos de interés previamente configurables por el [04][05]. La arquitectura del prototipo detecta obstáculos en el entorno cercano de la persona con discapacidad visual y genera patrones de sonido y vibración ante posibles riesgos.

En función de esto se estableció una colaboración estrecha con el Circulo de No Videntes (CINOVI) [06], que permite establecer cuáles son los obstáculos de interés desde la perspectiva de los desafíos que enfrentan estas personas. La interacción con el usuario real ha permitido desmitificar hechos como la necesidad de detectar personas en el entorno, e identificar los objetos y eventos que sí constituyen los problemas más frecuentes del entorno, aquellos de detección más compleja mediante el bastón que persiste como el apoyo fundamental. Considerando los resultados anteriores del trabajo realizado en el proyecto con miembros de la Biblioteca Argentina para Ciegos (BAC) [07], del centro CAETI de UAI [08] y del equipo actual conjunto al CINOVI, se han definido objetivos que se enfocan en la detección de objetos utilizando sistemas inteligentes conexionistas y un sistema experto para interpretar los resultados obtenidos de las redes neuronales. De esta manera se obtienen comunicaciones ubicuas inteligentes que asisten al invidente con información clara y precisa.

Los antecedentes de la propuesta actual se basan en múltiples avances en el campo del procesamiento de vídeo. Entre otros, López De Luise et. al. [9] han utilizado el cortado, la segmentación y el aprendizaje automático (ML) para inferir escenas 3D a partir de imágenes 2D (Figura 1), y la conclusión del estado de ánimo, donde un artista de IA cambia una obra de arte humana para incorporar el estado interno inferido del observador eligiendo entre «pinceles» inteligentes, y luego actualiza la imagen en tiempo real (Figura 2). Otros autores realizaron coloreado inteligente a partir de imágenes en escala de grises [10], reconstrucción endoscópica en 3D [11], restauración de imágenes [12], inferencia profunda de imágenes [13], compresión de vídeo [14], y otras amplias aplicaciones [15] de la inteligencia computacional en el procesamiento de vídeo.

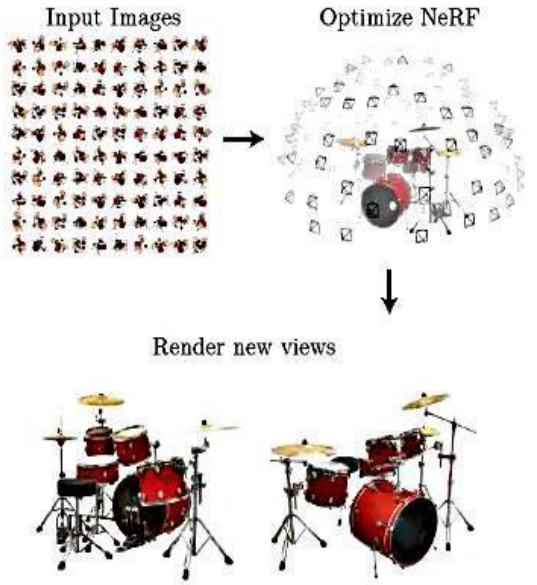


Figura 1 Imagen 2D (izquierda) e imagen 3D inferida resultantes (derecha

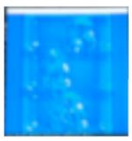
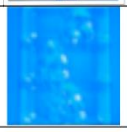
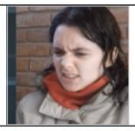
Mood	Table Column Head		
	Previous status	Final status	Expression
Neutral	NO		
Sad			
Horror			
Surprise			

Figura 2 Tabla de producciones de Lumiere (un artista IA) utilizando la inferencia de estados de ánimo.

Etapa II. Elección de Metodología y Materiales

Para el desarrollo del prototipo se emplea software y hardware que cumple con las restricciones y requerimientos de los análisis obtenidos de las investigaciones realizadas por el intercambio con la BAC. El conjunto de módulos de software que constituyen el proyecto detecta obstáculos y etapas que inician en la preparación esencial de los datos y culminan en el uso de modelos de comportamiento entrenados con sistemas conexionistas. Las etapas consisten en:

A. Generación sistemática y protocolizada de datos a partir de material audiovisual. Esta primera etapa consiste en generar una base de datos derivados de videos capturados en base a un protocolo predeterminado para acotar el sesgo de información a modelar con las redes neuronales. La duración promedio de las grabaciones es de 1 minuto. Para esta tarea se utilizan dispositivos móviles con capacidad de grabación de vídeo y memoria interna mínima de 64 GB. La grabación se realiza con el dispositivo a la altura del pecho, con un desplazamiento lento, y captando la mayor cantidad de posibles obstáculos que se encuentren durante el desplazamiento del sujeto. Estas capturas tienen una resolución estándar entre 320x240 dpi a 720x480 de dpi. El formato es XML [16].
B. Preprocesamiento: Una vez obtenidos los videos se tratan con el fin de mejorar la detección sistemática y consistente. Los procesos que se realizan durante esta etapa son: pasaje de imágenes a blanco y negro, manejo de blur y eliminación del contorno, filtrado de elementos [17][18] y por último reconstrucción [19].
C. Entrenamiento del modelo neuronal con las imágenes: Como última etapa del sistema, se toman los datos ya procesados y clasificados, y se entrena a la red neuronal. Por medio de un proceso de optimización se ajustan los metadatos hasta obtener una tasa de error y métricas de validación aceptables.

En cuanto a las herramientas de software, se emplean el lenguaje *Python* por su capacidad de ser multiparadigma y adaptabilidad, *OpenCV* [20] para el entrenamiento conexionista, la librería Pandas de Python, *Kivy* [21] como Framework y ambiente de trabajo [21], *Flet* [22] y *Coffe* [23] para el front-end de la aplicación, *YOLO* [22] junto con el previamente mencionado *OpenCV* para el procesamiento de imágenes, Buildozer [24] como empaquetador de la *APK* [25], y *Labelme* [26] que se utiliza para crear nuevas etiquetas en el Log de rastreo.

Cabe mencionar que *Androidstudios* [27] es una la herramienta empleada inicialmente para probar ciertos apartados, pero por cuestiones de compatibilidad, se descartó y fue cambiada por *Buildozer*.

Debido al contexto dado y la solución propuesta, el concepto de la arquitectura del prototipo implica el uso de hardware compacto y liviano, como un teléfono inteligente junto con un Arduino Nano. Esta combinación permite que el modelo incorpora hardware que normalmente no se encuentra en un teléfono inteligente, como un sensor de ultrasonido. El propósito de esta arquitectura es servir como un prototipo inicial, facilitando la exploración de la solución propuesta. Sin embargo, un aspecto crucial del prototipo es contar con un hardware capaz de capturar información, reconocer y ejecutar diversas rutinas y acciones. Por lo tanto, el requisito mínimo para la arquitectura es la capacidad de capturar vídeo y procesarlo utilizando un modelo entrenado para su reconocimiento.

La idea de reconocer objetos es predecir y filtrar el comportamiento. Una silla o un escritorio son objetos inamovibles por sí mismos, por lo que la única forma en que una persona discapacitada pueda chocar es mediante su propio movimiento, lo mismo se puede aplicar a otros objetos como los árboles. Por esa razón, es imperativo comprender que incluso si el requisito debe

detectarse en tiempo real, el proceso para reconocerlo puede llevar un tiempo razonable. Por ello la arquitectura propuesta se centra en el uso de un *smartphone*. Sin embargo para casos en tiempo real que necesitan reacciones rápidas de detección estamos incorporando sensores ultrasónicos que nos permitirán saber cuándo aparece un objeto repentino en la trayectoria y, en este caso, no importa la necesidad de saber qué objeto hay en el camino, sólo en la dirección de donde surgió, la arquitectura propuesta también pretende facilitar la comunicación con personas discapacitadas a través de diversos métodos, como el uso del sonido o la vibración en casos extremos. El artículo se centra en cómo se desarrolló el sistema que permitirá detectar y reconocer el objeto a cierta distancia: con qué precisión son los modelos y con la arquitectura utilizada como base.

Considerando las restricciones de hardware, hay una limitación estratégica en el modelo a diseñar para identificar cualquier objeto de interés. En investigaciones anteriores, las estadísticas muestran esta limitación impuesta en las imágenes utilizadas para entrenar el modelo con una calidad y resolución de alrededor de 320x240 dpi a 720x480 dpi.

Para probar y entrenar el modelo, se realiza un preprocesamiento para mejorar el contorno de cualquier objeto objetivo, utilizando una herramienta llamada *Labelme*. Esta herramienta realiza anotaciones gráficas automáticas en imágenes. Esto permite generar datos para detectar el objeto deseado utilizando polígonos y etiquetas para clasificar automáticamente, generando metadatos que se almacenan en un archivo *JSON*. El archivo *JSON* para ser utilizado es necesario realizar un proceso de adaptación utilizando otra herramienta *Labelme2yolo*[28] para que este pueda ser utilizado por *YOLO*, el resultado obtenido del proceso de adaptación es luego utilizado para el entrenamiento de la red. La herramienta de Red Neuronal utilizada aquí para la detección de objetos es *YOLO*, debido a su enfoque único que garantiza rapidez. Aunque existen otras herramientas populares como Faster *R-CNN* [29] y *SSD* [30], pero *YOLO* es más fácil de usar y más popular, con un mejor soporte de la comunidad.

Este proyecto emplea *PyTorch* [31] para implementar el modelo de detección de objetos basado en *YOLO* [32]. *PyTorch* ofrece herramientas para definir arquitecturas de redes neuronales, optimizar modelos, calcular gradientes automáticamente, permitir la capacitación distribuida y más. Además, *PyTorch* es compatible con la ejecución en *GPU* o *CPU*, lo que proporciona una opción valiosa para los usuarios sin acceso a una tarjeta gráfica habilitada para *CUDA*. Además, se utilizó *Ultralytics* [33] para entrenar la red neuronal.

Para establecer un criterio de rendimiento de los modelos creados las mismas se le compara con los resultados de los modelos pre entrenados existentes.

La opción dependerá del rendimiento de los modelos recién creados. En algunos casos, los modelos previamente entrenados no se cargan para reducir la sobrecarga de información, lo que reduce la cantidad de patrones y, por lo tanto, minimiza la tasa de activación de las tareas de filtrado de datos. Además, se procura entrenar modelos donde no exista un modelo pre entrenado, siendo así aplicar el procedimiento de la creación en estos modelos.

Los objetivos seleccionados para trabajar en el artículo actual y en el entrenamiento son "sillas", "mesas", "puertas" y "personas". Las condiciones de prueba para todos los objetos se encuentran principalmente en interiores y exteriores. Como procedimiento intermedio, hay una validación adicional de la necesidad de usar un modelo basado en el grupo de clasificación. Esto se debe a que en investigaciones anteriores surgieron casos especiales donde las formas de diferentes obstáculos inducen errores en los patrones. Por ejemplo, algunas sillas y mesas se ven muy similares dependiendo de la perspectiva.

Finalmente, dos objetivos especiales son las puertas y las personas. La primera es porque no presentan ningún modelo pre-entrenado en la herramienta. La última es porque serán detectados y filtrados. Las personas ciegas no necesitan ser avisados a través de una alarma cada vez que una persona se acerca, ya que las personas no representan ningún riesgo de colisión.

Etapa III. Prototipo y Pruebas de laboratorio

El ambiente para desarrollo del sistema dispone de carpetas que organizan por separado los videos que constituyen recortes, y las escenas completas. A su vez se discriminan entornos cerrados de exteriores. Adicionalmente, en el mismo grupo se identifican ambientes específicos: acera, vereda y eventuales vehículos en el camino.

Para poder utilizar los videos, deben ser editados a fin de eliminar las pistas de audio. De esta manera se evitan los datos que no son relevantes para la detección y se disminuye el peso del archivo. También se dividen algunos tests en segmentos menores, por caso, discriminando interiores de exteriores o si se cruzó la calle.

Gracias al CINOVI y los aportes en la información que compartió la entidad, se ha validado la lista de obstáculos y eventos de interés para el sistema en desarrollo. Debe destacarse que muchos de los elementos de interés son complicados de detectar ya que se encuentran por encima de la cintura.

Algunos de dichos elementos son:

- Autos mal estacionados en las veredas o en espacios no permitidos.
- Motos/bicicletas estacionadas en lugares no permitidos o circulando irregularmente.
- Portones abiertos o que abren hacia la calle.
- Ramas de plantas o algún objeto que sobresalga en la ruta de desplazamiento, a partir de la altura de las rodillas hacia arriba.
- Rampas municipales.
- Dispositivos instalados, como acondicionadores de aire a alturas no recomendadas, principalmente cuando sobresalen de la pared.
- Pozos o roturas en el suelo, con o sin vallado.
- Obstáculos móviles eventualmente peligrosos como cochecitos, vehículos diversos, ciclistas, etc.

Teniendo en cuenta la información previa en esta sección se explican las pruebas realizadas para confeccionar el modelo neuronal de reconocimiento de patrones con *YOLO* y se describen algunos de los pasos esenciales para ajustar el rendimiento en *YOLO* con el fin de construir un modelo mejor. La actividad se realiza en cinco versiones de la evaluación del modelo.

A. Primera versión: conjunto reducido

El entrenamiento tiene como objetivo construir un modelo que supere la precisión del modelo por defecto en *YOLO*. El procedimiento incluye un paso de validación, la evaluación del nivel de precisión y una comparación final entre los modelos. Merece especial atención el hecho de que la fuente de datos de entrenamiento de videos utilizada para la red neuronal pre entrenada en la plataforma no está disponible para los usuarios finales. Por lo tanto, hay que generar y etiquetar de antemano un nuevo conjunto de entrenamiento con obstáculos similares. Las clases de objetos se denominan aquí *ModelGroup*, y cada una de ellas tiene un modelo independiente entrenado específicamente para el subconjunto de objetos concretos que contiene. Considerando la lista reducida que abarca la prueba actual, el entrenamiento abarcó cuatro modelos, cada uno de los cuales utilizó un conjunto de 150 imágenes de cada tipo de objeto. En cada imagen los objetos de interés han sido convenientemente etiquetados utilizando *Labelme*.

Para establecer la información necesaria a partir de una imagen, se utiliza la herramienta *Labelme*, que agiliza el proceso. El etiquetado se realiza manualmente para garantizar que los obstáculos se identifican con precisión dentro de la imagen. La Figura 3 muestra una captura de pantalla del proceso.

Figura 3 Proceso de etiquetado utilizando Labelme

Como se puede ver en la imagen, *Labelme* permite al usuario definir metadatos sin la necesidad de utilizar conceptos complejos de procesamiento de imágenes para filtrar información no deseada y obtener los datos requeridos. Una vez que se etiqueta una imagen, *Labelme* proporciona las características del contexto de la imagen y las características del objeto etiquetado en un Formato *JSON* (Figura 4).

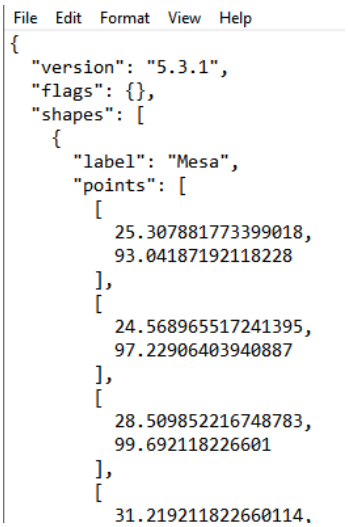


Figura 4 Ejemplo de características generadas automáticamente en formato JSON.

El archivo *JSON* requiere un procesamiento adicional para transformarlo en el formato apto para ser utilizado por *YOLO*. Este paso se realiza con una herramienta

llamada *Labelme2yolo* que genera la estructura para el entrenamiento del modelo. Además, la herramienta facilita la segmentación de las imágenes en subconjuntos de prueba, entrenamiento y validación y, si es necesario, puede aplicar técnicas de preprocesamiento para convertir las imágenes como en la Fig. 5.

Figura 5 Ejemplo de una etapa de pre-procesamiento

1) Problema de dimensión

Para mejorar el rendimiento del procesamiento de imágenes, se prueba el cambio de tamaño de las imágenes a tamaños estándar: pequeña (100 ppp o 320 x 240), mediana (200 ppp o 500 x 300) y grande (300 ppp o 720 x 480). A pesar de una posible distorsión menor de las formas de los objetos de aproximadamente un 1 % o menos durante el cambio de escala, este procedimiento centra y refina las imágenes eliminando el ruido durante el procesamiento y el desarrollo del modelo.

Para realizar este proceso de cambio de tamaño, se plantea un sistema compacto capaz de obtener imágenes con sus respectivos nombres y formatos especificados. La biblioteca de imágenes de Python (PIL) se utiliza como herramienta para cambiar el tamaño. Para cada categoría de tamaño, se establece una condición previa: si el tamaño de la imagen excede un cierto umbral, se cambiaría su tamaño para ajustarse al tamaño designado para esa categoría y el archivo actualizado se guardaría con el mismo nombre y extensión.

2) Modelo

El modelo de entrenamiento se compone de un conjunto de imágenes de sillas, escritorios y puertas. La premisa es aplicar el patrón de coincidencias para objetos similares, como se menciona previamente, para determinar los datos de confusión de modelo, evaluar el nivel de confianza de la identificación del objeto, y limitar la tasa de fallas.

Las sesiones de entrenamiento del modelo consisten en 100 épocas utilizando el tipo de modelo *yolov8m-seg* de *Yolo*. El entrenamiento se realiza en dos batches. Debido al uso de una estación de trabajo Mac Pro, el proceso de entrenamiento toma varias horas en completarse. Es importante destacar que esto no es un problema, ya que no se espera que el modelo sea re-entrenado durante su aplicación en el campo. Sin embargo, es posible reducir el tiempo de entrenamiento utilizando una tarjeta gráfica NVIDIA en un entorno Linux.

La configuración utilizada para entrenar los modelos es el paso inicial para comprender la respuesta del entrenamiento y ajustarla si es necesario para mejorar la precisión del modelo.

Para enfatizar el objeto en los modelos individuales, se agrega una etiqueta adicional “fondo” para indicar cuando se seleccionaba un objeto diferente al esperado. Esto se hace con el fin de crear falsos positivos y tener una comprensión clara del ruido de fondo (véase un ejemplo del problema en la Figura 6).

Figura 6 Ejemplo de etiquetado con fondo incluido

3) Resultados

Es interesante observar que el *ModelGroup*, consigue encontrar la categoría del objeto pero no identifica correctamente el objeto específico en los casos donde se entrena con objetos que comparten características similares, como podrían ser un escritorio o una silla. El problema se muestra en la Figura 7.

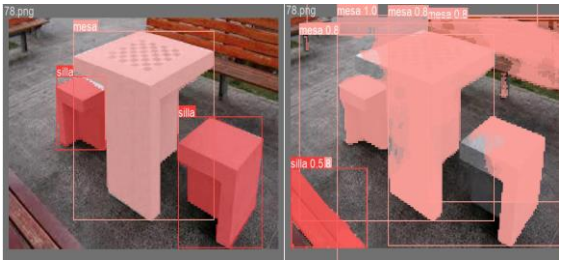


Figura 7 Comparación de los objetos marcados y el resultado previsto

La precisión de algunos objetos identificados es lo suficientemente alta como para que el modelo esté seguro de que el objeto es un escritorio, con una puntuación de 0,8. Sin embargo, en algunos casos, el modelo todavía

encuentra objetos incorrectamente y determina una clasificación incorrecta con una puntuación de confianza alta. El problema surge justo en los casos en que la forma del objeto es similar entre dos o más patrones (como en el caso de la silla y la mesa en la Fig. 7, donde el modelo termina identificando todo como un escritorio). Una solución para este caso particular es agregar una restricción de límites para enfatizar que una silla debe estar dentro de una determinada dimensión, y lo mismo ocurre con el escritorio. Sin embargo, esto soluciona casos puntuales como la confusión entre sillas y escritorios. A medida que se agreguen objetos más diversos al entrenamiento del modelo, no será ideal y las restricciones se volverán más complejas.
No obstante, los modelos entrenados individualmente incluyendo el fondo no alcanzan la precisión esperada, como se muestra en la Figura 8.

Figura 8 Resultado previsto (derecha) usando modelos dedicados

Esta vez, la precisión del resultado previsto no es perfecta. Como puede verse en la imagen, el modelo encuentra múltiples lecturas respecto a los objetos identificados, con un rango de confianza entre 0.3 y 0.7, identificando también un objeto como fondo. Esto indicaría que el principal problema del modelo podría ser las diversas interpretaciones de los objetos reconocidos, con puntuaciones de confianza que oscilan entre 0.3 y 0.7. Aunque la identificación del fondo como objeto puede eliminarse, la tendencia constante de las lecturas que caen dentro de este rango de confianza para los objetos previstos es preocupante. Este problema hace que los modelos no sean aptos para su uso, ya que podría resultar en situaciones potencialmente peligrosas para los individuos.
Para comprender mejor la usabilidad de los modelos, se realiza un estudio basándose en los resultados del entrenamiento. Estos datos pueden utilizarse para comprender estadísticamente los modelos y ajustarlos para hacerlos más precisos. Los resultados se presentan en las Figuras 9 y 10.

La adición del etiquetado de fondo para el modelo entrenado individual tiene como fin reducir el ruido de fondo encontrado en el análisis de la matriz de confusión entrenada en el modelo *ModelGroup* Figura 11.

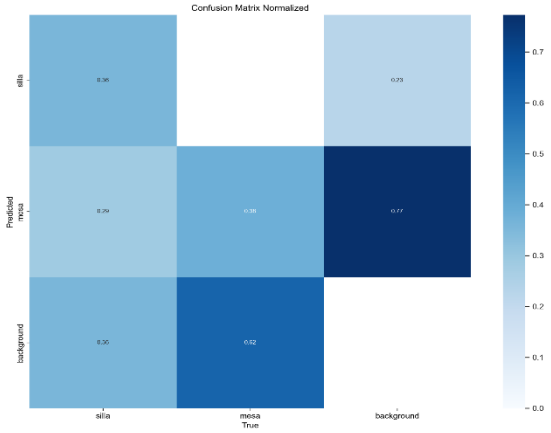


Figura 11 Matriz de Confusión del ModelGroup

Aunque los datos utilizados para entrenar el modelo no contienen una etiqueta de *background*, el modelo predijo el objeto de escritorio como uno en aproximadamente el 77% de los casos. Para reducir la tasa de predicciones erróneas, se proporciona el *background* durante el entrenamiento de los demás modelos.
La inclusión de la etiqueta de fondo no resulta en la mejora esperada en los modelos individuales. Esto deriva en que el fondo se identificara como un objeto separado, como se muestra en la Fig. 8. Esto indica que pueden ser necesarias modificaciones adicionales o un ajuste fino para resolver este problema. Aunque la tasa de predicciones incorrectas logra ser menor que en otros modelos, aún restan demasiados problemas con el resultado.

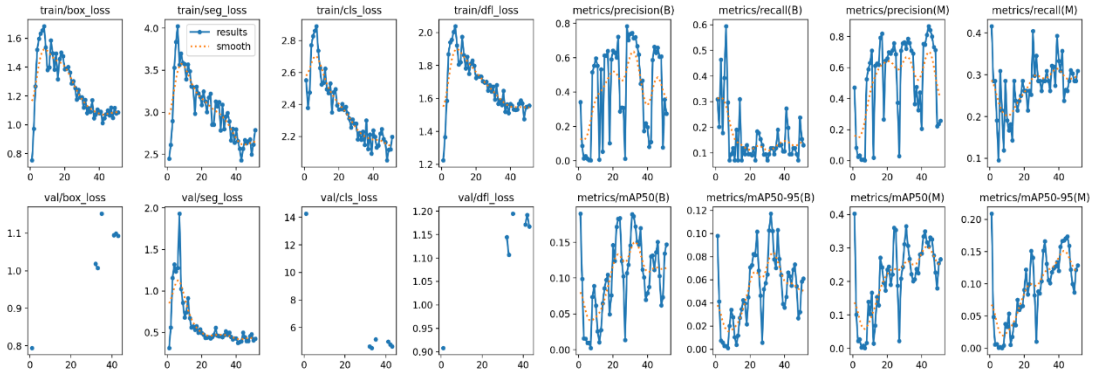


Figura 9 Resultados del modelo de entrenamiento

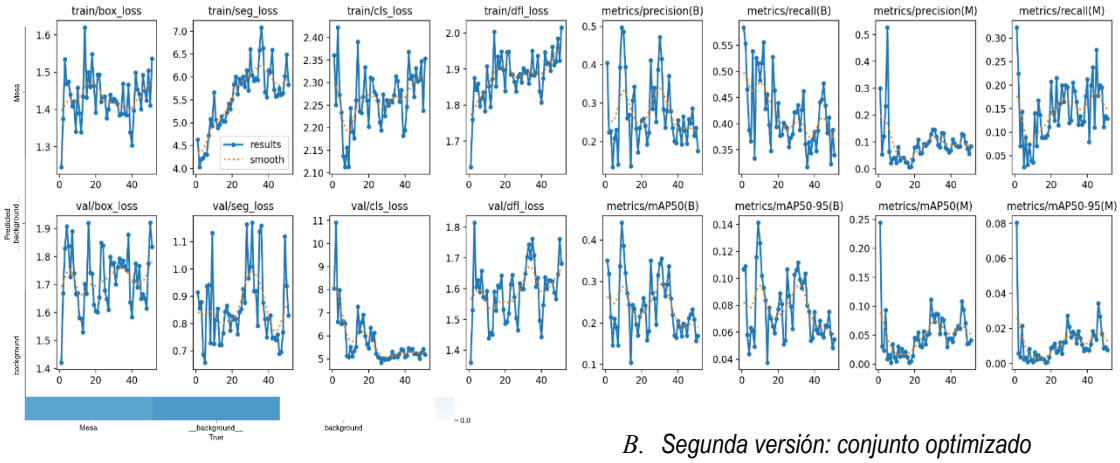


Figura 10 Resultados del entrenamiento del modelo de mesas

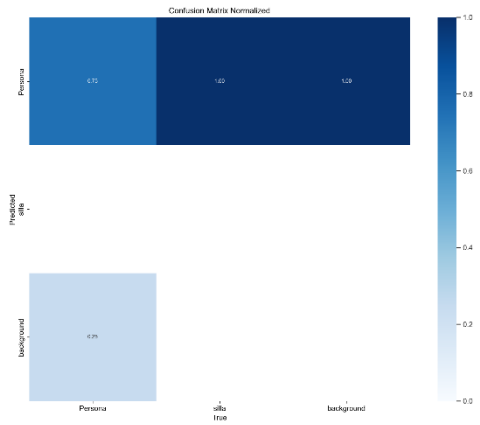


Figura 12 Matriz de Confusión del Modelo de Mesas

El resultado obtenido para el objeto de escritorio es similar en ambos modelos, con una predicción del 38% y el 36%, respectivamente.

B. Segunda versión: conjunto optimizado

El proceso de entrenamiento se lleva a cabo de manera similar a la versión anterior, pero esta vez el conjunto de datos consta de 38 imágenes seleccionadas que se centran principalmente en dos categorías: personas y sillas. Dada la naturaleza de los objetos de interés (personas y sillas), se introduce una nueva fase para identificar estos objetos. En este proceso, se recopilan imágenes adicionales para aumentar el conjunto de datos con fines de prueba. El objetivo principal es evaluar la eficacia del código recientemente desarrollado para detectar estos objetos específicos.

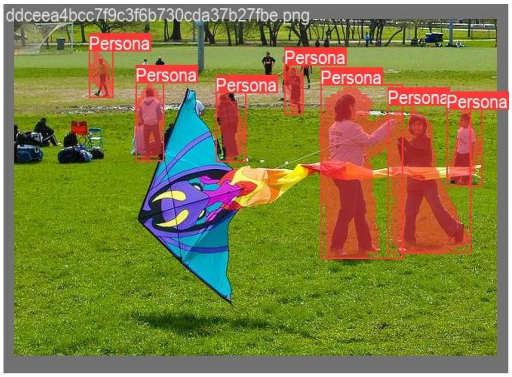


Figura 13 Etiquetado de personas

1) Modelo

El nuevo modelo lleva por nombre *ModelPeopleChairGroup* y está enfocado a detectar personas y sillas como ya se ha mencionado. Este modelo tiene un entrenamiento de 30 épocas en 2 lotes usando *YOLO*. La capacitación se lleva a cabo en una estación de trabajo con sistema operativo *Windows*, utilizando procesamiento de CPU, lo que lleva a tiempos de entrenamiento más prolongados, con imágenes que incorporan etiquetas de personas (Figura 14).

Figura 15 Etiquetado de personas

2) Resultados

Al evaluar los resultados obtenidos de *ModelPeopleChairGroup*, se observa una mejora en la detección de personas en comparación con iteraciones de entrenamiento anteriores. Sin embargo, el modelo todavía tiene dificultades para detectar múltiples individuos con precisión. Específicamente, en un caso representado en la imagen Fig. 16, la mitad del grupo de personas no fue etiquetada. La precisión del etiquetado de objetos en estos resultados alcanza un nivel de confianza de 0,6 (ver Figuras 17 y 18).

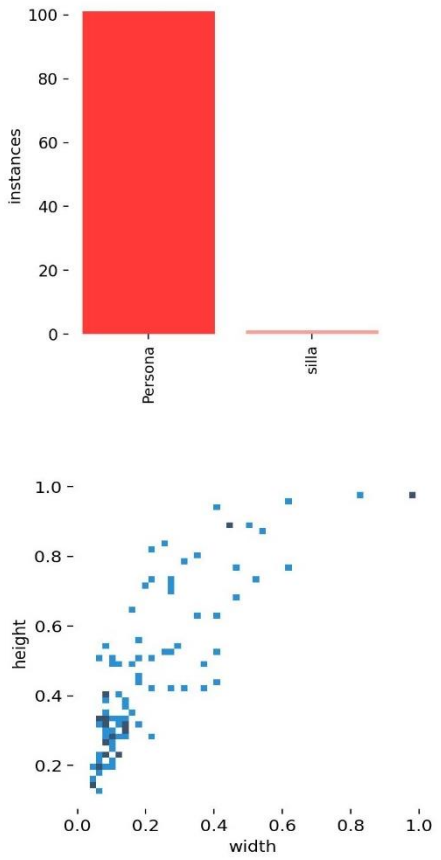


Figura 17 Resultado de la predicción basada en cada etiqueta

Figura 16 Comparación entre personas etiquetadas y predichas

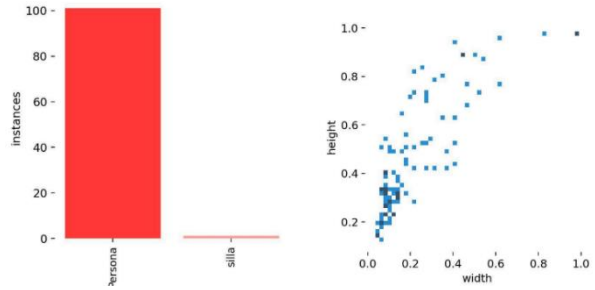


Figura 17 Resultado de la predicción basada en cada etiqueta

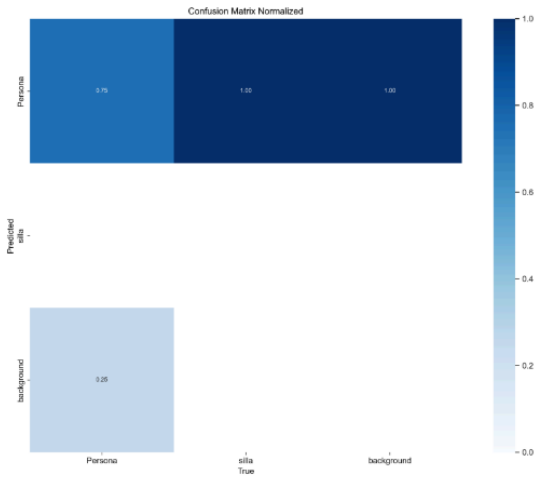


Figura 18 Matriz de confusión

Basado en el resultado, se puede observar una mejora en los resultados obtenidos en comparación con la instancia anterior

C. Tercera versión: conjunto de objetos adicionales

Para la tercera iteración, se realiza un procedimiento similar al anterior, pero con una modificación para el modelo *ModelPeopleChairGroup* en el conjunto de datos al incluir un objeto etiquetado adicional, específicamente tablas. Los objetos de interés, que ahora suman tres para este caso y entrenados simultáneamente, presentaron ciertos desafíos y dificultades para identificar y etiquetar cada elemento. El objetivo aquí es evaluar la eficacia del entrenamiento con un grupo de tres tipos de objetos diferentes.

El archivo *JSON* requiere un procesamiento adicional para transformarlo al formato *YOLO* requerido, ya que luego el conjunto se envía a *YOLO*.

Los resultados del entrenamiento aún no alcanzan la precisión esperada, según se puede observar en el resultado Figuras 19 y 20

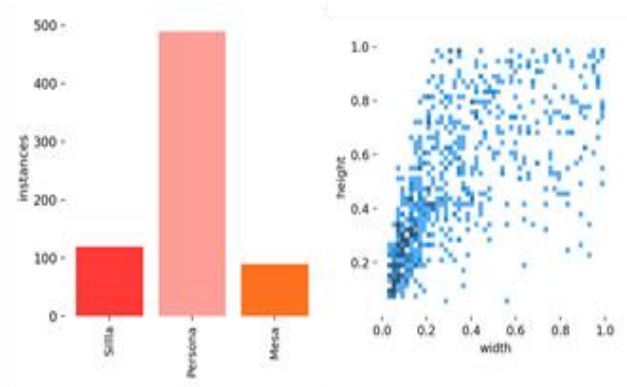


Figura 19 Índice de aciertos de los objetos etiquetados

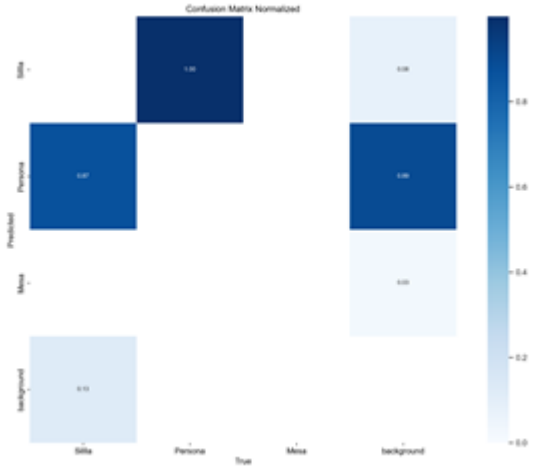


Figura 20 Métrica de Cohen de los objetos etiquetados extendidos.

D. Cuarta versión: conjunto ampliado

En esta prueba, se crea el modelo *ModelPeople* que focaliza las muestras con únicamente imágenes etiquetadas que contienen personas y se completa el conjunto de datos con imágenes adicionales, elevando el total a 271. Al centrarse en un único objeto en el escenario, al modelo le resulta más fácil detectar un objetivo. El objetivo de esta versión es centrarse únicamente en un único objeto y observar si este enfoque produce mejores resultados y tasa de detección. El entrenamiento se mantiene con 100 epochs y 6 lotes utilizando *YOLO*, siguiendo la misma configuración que en los modelos anteriores para facilitar la comparación.

Con una mayor cantidad de imágenes y objetos etiquetados para refinar la red neuronal, se logran los mejores resultados hasta el momento, con respuestas y detecciones que se acercan a la perfección, etiquetando con precisión todo el cuerpo de los individuos. Sin embargo, un problema persiste para objetos o personas situadas demasiado lejos o superpuestas con otros elementos, como se ilustra en las imágenes (Figura 21).

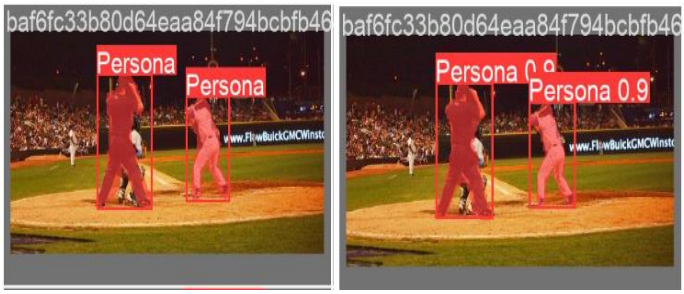


Figura 21 Imagen de resultado y entrenamiento

Como se puede observar en la Figura 22 los resultados obtenidos son prometedores

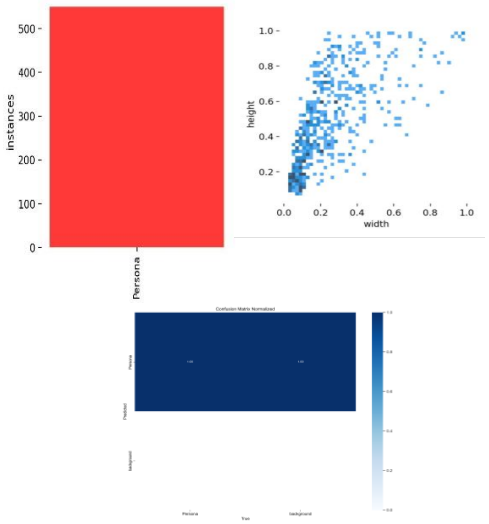
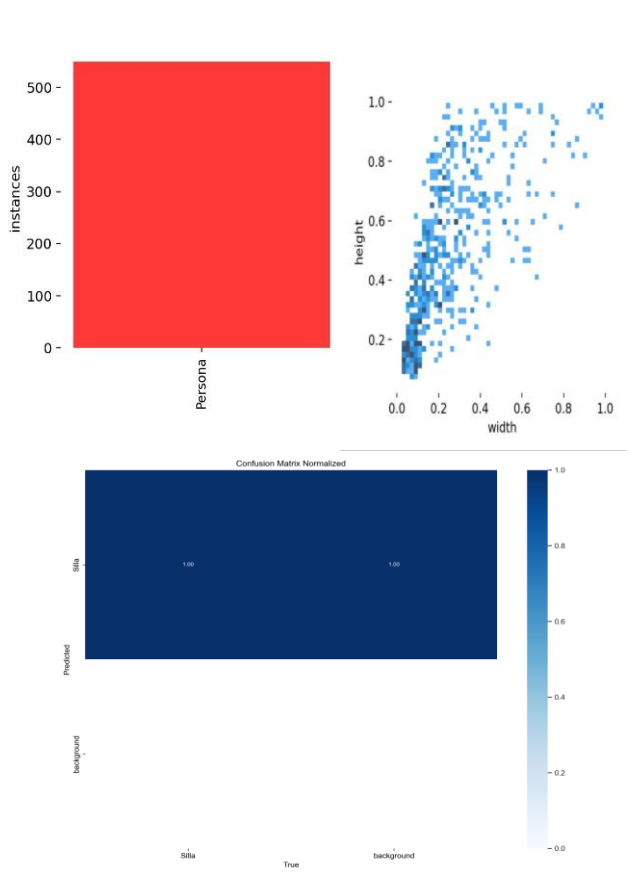


Figura 22 Resultado del entrenamiento cuarta versión

E. Quinta versión: automatización con las rutas

En esta versión se implementan ciertas mejoras en la herramienta permitiendo automatizar y facilitar la obtención de las carpetas de las etiquetas, las imágenes, los datasets, etc. Esta versión del sistema permite tener más seguridad y control en el uso de las rutas y en la optimización del código. Para las pruebas se utiliza el dataset de sillas, con un total de 157 imágenes, con 75 épocas y 6 lotes, creando un modelo *ModelChair*. (Resultados obtenidos Figuras 23, 24, 25 y comparación de las de objetos detectados Figura 26)



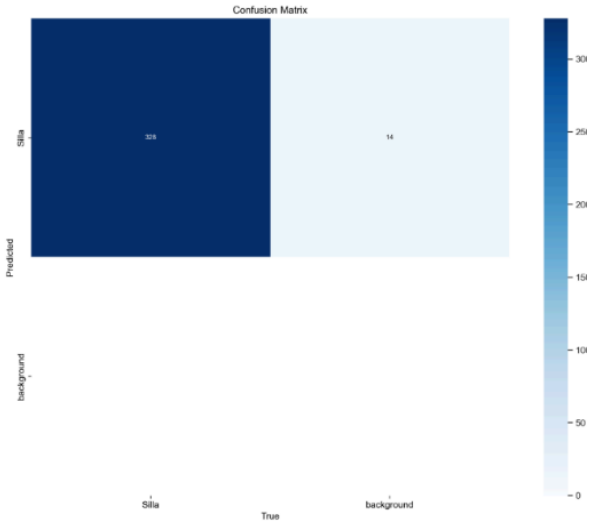


Figura 24 Resultado matriz de confusión

Figura 23 Resultado matriz de confusión normalizado

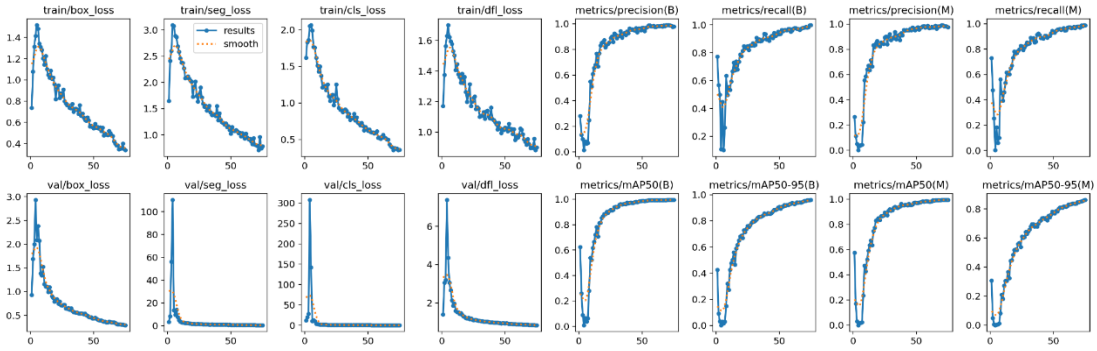


Figura 25 Resultado del entrenamiento quinta versión

Figura 26 Comparación entre la etiqueta y la detección de sillas

Resultados del entrenamiento de Sillas

epoch	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B)	metrics/precision(M)	metrics/recall(M)	metrics/mAP50(M)	metrics/mAP50-95(M)
1	0.27962	0.77134	0.62493	0.42717	0.26415	0.72866	0.57584	0.30705
2	0.1342	0.56707	0.25913	0.09084	0.11256	0.47561	0.17194	0.04826
3	0.09991	0.5	0.09024	0.03279	0.05056	0.25305	0.02984	0.00586
4	0.01361	0.10671	0.00972	0.00277	0.00039	0.00305	0.0001	3.00E-05
5	0.08286	0.44817	0.06487	0.02878	0.03382	0.18293	0.01504	0.0033
6	0.06448	0.10061	0.03685	0.01529	0.0403	0.06098	0.01762	0.00587
7	0.06947	0.2622	0.06221	0.02928	0.04249	0.10061	0.02418	0.01244
8	0.24787	0.63415	0.27937	0.14911	0.22261	0.56098	0.23512	0.08211
9	0.55034	0.49695	0.53996	0.32095	0.55485	0.45601	0.47244	0.20951
...	.....	.....	.....	.....	.....	.....	.....	.....
...	.....	.....	.....	.....	.....	.....	.....	.....
40	0.96391	0.8958	0.97002	0.85282	0.9541	0.88713	0.95218	0.7225
41	0.9558	0.90244	0.9725	0.86061	0.94809	0.8872	0.95548	0.7363
42	0.95655	0.8872	0.97075	0.85437	0.95326	0.88415	0.95588	0.73876
43	0.95227	0.91249	0.97081	0.85397	0.95185	0.90414	0.95254	0.73156
44	0.94733	0.92073	0.97711	0.8628	0.95093	0.90549	0.96397	0.73876
45	0.96712	0.90549	0.97964	0.8728	0.9616	0.89024	0.95496	0.72109
...	.....	.....	.....	.....	.....	.....	.....	.....
...	.....	.....	.....	.....	.....	.....	.....	.....
70	0.98916	0.98171	0.9941	0.9528	0.99304	0.97866	0.99392	0.84858
71	0.9908	0.98505	0.99442	0.95385	0.99067	0.9714	0.99362	0.84957
72	0.98823	0.98171	0.99457	0.95771	0.98183	0.98836	0.99448	0.85119
73	0.98473	0.98334	0.99454	0.95852	0.98473	0.98334	0.99454	0.85624
74	0.97608	0.9951	0.99441	0.95913	0.97929	0.99085	0.99441	0.86028
75	0.97909	0.99915	0.99444	0.96037	0.97588	0.9878	0.99421	0.86199

Figura 27 Resultado del entrenamiento de sillas

Conclusiones

El análisis de los modelos de identificación de distintos objetivos muestra una precisión de predicción similar independientemente del tipo de detección. Además, el trabajo adicional necesario para filtrar el ruido no mejora significativamente el porcentaje de aciertos. Como sólo se trata de objetos que representan un riesgo bajo para las personas ciegas, puede evitarse. En cuanto a la división de un modelo global en modelos de clase, las pruebas muestran que es necesario, pero no debe entrenarse para cada objeto, sino para clases o grupos ontológicos predeterminados, con el fin de mejorar la precisión. Como muestran las pruebas, hay diferentes formas de llevar esto a la práctica. Entre otras, crear modelos sin etiquetado de fondo, mejorar el conjunto de datos de entrada, añadir más falsos positivos y afinar el modelo.

En el estado actual de esta investigación no es posible alcanzar el mismo nivel de precisión que los modelos comerciales pre-entrenados en ciertos casos específicos presentados aquí sobre todo aquellos objetos que no tiene modelos pre-entrenados. Por lo tanto, se requiere de trabajo adicional. En cuanto a la arquitectura implementada en el prototipo, también sigue siendo una puesta a punto para la selección de una adecuada agregación de objetos.

La evaluación de los modelos de identificación de objetos objetivo revela que, tanto si se utiliza un grupo de objetos como si se analizan objetos individuales, los resultados predictivos son en gran medida comparables. No obstante, se observa un rendimiento superior con el análisis de objetos individuales, lo que se traduce en una mejor detección, una reducción del ruido y una notable disminución de los falsos positivos.

Desarrollo a Futuro

Este paper apunta a mejorar los últimos modelos creados en el proyecto, específicamente la versión 5, mejorando así la precisión de la detección y el aumento de objetos a localizar. Los modelos resultantes deben tener un rendimiento similar al de cualquier modelo comercial pre-

entrenado, como lo es el caso del primero utilizado *OpenCV*. El siguiente paso es comparar ambos modelos la versión 5 y el comercial pre-entrenado con el mismo set de pruebas y asegurar que funcione similar en la detección así luego poder iniciar la etapa de alimentar los resultados a un sistema experto para que interprete el contexto y utilice el modelo correcto en una secuencia.

Es necesario seguir investigando para evaluar el rendimiento de estos modelos con otros objetivos. La creación de este sistema permitirá la implementación en el dispositivo móvil y un intérprete automático del entorno utilizando tecnologías como el GPS, un sensor de movimiento o un ultrasonido; para así obtener un conocimiento adicional del entorno interior o exterior del individuo y analizar la existencia de cualquier riesgo en su entorno cercano o lejano.

El prototipo mejorado será adecuado para dar una respuesta en tiempo real y validar el contexto en el que se mueven los objetos y, eventualmente, producirá señales de alarma mediante un lenguaje sonoro compartido con los usuarios para una comunicación fluida entre el prototipo y su usuario.

La incorporación de una señal GPS podría mejorar sustancialmente la conciencia contextual al identificar si un individuo se encuentra en el interior o en el exterior, y podría aprovechar tipos específicos de modelos para este fin.

Bibliografía

1 J. Evangeline,"Guide Systems for the Blind Pedestrian Positioning and Artificial Vision", IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 3. 2014.
2 R. Velázquez, "Wearable Assistive Devices for the Blind". Chapter 17 in A. Lay-Ekuakille & S.C. Mukhopadhyay (Eds.), Wearable and Autonomous Biomedical Devices and Systems for Smart Environment: Issues and Characterization, LNEE 75, Springer, pp 331-349. 2010.
3 Park, J.S., De Luise, D.L., Hemanth, D.J., Pérez, J. (2018). Environment Description for Blind People. In: Balas, V., Jain, L., Balas, M. (eds) Soft Computing Applications. SOFA 2016. Advances in Intelligent Systems and Computing, vol 633. Springer, Cham. https://doi.org/10.1007/978-3-319-62521-8_30
4 H. Weiming, X. Xiao, D. Xie, T. Tan, and S. Maybank. Traffic accident prediction using 3-D model-based vehicle tracking. IEEE transactions on vehicular technology 53, no. 3 pp. 677-694. 2004.
5 P. Soo-Chang, W. Kuo, and W. Huang. "Tracking moving objects in image sequences using 1-D trajectory filter." IEEE Signal Processing Letters 13, no. 1, pp. 13-16. 2006.
6 CINOVI. Asociación sin fines de lucro. Formosa, Argentina. Sitio www.cinovi.org.ar. 2023.
7 Biblioteca Argentina para Ciegos. Asociación sin fines de Lucro. Ciudad Autónoma de Buenos Aires, Argentina. Sitio bac.org.ar. 2023.
8 CAETI. Centro de Altos Estudios en Tecnologías Informáticas perteneciente a la Universidad Abierta Interamericana. Sitio caeti.uai.edu.ar. 2023.
9 Furundarena, F., López De Luise, D., Veiga, M. (2022) Computational Creativity through AI modeling. CASE 2022
10 Komatsu, T., Saito, T. (2006). Color Transformation and Interpolation for Direct Color Imaging with a Color Filter Array. International Conference on Image Processing, pp. 3301-3304, doi: 10.1109/ICIP.2006.312878
11 Imtiaz, M. S., Wahid, K. A. (2014) Image enhancement and space-variant color reproduction method for endoscopic images using adaptive sigmoid function. In 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 3905-3908, doi: 10.1109/EMBC.2014.6944477
12 Wen, Y. W., Ng, M. K., Huang, Y. M. (2008) Efficient Total Variation Minimization Methods for Color Image Restoration. In IEEE Transactions on Image Processing, vol. 17, no. 11, pp. 2081-2088, doi: 10.1109/TIP.2008.2003406
13 Kuo, T., Hsieh, C., Lo, Y. (2013) Depth map estimation from a single video sequence. In IEEE International Symposium on Consumer Electronics, pp. 103-104, doi: 10.1109/ISCE.2013.6570130
14 Yakubenko, M. A., Gashnikov, M. V. (2023) Entropy Modeling in Video Compression Based on Machine Learning. In IX International Conference on Information Technology and Nanotechnology (ITNT), Samara, Russian Federation, pp. 1-4, doi: 10.1109/ITNT57377.2023.10139143
15 De Siva, N. H. T. M., Rupasingha, R. A. H. M. (2023) Classifying YouTube Videos Based on Their Quality: A Comparative Study of Seven Machine Learning Algorithms. In IEEE 17th International Conference on Industrial and Information Systems (ICIIS), Peradeniya, Sri Lanka, pp. 251-256, doi: 10.1109/ICIIS58898.2023.10253580
16 Documentación de Python - 3.11.5 XML. Sitio:docs.python.org. 2023.
17 E. Dubois, and S. Shakeri. Noise reduction in image sequences using motion-compensated temporal filtering. IEEE transactions on communications 32, no. 7 pp. 826-831. 1984.

18 M. Sezan, K. Ibrahim, K. Mehmet, and V. Fogel. Temporally adaptive filtering of noisy image sequences using a robust motion estimation algorithm. In Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on, pp. 2429-2432. IEEE Press. 1991.
19 A. Tekalp, L. Murat, K. Mehmet, and M. Sezan. High-resolution image reconstruction from lower-resolution image sequences and space-varying image restoration. In Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on, vol. 3, pp. 169-172. IEEE Press. 1992.
20 OpenCV. Sitio opencv.org/, /github.com/opencv/opencv, versión 4.8.0.76. 2017.
21 Kivy. Sitio kivy.org, version: 2.2.1, 2023.
22 Flet. Sitio flet.dev/docs, version 0.10.0 , 2023.
23 Y. Jia. Coffee. Sitio caffe.berkeleyvision.org/, version 1.0, 2014.
24 Buildozer: Sitio buildozer.readthedocs.io/en/latest/installation.html, version 1.5.0, 2023.
25 "Instalación del Buildozer y compilar un APK con Python y Kivy" del canal "Everardo MTZ", Jul 3. 2020.
26 Labelme. Sitio github.com/wkentaro/labelme, versión 5.3.0, 2023.
27 Android studios. Sitio developer.android.com/studio, version Giraffe, Mar 1, 2023.
28 labelme2yolo 0.1.3 (2023) Project Description. October 2023 release. https://pypi.org/project/labelme2yolo
29 Ren, S., He, K., Girshick, R., Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Cornell University. Doi: https://arxiv.org/abs/1506.01497
3L iu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A.C. (2016). SSD: Single Shot MultiBox Detector. Doi: https://arxiv.org/abs/1512.02325
31 Pytorch Foundation, (2016), PyTorch, https://pytorch.org/
32 Upulie, H. D. I., Kuganandamurthy, L. (2021). Real-Time Object Detection Using YOLO: A Review. DOI: 10.13140/RG.2.2.24367.66723
33 Ultralytics (2023), https://github.com/ultralytics/ultralytics