SISTEMA DE ORIENTACIÓN EN AMBIENTES CERRADOS PARA EL APOYO A PERSONAS CON CEGUERA TOTAL
(Noviembre de 2020)

J. Arcos

Resumen - Debido a diferente estudio, en los últimos años las personas con problemas de visión han venido ganando muchas batallas legales en el mundo, por su puesto Colombia no es la excepción. Esta es la razón del presente trabajo, buscar una forma de facilitarle la vida a las personas con pérdida total de la visión.

Ha habido muchos desarrollos para las personas invidentes, entre ellos la posibilidad de darle la información en lugares cerrados, con el fin de decirle que objetos se encuentran en él. Existen varios algoritmos los cuales permiten capturar datos de una cámara y otros tantos para identificar objetos de un espacio cerrado. Se busca escoger dos de ellos, uno de cada uno para ser implementados y luego evaluar la precisión con la que detecta los objetos.

Índice de Términos – Convolución-Darknet-Redes-Yolo

I. INTRODUCCION

La organización mundial de la salud hizo un estudio de las personas que tienen dificultades de visión y encontró que la tercera parte de la población del mundo tiene algún tipo de limitación visual, es decir más de 2.200 millones de personas. El mayor número de estas personas viven en la zona rural, con más de 60 años y pertenecen a algunos tipos de etnias e indígenas. Por regiones, la situación más crítica se encuentra en África seguida Asia meridional.[1]

Debido a lo anterior, en los últimos años las personas con problemas de visión han venido ganando muchas batallas legales en el mundo, por su puesto Colombia no es la excepción. Esta es la razón del presente trabajo, buscar una forma de facilitarle la vida a las personas con pérdida total de la visión.

Documento recibido el 8 de noviembre de 2022. Este artículo fue apoyado del trabajo de grado titulado Sistema de orientacion en ambientes cerrados para el apoyo a persona con ceguera total o J. Arcos, Facultad de ingeniería programa de electrónica, AUNAR, Pasto – Colombia (correspondiente al autor - Cel: 3126549055; e-mail: jeffersonarcosgil@gmail.com).

Todos los avances a nivel tecnológico sobre los algoritmos para la detección de objetos, han mejorado debido al uso de las redes neuronales específicamente las convolucionales. También, se ha venido incluyendo el uso de la GPU para que el proceso de localización se pueda hacer de una manera más rápida y lograr hacerlo en video en vivo, que es lo que se busca implementar en este trabajo. [2]

Finalmente, sean venido crean varios marcos para poder programar e implementar este tipo de soluciones, uno de ellos es denominado YOLO, el cual trabaja con una red neuronal convolucional que aprende a partir de una gran cantidad de imágenes, las cuales las divide y le da una probabilidad asignado pesos a cada uno. Es posible analizar cerca de 45 cuadro por segundo lo que lo hace muy efectivo para el presente caso. Otra gran ventaja es que se puede programar en varios tipos de lenguajes como Java, C++ y Python.

II. PROCEDIMIENTO PARA EL DESARROLLO DEL TRABAJO

La presente investigación se enfoca en las personas que presentan discapacidad visual siendo una alteración del sentido de la vista, que puede ver comprometido los nervios que están presentes en el ojo, como también en la estructura que compone el complejo sistema del mismo, esto se puede presentar en diferentes grados que va desde leve hasta total, las estadísticas dictan que en el mundo hay un total de 2.200 millones de personas que presentan una afectación o deficiencia en su visión, estos datos los arroja la Organización Mundial de la Salud en su informe del 2018.[3]

Por tal razón se busca desarrollar un sistema que satisfaga las necesidades de esta población, para mejorar su calidad de vida y hacerlos más independientes en cuanto a su ambiente, enfocándonos en los espacios en los cuales haya un desplazamiento constante “casa” y así eliminar las dificultades que presentan estas personas a la hora de desplazarse por estas áreas; todo esto se pretende realizar a través de un algoritmo capaz de reconocer lo distintos objetos que están presenten en los entornos cerrados que este tipo de personas habitan, logrando distinguir un espacio tridimensional en el cual los objetos están presentes, llevando a cabo una realimentación en

el cual la persona tenga la información de los obstáculos presentes en su área.

I) Redes neuronales

Las redes neuronales son modelos simples del funcionamiento del sistema nervioso. Las unidades básicas son las neuronas, que generalmente se organizan en capas, como se muestra en la siguiente ilustración. Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona simultaneando un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas.[4]

II) Redes neuronales convolucional

La CNN es un ejemplo de Red Neuronal Artificial con amaestramiento supervisado que procesa sus capas imitando al córtex óptico del ojo individuo para determinar distintas características en las entradas que en definitiva hacen que pueda establecer objetos y "ver".
Este tipo de redes son las más apropiadas, pues aplican una expresión matemática simple pero que se repite varias veces para ir minimizando el problema y dar a cada imagen un numero el cual sirve de referencia para saber si exista la posibilidad de que este un objeto determinado.[5]

III) OpenCV

OpenCV (Biblioteca de visión artificial de código abierto) es una biblioteca de software de aprendizaje automático y visión artificial de código abierto. OpenCV se creó para proporcionar una infraestructura común para las aplicaciones de visión por computadora y para acelerar el uso de la percepción de la máquina en los productos comerciales. Al ser un producto con licencia BSD, OpenCV facilita que las empresas utilicen y modifiquen el código.
Gracias a esta librería, que contiene más de 2500 algoritmos listos para ser usados en cuanto a la visión artificial y el aprendizaje, se maneja los datos que vienen de la cámara y luego se puede cambiar algunos parámetros importantes de cada imagen, como son el tamaño y la cantidad de colores que se quieran manejar.[6]

IV) PyTorch

PyTorch es una biblioteca de tensores optimizada para el aprendizaje profundo mediante GPU y CPU. Estas funciones se mantendrán a largo plazo y, por lo general, no debería haber grandes limitaciones de rendimiento ni lagunas en la documentación. También esperamos mantener la compatibilidad con versiones anteriores (aunque pueden ocurrir cambios importantes y se avisará con un lanzamiento por adelantado).
Esta librería permite hacer uso de los tensores que son matrices tridimensionales, las cuales se les aplica las diferentes operaciones con el fin de encontrar las otras posibilidades de que exista un objeto que pertenece a las clases de tipos en su base de datos.

V) YOLO

Un nuevo enfoque para la detección de objetos. En trabajos anteriores la detección de objetos reutiliza los clasificadores para realizar la detección. En cambio, enmarcamos la detección de objetos como un problema de regresión a cuadros delimitadores espacialmente separados y probabilidades de clase asociadas.
YOLO es el corazón del sistema, es quien integra los algoritmos de OpenCV y de PyTorch, así como la CNN red neuronal convolucional. Es quien determina la manera apropiada de buscar objetos en imágenes fijas, es volviendo el problema a una regresión de cuadros y así simplificar el problema grande a uno más pequeño y así hasta llegar a un tensor más pequeño posible, pero con toda la información importante.[7]

VI) Darknet

Darknet es un marco de red neuronal de código abierto escrito en C y CUDA. Es rápido, fácil de instalar y admite el cálculo de CPU y GPU.[8]

III. Resultados

En forma general, se puede hablar de la visión artificial como una disciplina de la ciencia que busca como obtener, estudiar, procesar y comprender las imágenes del mundo real, para de estas poder conseguir diferente tipo de información. Muchos desarrollos en esta área se han utilizado en detección de eventos, reconstrucción de sucesos, recuperación de imágenes y reconocimiento de objetos.
También se debe tener en cuenta que, con el desarrollo de la tecnología informática y la introducción de microprocesadores más complejos y rápidos, la historia de la visión artificial marcó su punto de inflexión en la década de los 80´s, dando lugar a microprocesadores capaces de capturar, procesar y reproducir simultáneamente imágenes capturadas por una cámara que puede ser conectado de forma remota.
Como se ve, una de las aplicaciones de la visión artificial es la localización y reconocimiento de objetos, tema de esta investigación, dentro de la detección de objetos en una imagen o video, se puede separar dos tipos también, la extracción de característica y la búsqueda de objetos. Para el presente trabajo se trata de identificar la presencia de objetos con el fin de ser una herramienta que ayude a las personas con discapacidad visual a llevar una vida más cómoda.
Es clave tener en cuenta que, para llegar a determinar la presencia de objetos en ambientes tridimensionales, primero se debe buscar la manera de adquirir las imágenes o videos del lugar, luego de esos datos, determinar la presencia de objetos y finalmente determinar qué tan preciso es su localización.

I) Adquisición de la cámara

Se realiza una descripción de cómo se realizó la implementación del módulo de captura de datos de la cámara.

En el código principal de Python se debe agregar la librería de OpenCV y esto se realiza con la instrucción:

```
import cv2
```

En esta librería se encuentran todas las opciones para el manejo de los datos que provienen de la cámara o también desde un archivo de video preexistente.

También, es necesario instalar la librería Torch, que contiene muchas funciones para aprendizaje automático, en la actualidad esta librería es la base de PyTorch, más completa con mayores beneficios y diseñada por FAIR (Facebook's AI Research lab) el Laboratorio de Investigación de Inteligencia Artificial de Facebook. La principal ventaja de Torch son las funciones con la que se puede trabajar con Tensores, que es un arreglo flexible N-dimencional. Para importa Torch se usa las instrucciones:

```
import torch
from torch.autograd import Variable
```

Por otra parte, para poder sacarle el mejor provecho a la tarjeta de video, si se tiene, se comprueba si esta es compatible con CUDA. Esto sí hizo con la siguiente instrucción:

```
device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
```

Si existe el hardware disponible y compatible con CUDA entonces se habilita la GPU para trabajar en paralelo, sino se cumple entonces solo se trabajará únicamente con la CPU. Hay que decir, que cuando se trabaja con la GPU el proceso se ejecuta más rápidamente, alcanzando hacer un análisis de hasta 45 cuadros por segundo.
Con esto, se hace el análisis cuadro a cuadro, haciendo uso de la siguiente instrucción:

```
frame = cap.read()
```

La función cap.read lo que hace es tomar un cuadro del video que se encuentra en un archivo o de la cámara de video y asignarlo a la variable frame. Luego se hacen algunas trasformaciones con en la siguiente instrucción:

```
frame = cv2.resize(frame, (1280, 960),
interpolation=cv2.INTER_CUBIC)
```

Aquí se usó la función resize que sirve para cambiar el tamaño de la imagen, en este caso se la deja de 1280 x 960, por otra parte, se realiza la interpolación del tipo bicúbica también llamada convolución cúbica, este es un algoritmo complejo que esta explicado en el marco teórico.

Algo importante a tener en cuenta, es que la biblioteca de OpenCV captura las imágenes en el formato RGB, pero en el orden BGR, razón por la que hay que implementar dos funciones que realicen este cambio para que en el momento de ver le video los colores no se cambien. Así que luego de redimensionar e interpolar se aplican estas funciones que en el código se denominan:

```
def Convertir_RGB(img):
def Convertir_BGR(img):
```

Con esto se termina el proceso de captura y adaptación de la información que viene de la cámara para ahora continuar con la detección de objetos. Para entender mejor este proceso se muestra el siguiente algoritmo:

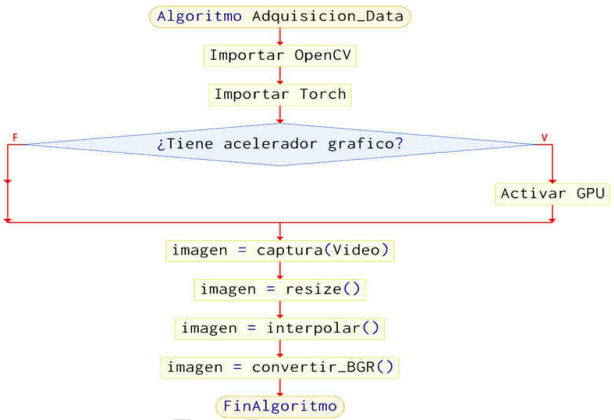


Fig. 1. Algoritmo para captura de datos de la cámara

II) Implementación de algoritmo

Lo primero que se hace es crear un ambiente usando la herramienta Anaconda, que es una distribución para los lenguajes R y Python, que ayuda a los programadores a la gestión e implementación de los paquetes. Esta herramienta tiene un sistema de gestión de paquetes llamado *conda,* que es muy sencillo de instalar, ejecutar y tener actualizado, es muy usado para el análisis de datos científicos (Data science) y el aprendizaje de máquina (machine learnin). Se usa el siguiente comando.
conda create -n yolov3 python=3.6

Luego de crear el ambiente este se debe activar, con el fin de que todos los paquetes que se vaya a instalar queden en el ambiente que acabamos de crear, se hace con la siguiente sentencia:
conda activate yolov3

Ahora si se procede a instalar todos los paquetes que son necesarios para ejecutar el programa de detección de objetos en video. Son varios por lo que la manera más fácil de hacerlo es usando un archivo plano donde tenga todos los paquetes y las versiones que se van a usar. Dentro de los paquetes que se instalan están OpenCV, Torch, tensorflow entre otros, el archivo de texto se llama paquetes.txt y se ejecuta la siguiente instrucción:
pip install -r paquetes.txt

Aquí ya queda listo todo, solo faltaría hacer el proceso de entrenamiento. En este punto podemos tomar dos decisiones, una es tomar ya clases pre entrenadas para este modelo o

realizar el entrenamiento particular para que detecto lo que se quiera en específico. Para la primera opción, se encontró un sitio donde ya estaban las imágenes, con los respectivos pesos para alimentar la red neuronal. Esto se hace mediante el siguiente comando y la url de donde se encuentra la información:

wget -c https://pjreddie.com/media/files/yolov3.weights

Para la segunda opción, se debe entregar a la red neuronal, para que aprende sobre las clases de objetos que se quieren detectar, primero un gran número de imágenes de los objetos, estos deben ser 500 o si quiere mayor precisión más de 1.000 imágenes etiquetadas. Se debe tener en cuenta que también existen imágenes en bases de datos ya etiquetadas, pero si lo que uno quiere es algo muy particular, se deben tomar más de 1.000 imágenes de la clase de se desea.

Para etiquetar las imágenes se usó el programa LabelImg, el cual permite etiquetar imágenes en formatos PascalVOC o YOLO. El programa carga las 1000 o más imágenes que ya están listas y se procede a etiquetar, que es simplemente, encerrar en un cuadro el objeto dentro de la imagen. Como ejemplo, en la **¡Error! No se encuentra el origen de la referencia.** se ve la etiqueta de un reloj, pues en la imagen hay platos, anillos y pulseras. Asi se hace con todas las imágenes que se tenga y pertenezcan a la misma clase. El programa permite decirle que clase pertenece el objeto marcado.

Fig. 2. Etiquetado de un reloj en una imagen

El programa crea unos archivos planos, que guardan un número, que es la clase de objeto y luego las coordenadas del cuadro en posición (x1, y1) a (x2, y2). Luego se debe generar un archivo con extensión .cfg, el cual contiene información sobre la red neuronal para correr las detecciones. Este archivo se lo encontró en un repositorio y se agrega como anexo a este documento. También, se debe descargar la estructura de la red neuronal, es del tipo darknet, y se la encontró en un repositorio también. Esto hace que la estructura de los pesos de YOLO se los transfiera a la red para que esta aprenda sobre las nuevas clases.

Para terminar el entrenamiento se ejecuta el siguiente comando:

```
python train.py --model_def config/yolov3-custom.cfg --
  data_config config/custom.data --pretrained_weights
      weights/darknet53.conv.74 --batch_size 5
```

Y finalmente, la ejecución del programa se puede hacer de dos maneras, la primera si es con la cámara conectada al PC y con las clases ya descargadas que logra ejecutando esta línea de código, que es el programa en Python, que se agregara a este documento como anexo:

```
python deteccion_video.py
```

Esta instrucción se cambia por la siguiente, cuando se va usar las clases especiales y con el proceso de entrenamiento que se usó anteriormente. Esta es la línea de código:

```
python deteccion_video.py --model_def config/yolov3-
custom.cfg --checkpoint_model
checkpoints/yolov3_ckpt_99.pth --class_path
data/custom/classes.names  --weights_path
checkpoints/yolov3_ckpt_99.pth  --conf_thres 0.85
```

III) Evaluación del algoritmo

Para obtener la evaluación que definen el grado de precisión del algoritmo, se etiquetaron 8001 imágenes mediante el cual se encontraban diferentes objetos que están presentes en un entorno cerrado, generando múltiples etiquetas en una misma imagen con el fin de abarcar la mayor cantidad de elementos posibles, para validar y entrenar el modelo con mayor precisión.
Para visualizar el algoritmo con un correcto funcionamiento, se define 99 épocas de entrenamiento, cada época hace un reconocimiento de las imágenes etiquetadas con sus respectivas coordenadas para mejorar la detección de los objetos en el espacio tridimensional, debido a que, con épocas inferiores al valor definido el algoritmo proporciona falsos positivos que se definen como “Diferentes objetos que no pertenecen al elemento original”.

Para comprender la precisión con respecto a las épocas definidas en el entrenamiento, se toma como evidencia cuatro épocas dentro del intervalo de la época inicial hasta la época final, entre ellas se tomaron las épocas que más tienen relevancia en el entrenamiento como lo es la época 0, época 33, época 66 y la época final que es la 99, con el propósito de ilustrar los diferentes procesos que ejecuta el sistema.

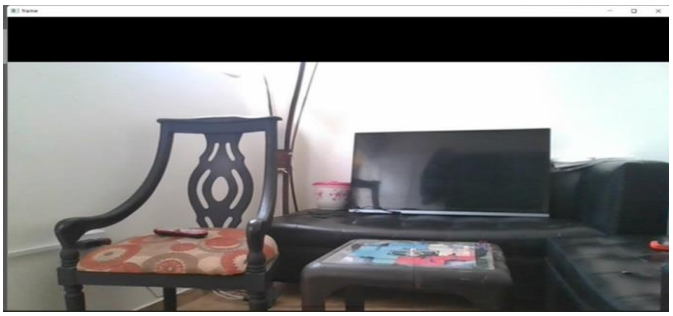


Fig. 3. *época 0 de entrenamiento*

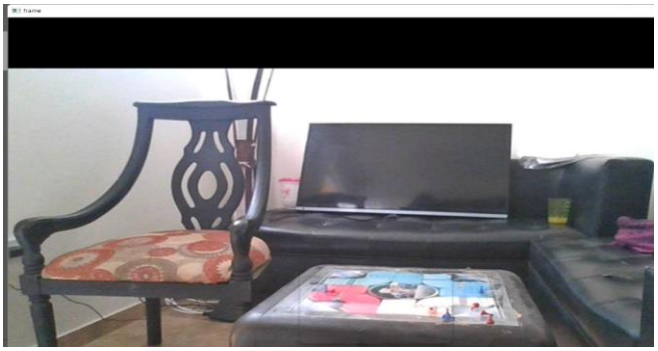


Fig. 4. *época 33 de entrenamiento*

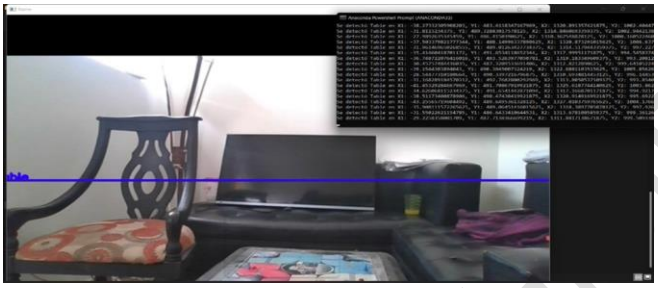


Fig. 5. *época 66 de entrenamiento*

Fig. 6. *época 99 de entrenamiento*

Al observar que el sistema de orientación funciono de manera asertiva en la época 99, se procedió a evaluar y clasificar su precisión en diferentes escenarios, entre ellos se tuvo en cuenta las aulas de la Corporación Universitaria Autónoma de Nariño, la Sala y un Cuarto de un hogar familiar, respecto a que estos espacios son cerrados, y presentan elementos que se aprecia en la cotidianidad.

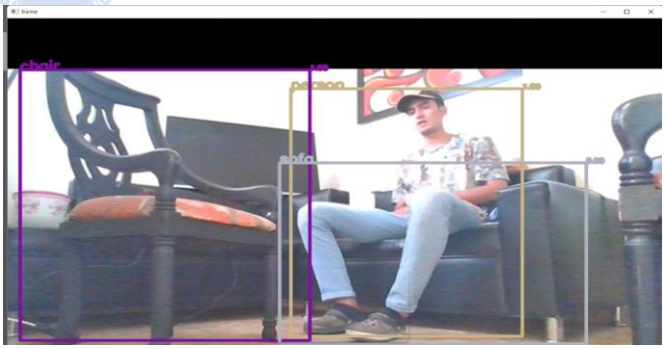


Fig. 7. *Presencia de objetos en un espacio cerrado "sala segundo enfoque".*

Se aprecia que el sistema se evaluó desde un Angulo diferente, donde el algoritmo logra reconocer nuevamente los elementos que se encuentran en la zona con una confiabilidad conforme a las necesidades que se amerita en la investigación, se puede aprecia que reconoció una silla con el 100%, una persona con el 100% y un sofá con el 96%

Fig. 8. *Presencia de objetos en un espacio cerrado "Cuarto"*

Se aprecia que la cámara de video se encuentra en un cuarto, como se puede observar, el algoritmo reconoció una persona con el 95%, una cama con el 99%, una silla con el 97% y otra cama con 96%. Algo relevante a destacar del método en este escenario es que, el sistema a pesar de tener otro objeto "cobija" sobre el elemento que se etiqueto, logra con buena exactitud predecir y detectar el objeto de manera asertiva respecto a que, el número de etiquetas donde se encontró el elemento obtenía objetos muy similares a los que se encuentran en la imagen, por lo tanto, el algoritmo logra comparar y validar la información de manera correcta.

Para evaluar el sistema de orientación en ambientes cerrados con múltiples objetos, se realizó las pruebas en las aulas de la Corporación Universitaria Autónoma de Nariño, con el propósito de evidenciar el comportamiento del sistema frente a objetos de una misma similitud.

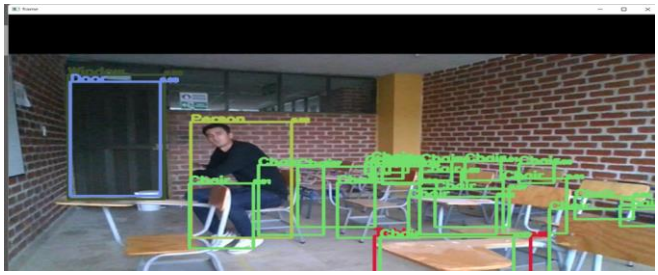


Fig. 9. *Confiabilidad de objetos modelo entrenado con iluminación"*

Se muestra que con mayor iluminación el sistema entrenado encierra con más cuadros de color verde un mismo objeto en el caso de la silla, con valores de confiabilidad entre el 91% y el 96%, también se puede observar que el sistema reconoce una persona con el 92%, una puerta con el 98% y una ventana con el 96%, en un cuadro de color rojo podemos observar un falso positivo, respecto a que la superficie de la silla posee un grado de similitud con respecto a una mesa**.**

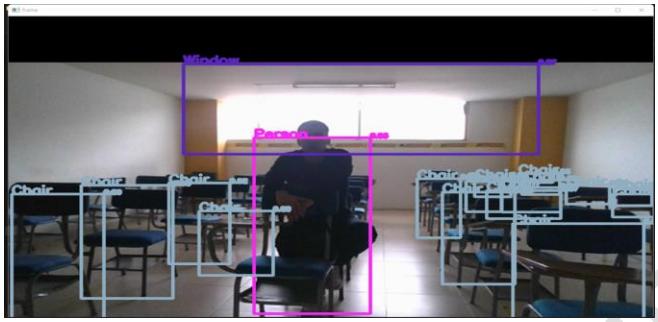


Fig. 10. *Confiabilidad de objetos modelo entrenado*

El modelo precisa los elementos que se encuentran en el aula de clases, indicando que se reconoció una persona con 93%, una venta con él 97% y múltiples sillas con diferentes valores de precisión que rondan entre el 92% y el 99%, esto se debe a que el sistema está recociendo cada estructura del objeto como silla, puesto a que, empieza a evaluar cada parte del objeto etiquetado para reconocer y precisar el elementó que se encuentra en el espacio tridimensional. También se puede destacar que el sistema funciona con poca iluminación, y que el proceso de aprendizaje de la red fue asertivo.

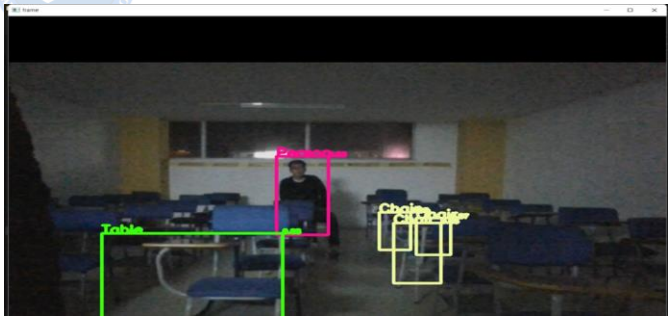


Fig. 11. *Confiabilidad de objetos modelo entrenado sin iluminación*

El sistema está en un aula con poca iluminación donde se puede apreciar que el modelo entrenado no precisa algunos elementos, y marca objetos que no se encuentran en el espacio, como se indica en el cuadro de color verde una mesa en la ubicación de una silla. Observamos que en algunos espacios el modelo no reconoce ningún elemento, pero las áreas donde hay presencia de luz el sistema logro reconocer una persona con el 98% y algunas sillas con el 97%.

Para conocer la distancia que logra detectar el sistema entrenado, se realizó las pruebas en un pasillo de la Corporación Universitaria Autónoma de Nariño en cinco diferentes distancias, para analizar su comportamiento y precisión respecto al reconocimiento de los objetos.

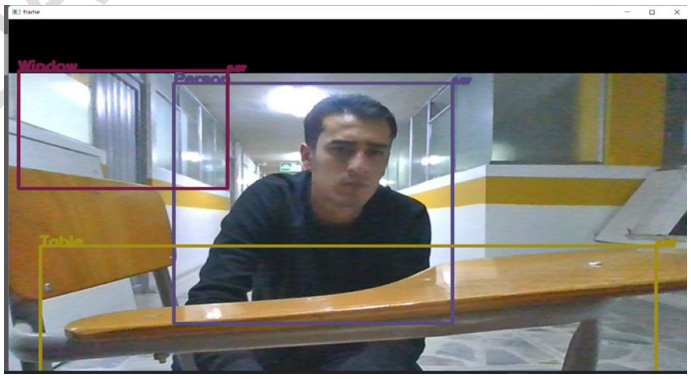


Fig. 12. *Detección de objetos a cincuenta centímetros*

Podemos analizar la distancia inicial que se encuentra a 50 cm donde observamos que el sistema reconoce una ventana con el 97%, una persona con el 97% y detecta un falso positivo marcándolo de color amarillo como mesa.

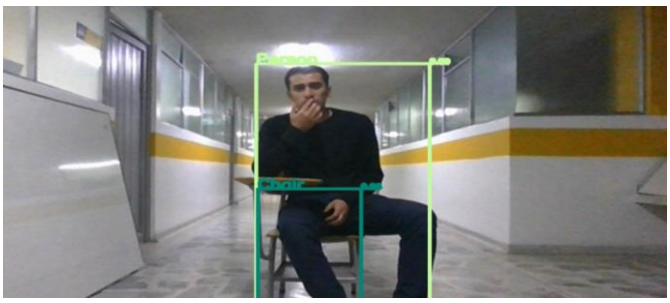


Fig. 13. *Detección de objetos a un metro con sesenta centímetros*

Se muestra que el sistema evidencia una persona con el 99% y una silla con el 95% a una distancia de un metro con sesenta centímetros.

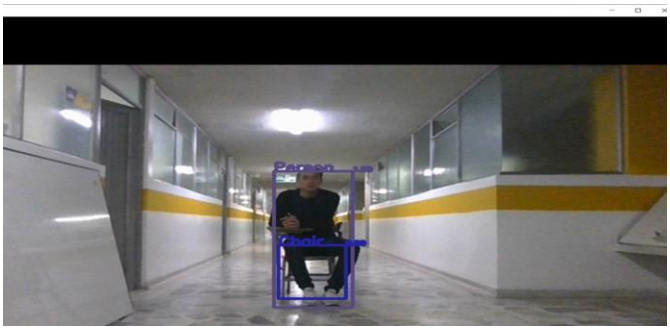


Fig. 14. Detección de objetos en tres metros con veinte centímetros

Se muestra que, el sistema continúa reconociendo una persona con un 100% y una silla con el 99% a una distancia de tres metros con veinte centímetros.

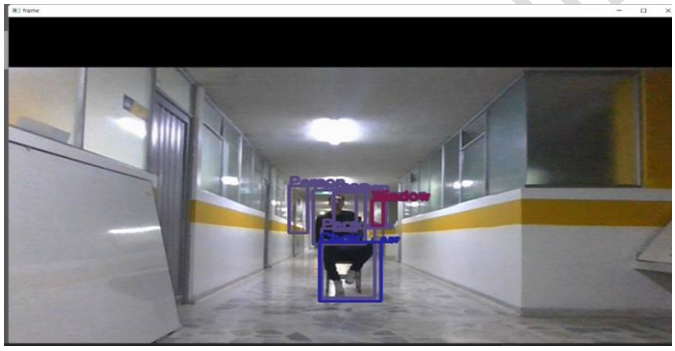


Fig. 15. *Detección de objetos en cuatro metros con ochenta centímetros*

Se muestra que, el sistema detecto la silla con el 97%, como también indica la imagen que se ha reconocido falsos positivos de varias personas y una ventana en la distancia de cuatro metros con ochenta centímetros sin una exactitud asertiva.

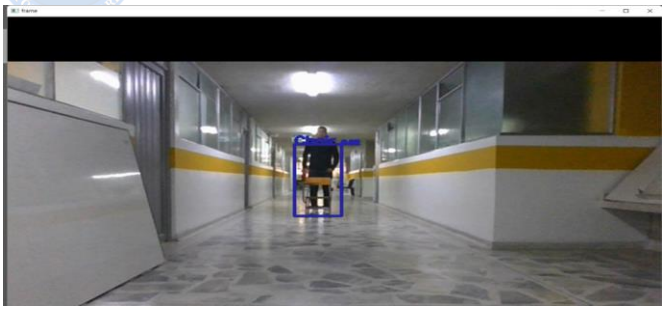


Fig. 16. *Detección de objetos en seis metros con cuarenta centímetros*

Observamos que, el sistema ya no reconoció la presencia de la persona que se encuentra en el espacio, pero continúa detectando la silla con una precisión del 96% a una distancia de seis metros con cuarenta centímetros.

Para evaluar el rendimiento y la precisión del sistema se toma los datos relevantes de las tablas y se hace la comparación con el modelo pre entrenado de Yolo, tomando como referencia las métricas de evaluación y los porcentajes de precisión que proporcionan los algoritmos con relación al espacio y distancia.

Se desarrolla estas comparaciones a partir de graficas de dispersión debido a que permiten conocer la relación entre dos variables numéricas. Para evaluar el sistema se tomó dos elementos que poseen en común los modelos como es el caso de la silla y la persona, se comparan sus resultados en diferente escenario como es el caso de condiciones ideales, baja iluminación, zonas sin iluminación y a diferentes distancias.

Fig. 17. *Comparación de modelos en condiciones ideales "SILLA"*

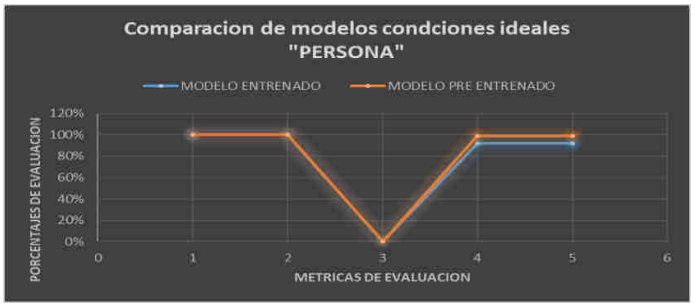


Fig. 18. *Comparación de modelos en condiciones ideales "PERSONA"*

Como se puede observar en la Figura 17, las métricas de evaluación de los sistemas presentan una dispersión con respecto a sus valores, a comparación de la Figura 18 donde podemos apreciar que ambos sistemas tuvieron un grado de similitud con respecto a su precisión y una leve variación con relación a su grado de confiabilidad.

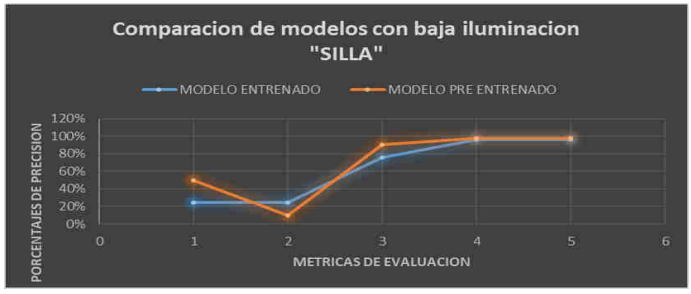


Fig. 19. Comparación de modelos con baja iluminación "SILLA"

La dispersión de datos con relación a los dos modelos es notoria y se puede apreciar su variación en los porcentajes de precisión y en su error con la realidad, pero a comparación con valores de confiabilidad se aprecia que ambos sistemas tienden a obtener valores similares.

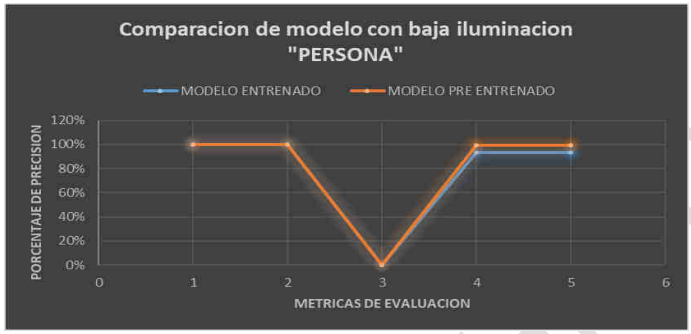


Fig. 20. Comparación de modelos con baja iluminación "PERSONA"

En la Figura 20 evidenciamos que los sistemas continúan obteniendo valores similares en la detección y la precisión de la persona en un espacio diferente.

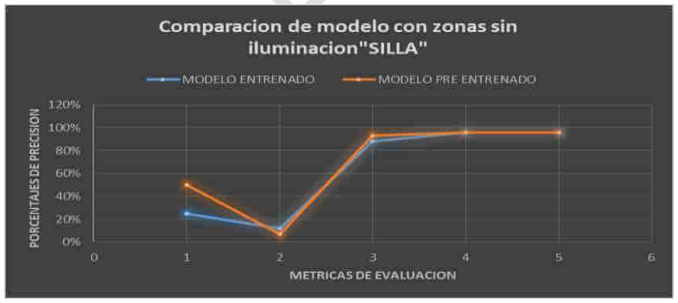


Fig. 21. Comparación de modelos con zonas sin iluminación "SILLA"

Fig. 22. Comparación de modelos con baja iluminación "PERSONA"

Se puede apreciar en la Figura 21 y Figura 22 los porcentajes de precisión de los sistemas, evidenciando que en condiciones de zonas sin iluminación la presencia de los objeto no se logran a detectar en el espacio, el error de la realidad es elevado y la confiabilidad del sistema que proporciona es errónea en el escenario, se observa que la persona al encontrarse en una zona que no presenta buena iluminación, el sistema logra detectar con precisión y confiabilidad del 100%.

En la Figura 23 observamos que las métricas de evaluación son evaluadas a diferentes distancias para evidenciar el comportamiento de los modelos a partir de la detección de los objetos en el espacio tridimensional; se puede apreciar que la precisión del modelo entrenado a 0.50 metros es nula, su detección es mas precisa en las distancias de 1,60 metros hasta 6,40 metros, proporcionando el sistema un valor del 100% , a mayor distancia el sistema no logra reconocer el objeto en el espacio; los grados de confiabilidad en este intervalo son elevados obteniendo así un sistema eficiente en estas distancias.

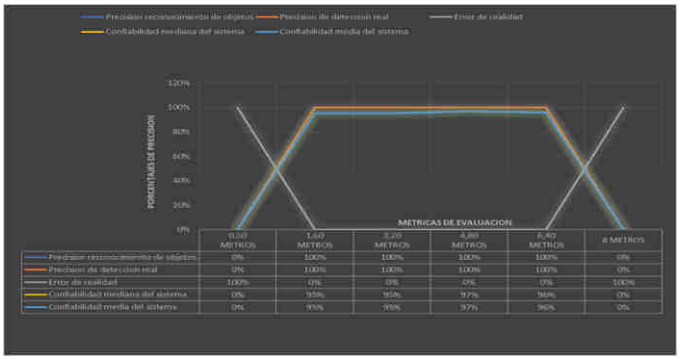
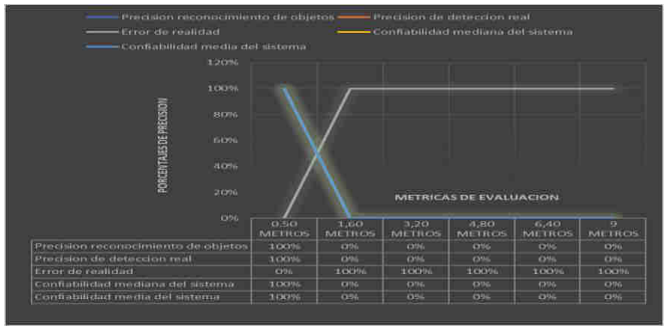


Fig. 23. Comparación de modelos con zonas sin iluminación "SILLA"

La figura 24 nos ilustra los porcentajes de precisión que tiene el modelo pre entrenado con relación a las distancias que tienen la figura 23**,** donde se puede evidenciar que el algoritmo de *yolo* tiene mayor precisión en la detección de los objetos a la distancia de 0.50 metros a diferencia del modelo entrenado, a su vez, obtenemos como resultado que el sistema a medida que la distancia aumenta no logra obtener un porcentaje confiable con respecto a su detección, por lo tanto, su grado de

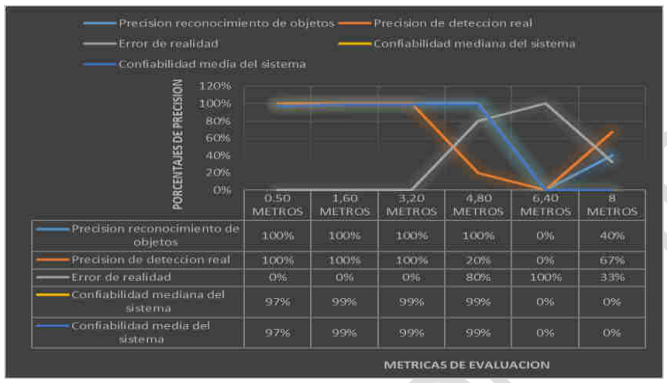
precisión desciende de manera línea volviendo este sistema ineficiente para el reconocimiento de objetos en el espacio tridimensional.



	0.50 METROS	1,60 METROS	3,20 METROS	4,80 METROS	6,40 METROS	8 METROS
Precision reconocimiento de objetos	100%	0%	0%	0%	0%	0%
Precision de deteccion real	100%	0%	0%	0%	0%	0%
Error de realidad	0%	100%	100%	100%	100%	100%
Confiabilidad mediana del sistema	100%	0%	0%	0%	0%	0%
Confiabilidad media del sistema	100%	0%	0%	0%	0%	0%

Fig. 24. Distancia de evaluación modelo pre entrenado "SILLA"

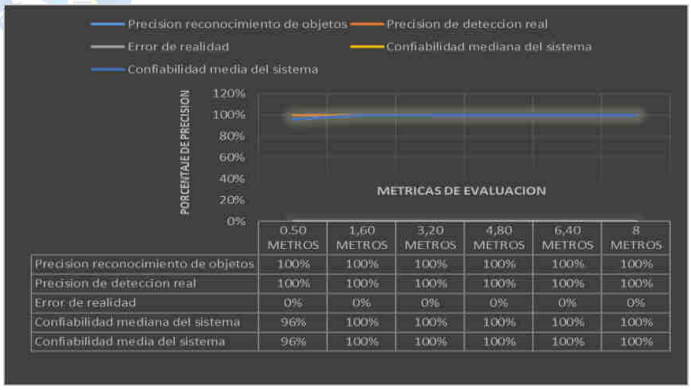
En la Figura 25 se obtiene como resultado que el sistema a comparación de las sillas logra detectar y reconocer la persona en diferentes distancias, mediante el cual el sistema proporciona una precisión del 100% hasta la distancia de 4,80 metros, ya que a partir de esta distancia su sistema baja su grado de precisión proporcionando falsos positivos.



	0.50 METROS	1,60 METROS	3,20 METROS	4,80 METROS	6,40 METROS	8 METROS
Precision reconocimiento de objetos	100%	100%	100%	100%	0%	40%
Precision de deteccion real	100%	100%	100%	20%	0%	67%
Error de realidad	0%	0%	0%	80%	100%	33%
Confiabilidad mediana del sistema	97%	99%	99%	99%	0%	0%
Confiabilidad media del sistema	97%	99%	99%	99%	0%	0%

Fig. 25. Distancia de evaluación modelo pre entrenado "SILLA"

A comparación del modelo entrenado visto en la Figura 46, este sistema marca con precisión del 100% a la persona en las diferentes distancias que se plantearon de evaluación, proporcionando grados confiabilidad a 0.50 metros del 96% y en distancias superiores del 100%, siendo un sistema mucho más eficiente en la detección y reconocimiento de personas.



	0.50 METROS	1,60 METROS	3,20 METROS	4,80 METROS	6,40 METROS	8 METROS
Precision reconocimiento de objetos	100%	100%	100%	100%	100%	100%
Precision de deteccion real	100%	100%	100%	100%	100%	100%
Error de realidad	0%	0%	0%	0%	0%	0%
Confiabilidad mediana del sistema	96%	100%	100%	100%	100%	100%
Confiabilidad media del sistema	96%	100%	100%	100%	100%	100%

Fig. 26. Distancias de evaluación modelo pre entrenado "PERSONA"

IV. CONCLUSIONES

A partir de la investigación se obtuvieron las siguientes conclusiones.

El sistema entrenado funciona de manera correcta cuando hay la presencia de un elemento de varios objetos en el espacio, debido a que, con la presencia de múltiples objetos de la misma forma, color y tamaño el sistema proporciona una detección errónea encerrando la pieza con demasiadas etiquetas, sin definir con precisión el objeto que se visualiza la cámara, a comparación de modelo pre entrenado que no logra reconocer la presencia de muchos objetos, pero si la encierra de manera correcta.

El modelo entrenado y el pre entrenado no funcionan de manera correcta en el reconocimiento de los objetos cuando no existe la presencia de iluminación en el espacio, ya que ambos sistemas no logran dimensionar ni percibir la estructura y silueta del elemento.

El sistema entrenado a la distancia de 0.50 metros es ineficiente en la detección de los objetos, pero en largar distancias y según el posicionamiento de la cámara logra reconocer y conservar los objetos que se hayan entrenado con mayor frecuencia, a comparación del modelo base que posee la estructura de Yolo que logra percibir y dimensionar los elementos que visualiza la cámara de video, pero a distancias superiores a 0.50 metros el algoritmo pierde su efectividad reconociendo solamente las personas en el espacio.

De los resultados obtenidos se define que el sistema entrenado al estar a la distancia de 0.50 metros marca falsos positivos debido a que al etiquetar varias imágenes que poseen características similares de un mismo elemento, hace que el modelo no perciba ni reconozca el objeto de manera correcta.

La precisión con la que los sistemas reconocen los objetos tiene porcentajes elevados alcanzando en el mayor de los casos una exactitud del 100%, pero el valor de reconocimiento lo hacen entorno a la manera como ellos encierran y perciben

el objeto, más no por el número de elementos que se pueden encontrar en un mismo espacio, proporcionando valores erróneos con relación al dimensionamiento de los objetos.

Para obtener una mayor eficiencia de múltiples objetos es necesario que en el proceso de entrenamiento se realice un etiquetado del objeto desde diferentes enfoques, logrando así obtener un reconocimiento más completo.

V. REFERECIAS

[1] Organización mundial de la salud. (26 de 02 de 2021). Obtenido de https://www.who.int/es/news-room/fact-sheets/detail/blindness-and-visual-impairment

[2] Organización Mundial de la Salud. (08 de 10 de 2019). Primer Informe mundial sobre la visión. Obtenido de https://www.who.int/es/news/item/08-10-2019-who-launches-first-world-report-on-vision#

[3] Instituto Nacional Para Ciegos. (08 de 06 de 2018). Los ciegos en el Censo 2018. Obtenido de https://www.inci.gov.co/blog/los-ciegos-en-el-censo-2018.

[4] IBM. (17 de 08 de 2021). Documentación SPSS Modelar. Obtenido de El modelo de redes neuronales: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model

[5] Eighties by Justin Kopepasah. (29 de 11 de 2018). Aprenda Machine Learning en Español. Obtenido de ¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

[6] Doxigen. (25 de 10 de 2021). OpenCV Visión artificial de código abierto. Obtenido de https://docs.opencv.org/4.x/d1/dfb/intro.html

[7] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). Solo mira una vez: Detección unificada de objetos en tiempo real. CVPR 2016, 779-788.

[8] Redmon, J. (2016). Darknet: redes neuronales de código abierto en C. Obtenido de http://pjreddie.com/darknet/