Sistema de detección de robots humanoides en ambientes semiestructurados basado en visión estereoscópica

Detection system for humanoid robots in semi-structured environments based on stereoscopic vision

Oscar Herrera [1], Yesenia González [2], Paola Cortez [3], Benito Granados [4]

Recibido: Junio 2021 Aceptado: Septiembre 2021

Resumen.- El presente trabajo propone el diseño, desarrollo e implementación de un sistema de visión artificial basado en vídeo estéreo, el cual es ejecutado en un sistema embebido, con el fin de identificar a robots humanoides dentro de un área semiestructurada. El sistema embebido utiliza una cámara RealSense de Intel que además de poder obtener distancias hacia los objetos debido a la visión estéreo que posee, es capaz de discriminar información después de un umbral de distancia determinado por el usuario, eliminando los objetos de fondo de la escena y mediante reconocimiento de imágenes basado en una red neuronal convolucional reconoce a robots humanoides dentro de esta. La aplicación del sistema se basa en el concurso Robocup Humanoid League, donde dos equipos de robots juegan fútbol, por lo que además de reconocer a robots humanoides a diferentes ángulos y distancias, el sistema puede clasificar a los robots reconocidos como compañeros o contrincantes (dependiendo del reconocimiento de marcas de color que estos porten), emulando que, a futuro, el sistema propuesto estará montado sobre otro robot humanoide.

Palabras clave: Visión artificial, video estéreo, red neuronal convolucional, robots humanoides.

Summary.- *This work proposes the design, development and implementation of an artificial vision system based on stereo video, which is executed in an embedded system, to identify humanoid robots within a semi-structured area. The embedded system uses an Intel RealSense camera that, in addition to being able to obtain distances to objects due to its stereo vision, is capable of discriminating information after a distance threshold determined by the user, eliminating objects in the background of the scene and, through Image recognition based on a convolutional neural network recognizes the humanoid robots within it. The application of the system is based on the Robocup Humanoid League contest, where two teams of robots play soccer, so in addition to recognizing humanoid robots at different angles and distances, the system can classify recognized robots as companions or opponents (depending on the recognition of color marks they carry), emulating that, in the future, the proposed system will be mounted on another humanoid robot.*

Keywords: *Artificial vision, stereo video, convolutional neural network, humanoid robots*

1 Facultad de Ingeniería, Instituto Politécnico Nacional – UPIITA (México), oherreraf1500@alumno.ipn.mx, ORCID iD: https://orcid.org/0000-0002-7805-3677
2 Doctora en Ciencias, Instituto Politécnico Nacional – UPIITA (México), ygonzalezn@ipn.mx, ORCID iD: https://orcid.org/0000-0003-2370-4660
[3] Maestría en Ciencias, Instituto Politécnico Nacional – UPIITA (México), pcortez@ipn.mx, ORCID iD: https://orcid.org/0000-0003-2338-8581
[4] Doctor en Ciencias, Instituto Politécnico Nacional – UPIITA (México), bgranadosr@cinvestav.mx, ORCID iD: https://orcid.org/0000-0003-1958-7780

1. Introducción. - Cada vez son más los sistemas donde se requiere que la toma de decisiones se realice de manera autónoma de acuerdo con las variaciones que se presenten en las señales de entrada. Esas señales de entrada pueden provenir de diferentes tipos de sensores, uno de ellos son los sensores ópticos (cámaras) y la información que este tipo de sensores adquieran será necesario aplicarle diferentes etapas de procesamiento para lograr evidenciar la información que interese. Además, se requiere que estos sistemas sean robustos, es decir, que mantengan un funcionamiento apropiado aún con variaciones no deseadas en el ambiente. La disciplina de la visión artificial ha ido evolucionando a través de los años, el surgimiento de las redes neuronales de aprendizaje profundo y en específico, de las redes neuronales convolucionales (CNN, por sus siglas en inglés), enfocadas en un inicio en el procesamiento de imágenes, ha permitido tener sistemas más robustos que desempeñan tareas de reconocimiento en ambientes no estructurados. Estos algoritmos también han podido implementarse en sistemas embebidos debido al surgimiento de nuevos sensores y tarjetas de desarrollo con más capacidades e incluso con arquitecturas específicas para la implementación de algoritmos que operan de manera paralela, como es el caso de las redes neuronales artificiales.

Este trabajo propone el reconocimiento y clasificación de robots humanoides usando visión estereoscópica. Esta propuesta se basa en el concurso "RoboCup Humanoids League [1]", donde dos equipos de robots humanoides deben jugar un partido de fútbol de manera autónoma. El juego se lleva a cabo dentro de una cancha de fútbol a escala pero que está rodeada de un entorno no controlado (comúnmente rodeada de espectadores humanos y otros objetos), por lo que la tarea de reconocimiento de objetos de interés representa un reto, desde el punto de vista de visión artificial.

El trabajo está organizado de la siguiente manera: en la sección 2 se describen algunos trabajos relacionados al tema, en la sección 3 se presenta la metodología utilizada, la sección 4 aborda la implementación y resultados obtenidos, en la sección 5 se presentan las pruebas realizadas y por último, en la sección 6 se presentan las conclusiones

2. Trabajos relacionados. – Con los avances de la Inteligencia Artificial, en años recientes se ha incrementado el uso de robots en distintas áreas de la industria y en diversos servicios. Además, muchos de estos robots operan de manera colaborativa, ya sea con humanos o entre robots. Tal es el caso de los robots dedicados al área de logística en compañías como Amazon [2] o robots autónomos para limpieza [3], que, si bien ya son máquinas altamente sofisticadas, el reconocimiento de su entorno o de la cercanía con otros robots lo realizan utilizando tecnologías como lectura de códigos QR, sensores LiDAR (Light Detection and Ranging) o bien utilizando técnicas de visión artificial para la detección de formas sencillas. Sin embargo, si la tendencia es tener entornos cada vez más colaborativos entre robots y humanos, es necesario explorar y mejorar técnicas más sofisticadas de reconocimiento de objetos de interés.

Robocup surgió como una iniciativa científica internacional con el objetivo de promover el estado del arte de los robots inteligentes [4]. Con relación a la categoría "RoboCup Humanoids League", donde dos equipos de robots humanoides deben enfrentarse en un juego de fútbol de manera autónoma, los algoritmos desarrollados para la tarea de detección de robots humanoides han sido diversos, aunque no tan abundantes comparados con otras áreas de desarrollo, debido a que los interesados en esta tarea son pocos grupos de investigación a nivel mundial. En [5] se aborda la importancia de que un robot pueda reconocer a otros si se requiere de la interacción autónoma entre ellos. Algunos de los algoritmos desarrollados incluyen aplicaciones de visión estereoscópica para el cálculo de distancia hacia objetos para su evasión, pero sin realizar el reconocimiento de los objetos [6], la aplicación de diferentes combinaciones de técnicas de procesamiento de imágenes y de extracción de rasgos [7], [8] hasta el surgimiento e implementación de las redes neuronales convolucionales. En [9], se realiza la detección de robots humanoides utilizando diferentes arquitecturas de CNNs con la intención de probar el rendimiento en diferentes sistemas como lo son GPU GTX980, GPU GT620, CPU Intel I5 y CPU Celeron dual Core.

3. Metodología. – Se plantea que el sistema debe reconocer dentro de una escena a robots humanoides y debe clasificarlos como compañero o enemigo (dos clases) tomando como premisa el hecho de que los robots portan un identificador de color en el cuerpo, siempre y cuando los robots se encuentren dentro de un rango de distancia definida por el usuario (dentro del rango operativo de la cámara) a partir del punto de referencia.
La solución propuesta es utilizar un algoritmo de reconocimiento de imágenes basado en una red neuronal convolucional, capaz de identificar robots humanoides dentro de un área delimitada en un espacio controlado haciendo uso de mapas de profundidad [10] y un ordenador de placa reducida (NUC barebone). El mapa de profundidad se obtiene de un módulo de cámaras (Realsense D435) conectado al ordenador de placa reducida en el cual se realiza la tarea de reconocimiento de imágenes y posteriormente la clasificación de los robots. La Figura I muestra la arquitectura del sistema.

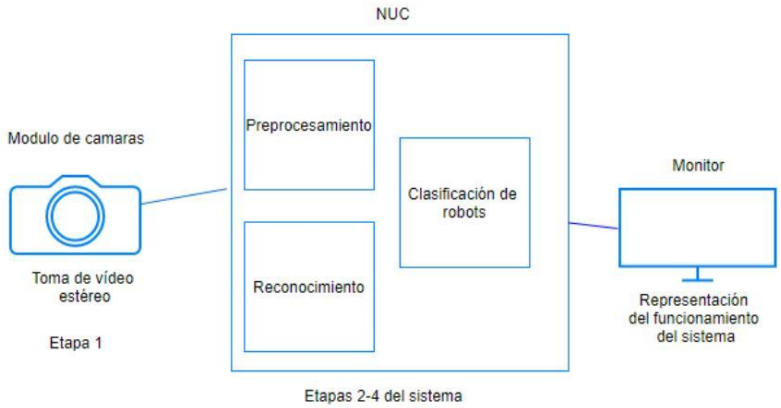


Figura I. – Arquitectura del sistema.

La descripción de la función que realiza cada uno de los componentes físicos del sistema se describe a continuación:

- Módulo de cámaras: Este módulo tiene 2 cámaras y un sensor infrarrojo que permiten el procesamiento de las imágenes para obtener un mapa de profundidad.
- NUC: Dispositivo donde se lleva a cabo el preprocesamiento de la imagen, reconocimiento y clasificación de los robots humanoides.
- Monitor: Dispositivo usado para visualizar los resultados del sistema.

El módulo de cámaras obtiene 2 imágenes (una por cada cámara que contiene el módulo), estas imágenes se procesan de manera que se pueda obtener el mapa de profundidad de la escena y una imagen RGB de esta. Una vez obtenido el mapa de profundidad se buscan en ella todos los píxeles que tengan una distancia mayor a un umbral de distancia definido por el usuario. Al tener localizados los píxeles que cumplan esta condición, se le cambia el color a un valor basado en la escala de grises que el usuario determine. Teniendo la imagen de la escena con el fondo eliminado, se procede a insertarla al algoritmo de reconocimiento, este encuentra y marca a los objetos que se reconocen como robots en un recuadro de color aleatorio. De la imagen resultante del reconocimiento se crean N imágenes donde N es el número total de objetos reconocidos para que en cada recuadro se aplique un filtro de color. Para la propuesta aquí desarrollada, se seleccionaron los colores rojo y azul. Una vez teniendo las áreas de los colores especificados, se reemplazan estos

píxeles de la imagen RGB, así se obtiene la imagen de la escena con los robots clasificados por colores, finalmente se procede a insertar texto en las áreas de color azul con la palabra "Compañero" y en las áreas de color rojos "Enemigo" (puede invertirse esta asignación).

4. Implementación y resultados. – A continuación, se describe la implementación de cada una de las etapas del sistema y los resultados obtenidos. El sistema se implementó con la cámara RealSense D435 de Intel y la NUC Barebone, donde se ejecutó el algoritmo de reconocimiento y clasificación de robots humanoides utilizando el lenguaje de programación Python en su versión 3.8.

4.1 Etapa 1: Toma de video estéreo. – El video estéreo ayuda a obtener el mapa de profundidad, este se obtiene mediante el módulo de la cámara Realsense configurándolo mediante programación y la librería de la que depende el módulo pyrealsense2.

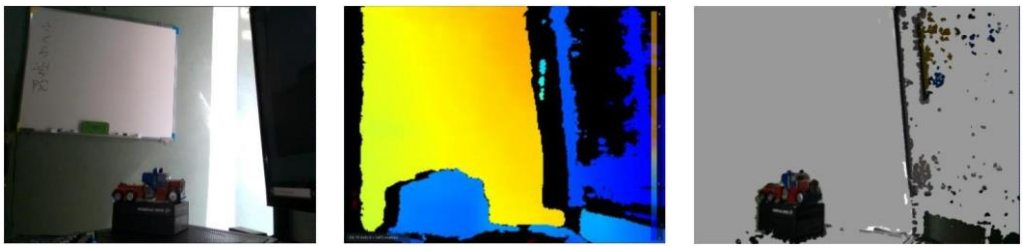
4.2 Etapa 2: Preprocesamiento. – Las imágenes a color que entrega la cámara son en formato RGB y cada imagen a color es un conjunto de 3 matrices de datos con la misma resolución (número de filas y columnas). Aparte, se genera otra matriz con formato Z16 donde cada dato representa la distancia de cada píxel con respecto a la cámara. La librería pyrealsense2 tiene una función llamada "clipping_distance" [11], la cual tiene como argumentos:

a) La matriz de distancia representada con valores numéricos
b) 3 matrices que representan una imagen a color en formato RGB.
c) Distancia máxima de visualización expresada en metros.

En la Ecuación (1) se ejemplifica el formato de ingreso de datos considerando 30 FPS, una resolución de imagen en formato VGA (640 × 480 píxeles) y un umbral de distancia de 1 m.

$$[30, (640,480), 1] \qquad (1)$$

La Figura II muestra un ejemplo de una imagen a la cual se le aplicaron los parámetros de la Ecuación (1). Puede observarse la imagen original en formato de color RGB, el mapa de profundidad obtenido y la discriminación de píxeles con distancias mayores a 1 m, a los cuales se le cambió el color original a color gris (se seleccionó el valor numérico 153).



(a) (b) (c)

Figura II. – a) Imagen original en formato de color RGB. b) Mapa de profundidad. c) Discriminación de píxeles con distancia al punto de referencia mayor a 1 m..

4.3 Creación del banco de datos. – Una vertiente de la robótica es construir robots muy semejantes al humano y en este trabajo lo que se persigue es identificar robots humanoides más semejantes a máquinas. El tipo de humanoides que se quiere reconocer se muestra en la Figura III.

Figura III. Ejemplo de robots que se quiere reconocer.

Para recolectar datos con las características deseadas se usaron videos obtenidos de internet. De estos videos, mediante un programa escrito en Python se obtuvieron todos los fotogramas que lo formaban. El resultado fue un conjunto de datos de 17,280 imágenes. Este conjunto de datos es bastante grande, sin embargo, al ser fotogramas de videos, las imágenes en su mayoría no tenían diferencia, esto es algo que se debe evitar para entrenar a una red neuronal ya que los modelos pueden sufrir de sobreajuste y no funcionan más que con los datos con los que fueron entrenados.

Para evitar esto, se realizó la limpieza de datos, solo se tomaron un numero pequeño de imágenes de estos fotogramas y de las imágenes obtenidas en la web. El resultado fue un conjunto de datos de 485 imágenes para entrenamiento (90 %) y 54 para pruebas (10 %). A estos datos se les aplicó el proceso de etiquetado por medio de un programa en Python que genera un archivo XML con la información de la imagen (nombre, ancho, alto, esquina superior izquierda donde inicia el objeto a reconocer, esquina inferior derecha donde termina el objeto a reconocer). Algunas de las imágenes de la base de datos utilizada para el entrenamiento de la red neuronal se muestran en la Figura IV.

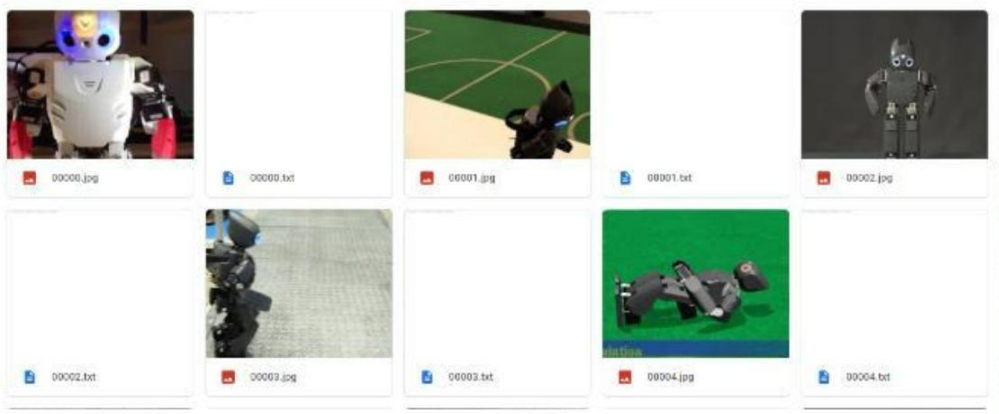


Figura IV. Imágenes del banco de datos utilizado para el entrenamiento de la red neuronal convolucional.

4.4 Etapa 3: Reconocimiento de robots. – En esta etapa se utiliza una red neuronal convolucional. A continuación, se describe el proceso de selección de la arquitectura de red.

El trabajo desarrollado se basó en la arquitectura de red neuronal convolucional YOLO [12], ya que esta cuenta con distintas versiones (extendidas y reducidas), lo que permitió evaluar el desempeño de sus versiones en cuanto a métricas de velocidad de procesamiento y eficiencia. Para elegir una versión para el sistema propuesto, se realizaron pruebas con las versiones 2, 3 y 4 de YOLO. Estas se realizaron bajo las mismas condiciones para que las métricas pudieran ser comparadas sin factores que alteren los resultados.

Para el entrenamiento se utilizó la plataforma GOOGLE Colab, la cual ayuda a ejecutar código en una máquina virtual con la posibilidad de usar GPU dedicada modelo Tesla T4, para acelerar el proceso. El número de épocas (número de barridos sobre los datos de entrenamiento) se estableció de 6000 en el entrenamiento de todas las versiones. Como se puede ver en la Tabla I los tiempos son distintos, siendo la versión 2 la más lenta de todas, sin embargo, la diferencia es mucho menor cuando se compara la versión 3 con la 4.

.

Versión de YOLO	Tiempo de entrenamiento (horas)
2	30
3	14
4	10

Tabla I. Horas de entrenamiento de cada versión de YOLO

El algoritmo de entrenamiento genera archivos con la extensión "weigth" estos archivos se generan cada 1000 épocas, lo que ayuda a entrenar la misma red desde ese punto en caso de que el entrenamiento se interrumpa sin la necesidad de iniciar de cero. De igual forma estos archivos ayudan a monitorear cuando el modelo se sobre ajusta (overfitting).
Para las tres versiones entrenadas, se probaron los archivos de las épocas 1000, 2000, 3000, 4000, 5000 y 6000. Eligiendo para la comparativa el que tuviera menor número de errores en el reconocimiento como lo son falsos negativos y falsos positivos, métricas especificadas a continuación.
Los archivos según el número de épocas que mejor resultado dieron en todas las versiones se muestran en la Tabla II.

Versión de YOLO	Numero de épocas
2	3000
3	3000
4	4000

Tabla II. Numero de épocas de cada versión de la CNN YOLO.

Con esta información, se observa que las versiones 2 y 3 entran en un estado de sobreajuste en un numero de épocas después de las 3000, mientras que la versión 4 después de las 4000 épocas empieza a entrar en ese mismo estado. Esto es importante ya que entre menos épocas pasen para llegar al estado de sobreajuste, más posibles errores se tendrán en pruebas reales [13].
Se realizó la comparación del algoritmo funcionando en tiempo real con 4 videos (videos que no se usaron para el conjunto de datos anteriormente descrito) donde el número máximo de robots en cámara es de 4.
El uso de recursos del sistema tales como lo son CPU y memoria RAM, se muestra en la Tabla III.
.

Versión de YOLO	% de CPU	RAM (en gigabytes)
2	90	3
3	87	3.1
4	85	3.4

Tabla III. Consumo de recursos por versión de red CNN YOLO

En la Figura V se muestra el número de fotogramas por segundo (FPS) promedio de los videos resultantes después de la predicción de cada versión de YOLO.

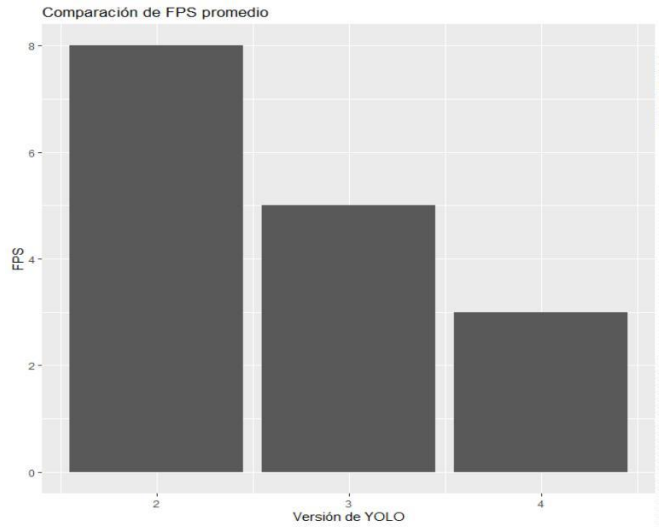


Figura V. Comparación de FPS promedio por versión de YOLO.

Para comparar la eficiencia de predicción de cada versión, se ocuparon las siguientes métricas:

- Verdadero positivo (VP): positivos que fueron clasificados correctamente como positivos.
- Verdadero negativo (VN): negativos que fueron clasificados correctamente como negativos.
- Falso positivo (FP): positivos que fueron clasificados incorrectamente como negativos.
- Falso negativo (FN): negativos que fueron clasificados incorrectamente como positivos.

En la Figura VI se muestran el número promedio de estas métricas de cada versión utilizando matrices de confusión.

Versión YOLO 2

Observación		Predicción: Positivos	Predicción: Negatvos
	Positivos	Verdaderos Positivos (VP) 1	Falsos Negativos (FN) 5
	Negativos	Falsos Positivos (FP) 2	Verdaderos Negativos (VN) 4

(a)

Versión YOLO 3

Observación		Predicción: Positivos	Predicción: Negatvos
	Positivos	Verdaderos Positivos (VP) 2	Falsos Negativos (FN) 2
	Negativos	Falsos Positivos (FP) 4	Verdaderos Negativos (VN) 3

(b)

Versión YOLO 4

Observación		Positivos	Negatvos
	Positivos	Verdaderos Positivos (VP) 3	Falsos Negativos (FN) 0
	Negativos	Falsos Positivos (FP) 2	Verdaderos Negativos (VN) 4
		Predicción	

(c)

Figura VI. Matriz de confusión por versión de arquitectura de red convolucional YOLO.

Con base a los resultados de las métricas realizadas, se seleccionó la arquitectura YOLO versión 4 [14], ya que ofrece un mejor desempeño en cuanto a reconocimiento de los robots y menor tiempo de entrenamiento. El hecho de tener menos fotogramas por segundo no afecta al sistema dado que la velocidad de movimiento de los robots no amerita tener una gran cantidad de imágenes reconocidas por segundo.

La Figura VII muestra un ejemplo de la salida esperada de la etapa de reconocimiento.

Figura VII. Ejemplo de salida de la etapa de reconocimiento.

4.5 Resultados del entrenamiento de la red convolucional. – Como se mencionó en la sección previa, para lograr el reconocimiento de los robots humanoides, se usó la arquitectura YOLO en su versión modificada, desarrollada por los Doctores Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao que denominaron YOLO v4.
Para poder entrenar la red neuronal, se requiere modificar los siguientes parámetros en el archivo de configuración con sus respectivos valores:

- Batch = 64
- Subdivisions = 16
- Max_batches = 6000
- Steps = 4000
- Classes = 1
- Filters = 18
- Width, height = (tamaño de imagen de entrada al sistema)

El valor de batch es el número de ejemplos que se pasan al algoritmo en cada iteración de aprendizaje, el valor de subdivisions es el numero por el cual se divide el número de batch, max batches es el valor resultante de la multiplicación del número de clases por 2000, según la publicación, se debe poner 6000 en caso de tener menos de 3 clases. Steps es el número de ciclos que ejecutará la red neuronal. Classes es el número de objetos que se quieren reconocer, filters es el número de capas que la red tiene.

De igual manera es necesario tener un archivo de pesos previos al entrenamiento, este archivo se genera al usar la red neuronal con clases y base de datos (datasets) preacondicionados; en este caso, el grupo de investigadores proporcionan el archivo en su github de sus pesos resultantes del entrenamiento que hicieron con la base de datos "coco" de GOOGLE [15].

El entrenamiento se realizó en la plataforma GOOGLE Colab, esta plataforma ofrece un uso gratuito limitado de GPU, en específico se utilizó una GPU Tesla T4 de NVIDIA. La gráfica de pérdida y precisión (color azul y café respectivamente) de las etapas de entrenamiento y validación se muestran en la Figura VIII.

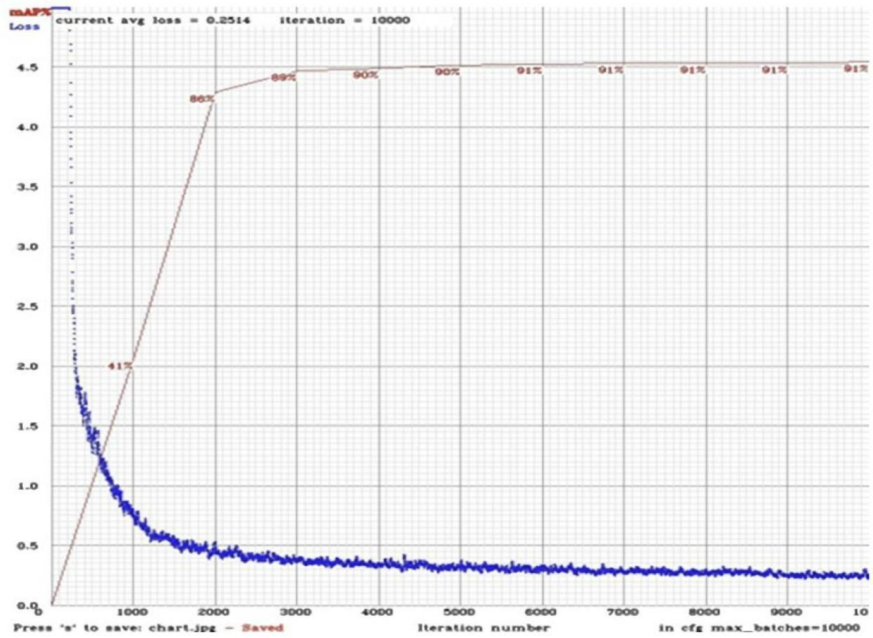


Figura VIII. Gráfica de pérdida y precisión de la etapa de entrenamiento de la red convolucional.

4.6 Etapa 4: Clasificación de robots. – De acuerdo con el reglamento de la categoría "RoboCup Humanoids League", los robots deben tener marcas en ciertas partes de cuerpo de color rojo o azul y dependiendo del color se les asigna la etiqueta de enemigo o compañero. Para este trabajo las marcas se consideraron como rectángulos con los colores mencionados.

Ya que la etapa previa a la clasificación de los robots es el reconocmiento de estos en la escena, la Figura IX ejemplifica un proceso donde fueron reconocidos dos robots y uno de ellos tiene una marca roja y el otro una marca azul.

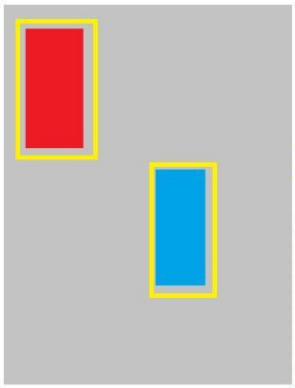
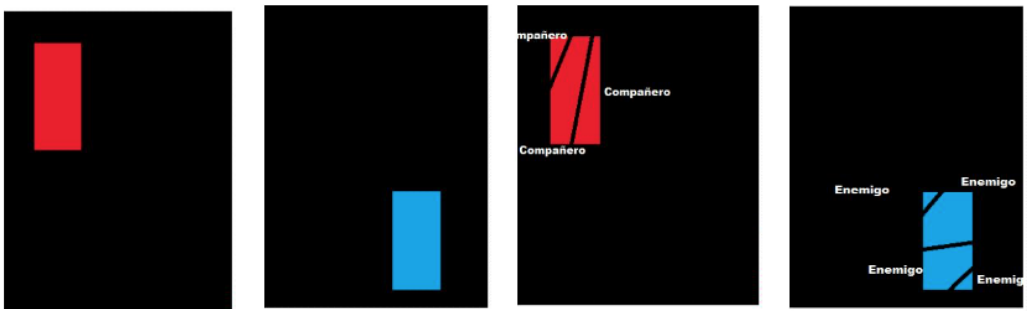


Figura IX. Imagen que ejemplifica el reconocimiento de dos robots en la escena y uno de ellos tiene una marca rectangular de color rojo y el otro una marca rectangular de color azul.

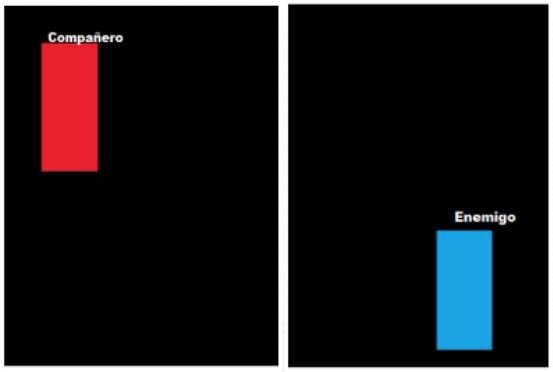
El sistema toma cada elemento (región de la imagen original) reconocido como robot y los procesa de forma independiente (ver Figura X.a) aplicando un segmentador de color [16] para clasificar a cada robot. Si la etapa de segmentación de color reconoce las componentes del color rojo o azul, se agregan entonces en la imagen de salida las etiquetas de "compañero" o "enemigo" a los recuadros donde la CNN reconoció a robots humanoides. Un inconveniente es que puede ocurrir ruido visual, que genere rupturas en las áreas de colores que se busca clasificar, como se muestra en la Figura X.b.



(a) (b)

Figura X. Proceso de separación de objetos reconocidos en una imagen (robots). a) Robot con marca roja y robot con marca azul. b) Representación de las marcas obtenidas por la etapa de segmentación de color afectadas por ruido visual.

Para solucionar la partición de un objeto y tener solo una etiqueta por robot, se aplicó un filtro de desenfoque gaussiano [17], el cual promedia el color de los pixeles a partir del color de los pixeles vecinos. Asi la imagen queda reconstruida como se ejemplifica en la Figura XI.



(a) (b)

Figura XI. Ejemplo de imagen reconstruida.

Finalmente se sobrepone cada píxel extraído y procesado de la imagen RGB sobre la imagen RGB original, como se muestra en la Figura XII.

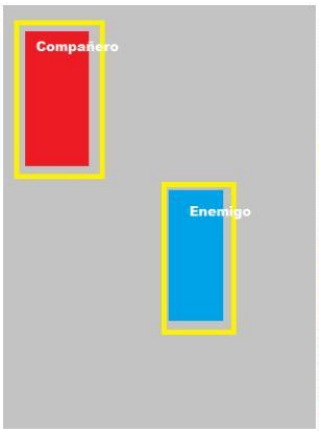


Figura XII. Ejemplo de imagen final del sistema con los recuadros donde se han reconocido a los robots humanoides y se han etiquetado de acuerdo con los colores de las marcas.

4.7 Implementación del sistema en el ordenador de placa reducida. – El sistema se implementó en una tarjeta NUC de Intel® con 16 GB de memoria RAM. El sistema operativo instalado fue UBUNTU 20.04. La Tabla IV presenta las librerías instaladas y sus versiones.

Librería	Versión	Librería	Versión
Python 3	8.0.23	Open CV	3.8.0
Tensor Flow	2.0	Pyrealsense2	2.33.1

Tabla IV. Librerías instaladas en la tarjeta NUC.

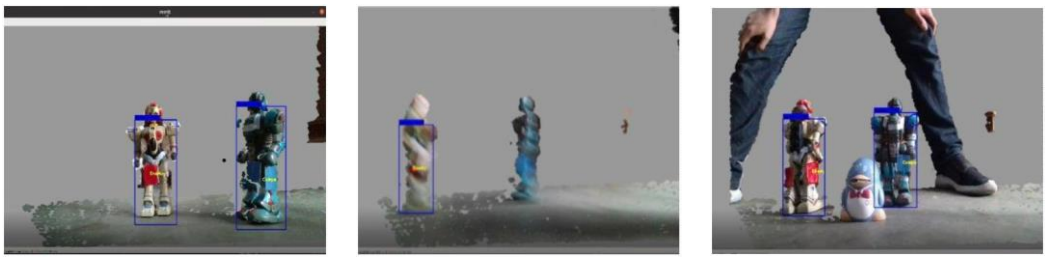
5. Pruebas al sistema. – El sistema fue probado en la tarjeta NUC dentro de una habitación de un área de 3 m2 y como robots de prueba se usaron 2 robots de juguete Metal Fighter marcados con 4 rectángulos de papel de color rojo y azul colocados en sus piernas como se muestra en la Figura XIII.

Figura XIII. Robots usados para las pruebas al sistema.

El sistema logró procesar 4 fotogramas por segundo en promedio. Se usaron 3 escenarios para validar el funcionamiento
del sistema:

- Escenario 1: Cámara y robots estáticos
- Escenario 2: Cámara y robots en movimiento
- Escenario 3: Cámara y robots estáticos con agentes diferentes a los robots humanoides

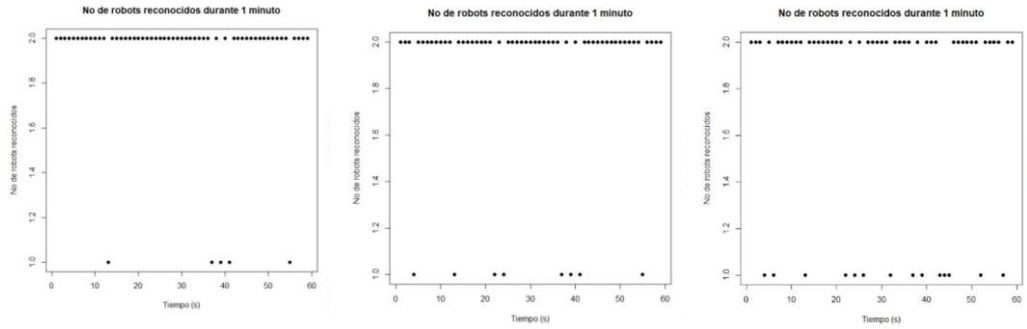
La Figura XIV muestra un ejemplo de las imágenes utilizadas que fueron ingresados al sistema para cada uno de los distintos escenarios de prueba.



(a) (b) (c)

Figura XIV. a) Escenario con cámara y robots estáticos. b) Escenario con cámara y robots en movimiento. c) Escenario con agentes diferentes a robots humanoides.

En el escenario donde los robots y la cámara están en movimiento (escenario más cercano a la realidad) se observaron fotogramas como se muestran en la Figura XIV.b donde no se reconoce a uno de los robots, a esto se le conoce como falso negativo y se da por lo borroso que se captura ese fotograma por el movimiento de ambos dispositivos, sin embargo, estos fotogramas son minoría. Para observar el rendimiento del sistema, se tomó una ventana de 60 segundos y se graficó el número de robots reconocidos por cada segundo siendo 2 el máximo número de robots. La Figura XV muestra la cantidad de robots reconocidos durante el minuto de prueba por cada segundo.



(a) (b) (c)

Figura XV. Pruebas al sistema. A) Escenario 1. B) Escenario 2. C) Escenario 3.

En los tres escenarios probados el sistema logró reconocer a los robots humanoides, aunque el número de falsos positivos y falsos negativos fue mayor en el escenario 3, seguido por el escenario 2. Lo que implica que el sistema presenta más errores al introducir objetos distintos a los robots humanoides en el área de interés.

6. Conclusiones. – De acuerdo con los resultados obtenidos, la posibilidad de reconocer robots humanoides en lugares semiestructurados teniendo como base la toma de video estéreo sí es posible. Al momento de obtener el mapa de profundidad, se cuenta con una imagen en colores donde cada color representa una distancia, por lo que puede obtenerse la distancia en cada píxel de la imagen. La discriminación de fondo se logró de una manera simple, se delimitaron las distancias obtenidas y todo píxel que tuviera una distancia mayor a la establecida, se iluminaría de color gris. Esta delimitación se llevó a cabo en los mismos píxeles en cada capa de la imagen RGB, esto para tener una imagen viable para el reconocimiento.

Con respecto a la etapa de reconocimiento de los robots, se empleó una red neuronal convolucional con arquitectura YOLO, se realizaron pruebas con distintas versiones de esa arquitectura para seleccionar la opción con el mejor compromiso entre FPS de procesamiento, uso de recursos y exactitud entregada. Seleccionando para esta aplicación la versión YOLO 4.0.

Las regiones de la imagen identificadas como robot son procesadas utilizando segmentación de imagen por color en conjunto con un filtro gaussiano, para reconocer marcas de color rojo o azul. Al momento de implementar la clasificación de los robots, hubo problemas ya que, a pesar de tener una imagen con un elemento del color rojo o azul, la cámara ingresaba datos con ruido, lo que hacía que el algoritmo detectara diferentes objetos (etiquetas de color) en donde solo había uno. Este filtro realiza un promedio de los valores de pixeles cercanos entre sí, lo cual genera menor ruido en la imagen. La aplicación de este filtro presentó una mejor relación rendimiento-eficiencia, ya que la cantidad de FPS no bajaba drásticamente al aplicar este procesamiento. El sistema completo procesa 3 FPS en promedio.

Como trabajo futuro se propone comparar los resultados del entrenamiento de la CNN con imágenes sin preprocesar (eliminación de fondo), además de probar el sistema con la base de datos usada en [9], evaluar otras arquitecturas de CNNs e implementar el sistema en tarjetas dedicadas a la Inteligencia Artificial, como las NVIDIA Jetson, con el objetivo de seleccionar los componentes para el diseño y construcción de un robot humanoide.

7. Agradecimientos. – Los autores agradecen a la Secretaría de Investigación y Posgrado del Instituto Politécnico Nacional por el apoyo recibido durante el desarrollo de este proyecto.

8. Referencias

[1] «RoboCup Humanoid League» 2020. [En línea]. Available: https://humanoid.robocup.org [Accessed:18 – abril 2021].
[2] «Amazons collaborative robots offer peek into the future» Available: https://www.businesstimes.com.sg/technology/amazons-collaborative-robots-offer-peek-into-the-future Accessed:14 - septiembre - 2021].
[3] «TASKI. The ultimate cleaning machine» [En línea]. Available: https://taski.com/taski-products/swingobot-000/?lang=es [Accessed:14 - septiembre - 2021].
[4] Robocup Federation official website. Available: https://www.robocup.org [Accessed:18 - abril - 2021].
[5] T. Abbas Shangari, S. Sadeghnejad, J. Baltes. «Importance of Humanoid Robot Detection». In: Goswami A., Vadakkepat P. (eds) Humanoid Robotics A Reference. Springer, Dordrecht. 2018: https://doi.org/10.1007/978-94-007-7194-9_141-1
[6] K. Sabe, M. Fukuchi, JS Gutmann, T. Ohashi,K. Kawamoto, T. Yoshigahara. «Obstacle avoidance and path planning for humanoid robots using stereo vision. In obotics and Automation,» 2004. Proceedings. ICRA'04.2004 IEEE International Conference on 2004 Apr (Vol. 1, pp. 592-597). IEEE.
[7] H. Farazi, S. Behnke. «Real-Time Visual Tracking and Identification for a Team of Homogeneous Humanoid Robots,» In Proceedings of 20th RoboCup International Symposium, Leipzig, Germany, July 2016.
[8] TA. Shangari, V. Shams, B. Azari, F. Shamshirdar, J. Baltes, S. Sadeghnejad. «Inter-humanoid robot interaction with emphasis on detection: a comparison study». The Knowledge Engineering Review. 2017 Feb; 32.
[9] J. Mohammad, A. Sina Mokhtardeh, A. Sajjad, S. Saeed, S. Soroush y B. Jacky, «Humanoid Robot Detection using Deep learning: A speed-Acuracy Tradeoff,» Taiwan, 2017.
[10] A. Saxena, S.H. Chung & A. Ng. «3-D Depth Reconstruction from a Single Still Image». Int J ComputVis 76, 53–69 (2008).
[11] I. R. SDK, «pyrealsense2's documentation,» 2018. [En línea]. Available: https://intelrealsense.github.io/librealsense/python_docs/_generated/pyrealsense2.html.
[12] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.
[13] M. Mitchel Tom, Machine Learning, McGraw-Hill, 1997
[14] A. Bochkovskiy, C.-Y. Wang y M. L. Hong-Yuan, «arXiv.orgLogin,» 23 Apr 2020. [En línea]. Available: https://arxiv.org/pdf/2004.10934.pdf. [Último acceso: Mayo 2020].
[15] A. Bochkovskiy, «github,» 2020. [En línea]. Available: https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137.
[16] R.C. Gonzalez, R. E. Woods. «Digital Image Processing, » Prentice Hall, 3ª edición, 2006.
[17] Siddharth Misra, Yaokun Wu, «Machine learning assisted segmentation of scanning electron microscopy images of organic-rich shales with feature extraction and feature ranking », Machine Learning for Subsurface Characterization, 2020.