

INSTITUTO TECNOLÓGICO DE LA PAZ
DIVISIÓN DE ESTUDIOS PROFESIONALES

REPORTE FINAL DE RESIDENCIA PROFESIONAL

“Reconocimiento y seguimiento de objetos en entornos controlados”

CARRERA

Ingeniería en Sistemas Computacionales

PRESENTA

Eloy Antonio Clemente Rosas

17310793

La Paz, Baja California Sur, México, 20 de Julio del 2022

**“Reconocimiento y seguimiento de objetos en
entornos controlados”**

Resumen

Debido a la escasez y costos de personal de salud y de asistencia en muchos países, los pacientes que están postrados en cama debido a problemas de salud, accidentes o por envejecimiento, en ocasiones no cuentan con el apoyo de personal humano que se ocupe de sus necesidades. Esto ha conducido a realizar esfuerzos por aplicar la robótica y la automatización para cubrir estas tareas de asistencia. De esta forma, se espera que los robot hagan la vida de los pacientes más cómoda, por ejemplo, entregando algún bien requerido o ajustando la temperatura e iluminación de la habitación.

En este proyecto, se presenta un sistema de reconocimiento y clasificación de objetos que se encuentran en su entorno, para un robot de asistencia, así como la estimación de la posición relativa de estos con respecto al robot. Para el reconocimiento y clasificación de los objetos, se aplicaron técnicas de Visión Artificial basadas en herramientas de segmentación semántica, como son Redes Neuronales Convolucionales. Para la estimación de la posición relativa de los objetos, una vez identificados, se implementó una técnica de visión estereoscópica. Los resultados de los experimentos muestran un 90.6 % de precisión en el reconocimiento y clasificación, y un error medio de 5 cm al estimar la posición relativa de los objetos.

Palabras Clave: Visión artificial, Redes neuronales convolucionales, Segmentación semántica, Visión estereoscópica, Inteligencia artificial.

Abstract

Due to the shortage and costs of health and care personnel in many countries, patients who are bedridden due to health problems, accidents or aging, sometimes do not have the support of human personnel to take care of their needs. This has led to efforts to apply robotics and automation to cover these care tasks. In this way, robots are expected to make patients' lives more comfortable, for example, by delivering a required good or adjusting the room temperature and lighting.

In this project, a system for the recognition and classification of objects found in its environment, for an assistive robot, as well as the estimation of their relative position with respect to the robot, is presented. For object recognition and classification, computer vision techniques based on semantic segmentation tools, such as convolutional neural networks, were applied. For the estimation of the relative position of the objects, once identified, a stereoscopic vision technique was implemented. The results of the experiments show a 90.6% accuracy in recognition and classification, and an average error of 5 cm when estimating the relative position of the objects.

Keywords: Computer vision, convolutional neural networks, Semantic segmentation, Stereoscopic vision, Artificial intelligence.

Dedicatoria

Me gustaría dedicar esta Tesis a toda mi familia, pareja, amigos. En conjunto me han enseñado que puedo ser el mejor si me lo propongo. Me han dado todo lo que soy como persona, mis valores, mis principios, mi perseverancia y mi empeño, y todo ello con una gran dosis de amor y sin pedir nunca nada a cambio.

Agradecimientos

En estas líneas quiero agradecer a todas las personas que hicieron posible este proyecto y que de alguna manera estuvieron conmigo en los momentos difíciles, alegres, y tristes. En especial a mí mismo por no rendirme y seguir esforzándome en ser mejor día con día.

Quiero agradecer a mis padres Eloy Clemente Vasquez y Maribel Gabriela Rosas Quijada, maestros MC. Enrique Luna Taylor, en especial a mi pareja Juliette Sánchez Monteverde y hermanos Nicolas y Yahir Clemente Rosas. Por todo su amor, comprensión y apoyo, pero sobre todo gracias infinitas por la paciencia que me han tenido. No tengo palabras para agradecerles las incontables veces que me brindaron su apoyo en todas las decisiones que he tomado a lo largo de mi vida, unas buenas, otras malas, Gracias por darme la libertad de desenvolverme como ser humano.

Índice

Resumen	3
Abstract	4
Dedicatoria	5
Agradecimientos	5
Índice	6
Índice de figuras	7
Índice de tablas	8
Índice de algoritmos	8
Capítulo 1. Introducción	9
1.1 Antecedentes	9
1.2 Objetivos	12
1.2.1 Objetivo General	12
1.2.2 Objetivos específicos	12
1.3 Justificación	12
1.4 Organización del documento	13
Capítulo 2. Planteamiento del problema	14
2.1 Marco Teórico.	14
Inteligencia Artificial	14
Redes Neuronales Artificiales	18
Visión artificial	22
Sistemas embebidos	24
Pytorch	26
2.2 Definición del problema.	26
2.3 Trabajo previo	27
Capítulo 3. Metodología aplicada.	30
3.1 Sistema propuesto	30
3.2 Preparación de imágenes	31
3.3 Entrenamiento del modelo de red neuronal	34
3.4 Reconocimiento y clasificación	39
3.5 Estimación de la posición relativa	42
Capítulo 4. Resultados y Conclusiones	49
4.1 Pruebas de reconocimiento y clasificación	49
4.2 Pruebas de estimación de distancias relativas	53
4.3 Análisis e interpretación de resultados	55
4.4 Conclusiones	56
4.5 Recomendaciones	57
Apéndice 1	58
Preparación del entorno de trabajo	58
Apéndice 2	70
Pre-procesamiento mediante VIA, VGG Image Anotattor	70
Apéndice 3	74
Explicación de código para replicar estos experimentos	74
Creación de archivos de máscaras en formato *png	75
Creación del conjunto de datos en formato Npy	76
Entrenamiento del modelo	78
Configurar la cámara	80
Referencias	81
Anexos.	85

Índice de figuras

Figura 1: Relación entre Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo.	15
Figura 2: Arquitectura general de una Red Neuronal Artificial.....	19
Figura 3: Arquitectura de un perceptrón.....	20
Figura 4: Estructura general de una red neuronal convolucional.....	20
Figura 5: Arquitectura general de la red U-Net.	21
Figura 6: Campo de vista estereoscópico aplicado.	¡Error! Marcador no definido.
Figura 7: Índice de una matriz de píxeles.	23
Figura 8: Imagen original	24
Figura 10: Tarjeta Jetson Nano.....	25
Figura 11: Diagrama general del funcionamiento del sistema.....	30
Figura 12: Delimitación del contorno de los objetos de interés.....	32
Figura 13: Segmentación de los objetos de interés.....	33
Figura 14: Estadística del entrenamiento.....	38
Figura 15: Ejemplo de tensor de probabilidades para tres clases (superior) y del tensor con predicción final de clases (inferior).	40
Figura 16: Imagen original de entrada a la red	41
Figura 17: Segmentación inferida por la red neuronal.....	41
Figura 18: Captura de puntos de control desde dos posiciones distintas.	42
Figura 19: Regresión lineal.....	45
Figura 20: Regresión exponencial.	45
Figura 21: Regresión hiperbólica.....	46
Figura 22: Objeto parcialmente fuera del campo de visión.	48
Figura 23: Ejemplo de la generación de contornos y obtención de centroides.	49
Figura 24: Ejemplos de pruebas de reconocimiento y clasificación.....	51
Figura 25: Mapa de calor de las pruebas de clasificación.....	52
Figura 26: imágenes de la prueba número uno de estimación de distancias	53
Figura 27: Estimación de las distancias con base a centroides.	55
Figura 28: Estimación de las distancias con base a centroides y bordes.....	55
Figura 29: Instalación del disco sólido.	58
Figura 30: Página para descargar el sistema operativo Ubuntu.	59
Figura 31: Menú de selección de arranque.	60
Figura 32: Descompresión de archivos necesarios para la instalación.	60
Figura 33: Menú de prueba o instalación de Ubuntu.	61
Figura 34: Menú de selección de teclado.....	62
Figura 35: Menú de selección de tipo de instalación.....	62
Figura 36: Menú de selección de disco para instalación.....	63
Figura 37: Menú de selección de zona horaria.	63
Figura 38: Formulario donde se registra el nombre de usuario y contraseña.	64
Figura 39: Inicio de Ubuntu.....	64
Figura 40: Resultado de consultar la versión de drivers de CUDA instalados	65
Figura 41: Ejemplo de instalación de drivers recomendados.....	66
Imagen 42: Drivers y versiones instalados.	67
Figura 43: Página principal del software VIA	70
Figura 44: Conjunto de imágenes cargadas.	71
Figura 45: Tipos de seleccionadores disponibles.....	71
Figura 46: Ejemplo de selección de objeto en VIA.	72
Figura 47: Menú de configuración del anotador de clases.....	72
Figura 48: Ejemplo de objeto seleccionado, índice y clase perteneciente en VIA	73
Figura 49: Forma de descargar los archivos del repositorio.	74
Figura 50: Definición de colores de los objetos.....	75
Figura 51: Directorios usados en la creación de máscaras *.png.....	75
Figura 52: Orden en que se va a colorear los objetos.	76
Figura 53: Definición de índices para cada objeto.....	76

Reconocimiento y seguimiento de objetos en entornos controlados

Figura 54 Código para redimensionar las imágenes y mascarar.....	77
Figura 55: Código para genera los conjuntos de datos en formato NPY.	77
Figura 56: Configuración de rutas de archivos *Npy en entrenamiento.	78
Figura 57: Configuración de número de clases en entrenamiento.	78
Figura 58: Configuración de cantidad de imágenes de prueba e imágenes de entrenamiento.	79
Figura 59: Configuración de la red neuronal.	79
Figura 61: Código para captura de imágenes.....	80

Índice de tablas

Tabla 1: Definición de los objetos de interés.....	31
Tabla 2: Especificaciones de la cámara.	32
Tabla 3: Arquitectura de la red neuronal utilizada.....	36
Tabla 4: Parámetros del entrenamiento.	37
Tabla 5: Características del equipo de cómputo para el entrenamiento.....	37
Tabla 6: Desplazamiento de los puntos de control.	43
Tabla 7: Pruebas de reconocimiento y clasificación.....	50
Tabla 8: Pruebas de estimación de la posición relativa de los objetos.....	54
Tabla 9: Resultados de las pruebas de estimación de distancias.....	54

Índice de algoritmos

Algoritmo 1: Generación de archivos PNG con imágenes segmentadas	33
Algoritmo 2: Generación de archivos Numpy	34
Algoritmo 3: Reconocimiento de objetos sobre la Jetson Nano	39
Algoritmo 4: Análisis de Regresión.....	44
Algoritmo 5: Cálculo de Distancia Relativa	47

Capítulo 1. Introducción

Se estima que en las próximas décadas, en algunos países se duplicará el número de personas mayores de 80 años, y que se tendrá un déficit muy importante de profesionales de la salud. Si se pudiera desarrollar robots de cuidado personal capaz de realizar tareas de cuidado y asistencia básicas, el déficit de personal humano necesario se subsanaría significativamente (Yang et al., 2019),

Esto ha conducido a realizar esfuerzos por aplicar la robótica y la automatización para cubrir estas tareas de asistencia. Para que un robot de asistencia personal realice acciones que satisfagan determinadas necesidades fisiológicas, cuando no se cuenta con apoyo humano, debe de contar con las siguientes funciones: estimar su propia ubicación y la de los objetos, llegar al destino deseado evitando obstáculos, manipular los objetos, identificar las necesidades fisiológicas de los pacientes, y razonar sobre los objetos y operaciones necesarias para satisfacer estas necesidades fisiológicas. Sin embargo, a pesar de que se han desarrollado varios tipos de robots para brindar asistencia y cuidados, aún quedan retos importantes por superar, hasta lograr que los robots sean completamente autónomos y realicen de forma efectiva y segura estas tareas (Miseikis et al., 2020).

El objetivo de este proyecto, es aportar en el desarrollo de este tipo de tecnologías, con la intención de que en un futuro estén al alcance de todos los sectores de la población que la requieran.

1.1 Antecedentes

En los últimos años, se han propuesto diferentes sistemas para tareas de asistencia y cuidado personal. Por ejemplo, para este fin, en (Miseikis et al., 2020) presentan el desarrollo de una plataforma de robot móvil con un brazo multifuncional, combinando dispositivos de visión, audio, láser, ultrasonido y sensores mecánicos, controlados a través del sistema ROS (Robotic Operating System) y algoritmos de aprendizaje profundo. Durante la pandemia de

COVID-19, ajustaron el robot, en corto tiempo, para realizar tareas de desinfección y toma de temperatura.

En (Jishnu et al., 2020) proponen el diseño de un robot de asistencia, controlado a través de comandos de voz, para transportar objetos a cortas y largas distancias. La comunicación humano-robot se establece a través de dispositivos móviles vía Bluetooth. El robot cuenta con una cámara para la detección de los objetos, así como para calcular la distancia aproximada de estos, aplicando el algoritmo YOLO (*You Only Look Once*), basado en Redes Neuronales Convolucionales.

Así mismo, en (Diddeniya et al., 2018) presentan el diseño de un robot que opera en ambientes de oficina, utilizando sensores para visión 3D, controlados también a través del sistema ROS. Los comandos se transmiten a través de un dispositivo móvil, que se conecta con una estación de trabajo central y el robot, vía Wi-Fi.

Independientemente de la aplicación, se han desarrollado una variedad de sistemas para localización de objetos y navegación autónoma. Por ejemplo, en (Ferreira, 2013) muestran el diseño de un vehículo para tareas de navegación autónoma, a partir de un mapa que crea de su entorno el propio sistema. El vehículo recopila la información a través de un sistema de visión, para identificar la trayectoria a seguir y detectar obstáculos. Para completar todas sus tareas, el vehículo es controlador de forma remota.

En (Purwanto et al., 2017) presentan el desarrollo de un sistema de navegación para interiores, a través de un sistema de visión multipunto a nivel del piso, para detectar obstáculos y áreas libres. Con esta información, aplican un sistema de inferencia difusa, para establecer la navegación del vehículo de forma autónoma.

Además, se han desarrollado propuestas enfocadas en el reconocimiento y estimación de la distancia de objetos, a través de visión artificial. Por ejemplo, en (Emani et al., 2019), proponen un sistema de visión estéreo, que parte del reconocimiento de los objetos a través de una arquitectura MobileNet, con la cual obtienen recuadros que encierran a los objetos de

interés. Posteriormente, estiman la distancia de los objetos detectados, aplicando una función de triangulación que toma como datos el desplazamiento en el eje X de los recuadros generados en las imágenes tomadas por ambas cámaras, la distancia entre las cámaras y la longitud focal de estas. En sus experimentos, utilizaron dos tipos de cámaras, la Web-cam rig y la cámara ZED, reportando un Error Cuadrático Medio (MSE) de 48.8 y 58.1, respectivamente para ambos tipos de cámaras.

En (Rahul & Nair, 2018), presentan un sistema que combina aprendizaje profundo y visión estéreo, para la detección y estimación de la distancia de objetos. Utilizan una red neuronal convolucional (CNN), para detectar e identificar los objetos, y después estiman su distancia construyendo una nube de puntos 3D utilizando el método de triangulación. El sistema se probó con una cámara estéreo ZED, logrando una precisión de reconocimiento del 84%, y un error promedio en la estimación de las distancias del 4.7 %.

En (Mukherjee et al., 2020), presentan un sistema basado en la plataforma ROS, para la detección y estimación de distancia de peatones. Para la detección, utilizan una cámara de visión estéreo ZED y la arquitectura de red convolucional YOLOv2. Generan múltiples recuadros delimitadores alrededor de los peatones, con las probabilidades de clase y los centroides de estos recuadros. Para la estimación de la distancia, toman las coordenadas de los centroides y extraen las coordenadas 3D de la nube de puntos correspondiente. Posteriormente, aplican por separado un sistema Leddar M16 y la información de profundidad de un sistema estéreo ZED, para calcular las distancias. Finalmente, aplican una combinación de ambos sistema para mejorar la precisión de la estimación. Sus experimentos reportan un RMSE de 10.08 cm con el sistema Leddar, un RMSE de 15.79 cm con el sistema ZED y un RMSE de 6.94 cm con la fusión de ambos.

1.2 Objetivos

1.2.1 Objetivo General

Diseñar e implementar un sistema de reconocimiento y seguimiento de objetos en entornos controlados.

1.2.2 Objetivos específicos

- Implementar un sistema para el reconocimiento de objetos a través de redes neuronales artificiales.
- Diseñar e implementar algoritmos para la automatización del procesamiento de las imágenes de entrenamiento.
- Diseñar e implementar un sistema para el seguimiento de objetos en entornos controlados.
- Diseñar e implementar un sistema para el cálculo de la distancia relativa de los objetos a la cámara.

1.3 Justificación

El desarrollo de robots de cuidado personal capaces de realizar tareas de cuidado y asistencia básicas, representaría un apoyo invaluable que permitiría reducir significativamente la necesidad de contar con personal humano calificado para la atención de personas que se encuentran postradas en cama.

Sin embargo, a pesar de que se han desarrollado varios tipos de robots para brindar asistencia y cuidados, aún quedan retos importantes por superar, hasta lograr que los robots sean completamente autónomos y realicen de forma efectiva y segura estas tareas. Por otro lado, debido a los altos costos, para la mayoría de las familias resulta casi imposible adquirir las nuevas tecnologías de asistencia que van surgiendo.

Por lo anterior, consideramos importante el desarrollar este tipo de tecnologías de forma local, con la intención de ofrecer alternativas más accesibles en lo económico, así como adaptadas a las necesidades específicas de nuestra región.

1.4 Organización del documento

La tesis está organizada en cinco capítulos, en donde el primer capítulo se encuentra los antecedentes, los objetivos generales y específicos, la justificación. El segundo capítulo establece el planteamiento del problema, el marco teórico que contiene investigaciones previas y consideraciones teóricas en las que se sustenta el proyecto de investigación. El tercer capítulo presenta el sistema propuesto y el desarrollo del proyecto de reconocimiento, clasificación y estimación relativa de la posición de objetos. El cuarto capítulo presenta las pruebas y los resultados obtenidos de reconocimiento, clasificación y estimación de objetos. En el último capítulo se encuentra el análisis de los resultados obtenidos, las discusiones finales, las conclusiones y algunas recomendaciones.

Capítulo 2. Planteamiento del problema.

2.1 Marco Teórico.

Inteligencia Artificial

Durante las últimas décadas han surgido diferentes definiciones de la Inteligencia Artificial (IA). John McCarthy, (2004), considerado uno de los pioneros de la inteligencia artificial, la define como: “La ciencia y la ingeniería para crear máquinas inteligentes, especialmente programas informáticos inteligentes. Está relacionada con la tarea de utilizar ordenadores para comprender la inteligencia humana, pero la IA no tiene por qué limitarse a métodos biológicamente observables.”

Décadas antes de esta definición, la discusión sobre la inteligencia artificial estaba presente el trabajo de Alan Turing, con su interrogante: “¿Puede pensar una máquina?”. Para esto, propone un test, conocido como la "prueba de Turing", donde un interrogador humano trataría de distinguir entre la respuesta de texto de una computadora y la de un humano (Alan Turing, 1950). Si bien esta prueba ha sido objeto de mucho escrutinio desde su publicación, sigue siendo una parte importante de la historia de la IA, así como un concepto en curso dentro de la filosofía, ya que utiliza ideas en torno a la lingüística.

Recientemente, en (Russell y Norvig, 2021), presentan una revisión extensa sobre las distintas definiciones de la Inteligencia Artificial. Las clasifican en cuatro grandes grupos, divididos a su vez entre aquellos que presentan un enfoque humano y los que presentan un enfoque ideal:

Enfoque humano

- Sistemas que piensan como los humanos
- Sistemas que actúan como los humanos

Enfoque ideal

- Sistemas que piensan racionalmente
- Sistemas que actúan racionalmente.

En su forma más simple, la inteligencia artificial es un campo que combina la ciencia informática y los conjuntos de datos robustos para permitir la resolución de problemas. También abarca los sub campos del machine learning y el deep learning, que se mencionan frecuentemente junto con la inteligencia artificial. Estas disciplinas están compuestas por algoritmos de IA que buscan crear sistemas expertos que hagan predicciones o clasificaciones basadas en datos de entrada (Russell & Norvig, 2021).

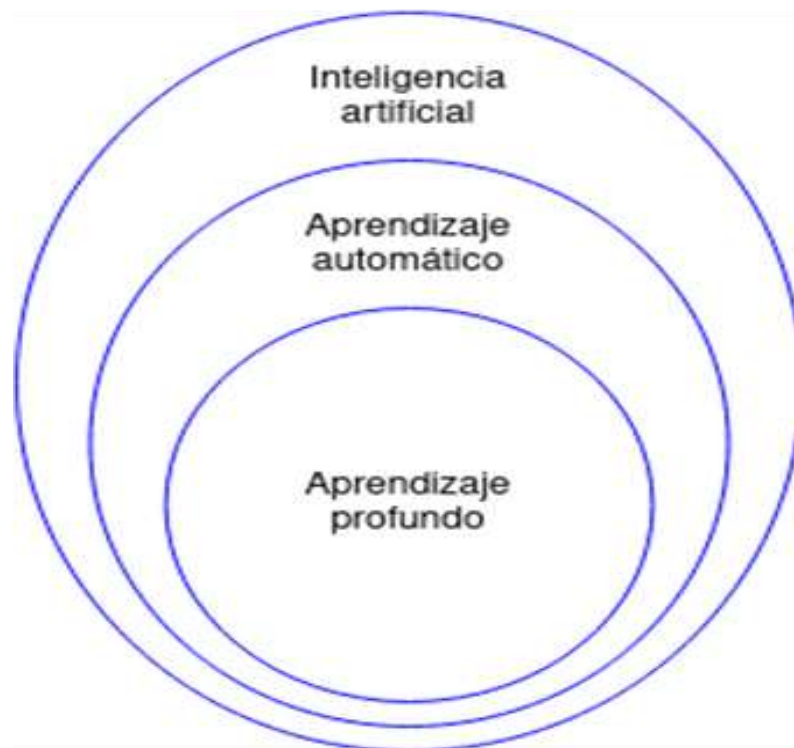


Figura 1: Relación entre Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo.

Aprendizaje Automático

El Aprendizaje Automático (Machine Learning), es una rama de la IA que consiste en lograr que los sistemas aprendan directamente, a partir de muestras de datos, en lugar de que se le enseñe explícitamente mediante programación. Conforme a un algoritmo se le presenta un mayor volumen de muestras de entrenamiento, es posible producir modelos más precisos. Se le nombra *modelo* a la salida que se genera cuando se entrena un algoritmo, el cual se utiliza posteriormente para procesar nuevos datos, los cuales no fueron utilizados en el entrenamiento.

Algunos modelos generados por aprendizaje automático, por ejemplo, modelos para el reconocimiento de objetos a partir de imágenes, están disponibles en línea y pueden ser mejorados o adaptados a nuevos conjuntos de datos. Debido a su complejidad y tamaño, los patrones y asociaciones que son capaces de reconocer algunos modelos, pueden superar la observación humana. Dependiendo de la naturaleza del problema que se está atendiendo, existen diferentes enfoques basados en el tipo y volumen de los datos.

Aprendizaje supervisado

El aprendizaje supervisado comienza típicamente con un conjunto establecido de datos y una cierta comprensión de cómo se clasifican. Estos datos tienen características etiquetadas que definen su significado. Por ejemplo, se puede crear una aplicación con base en imágenes y descripciones escritas que distinga entre millones de animales. Aprendizaje supervisado es una tarea de aprendizaje automático de aprender una función que mapea una entrada a una salida basada en los pares de entrada-salida de ejemplo.

Aprendizaje no supervisado

El aprendizaje no supervisado es la tarea de aprendizaje automático de inferir una función para describir una estructura oculta a partir de datos no marcados. El aprendizaje no supervisado se utiliza cuando el problema requiere una cantidad masiva de datos sin etiquetar.

Por ejemplo, las aplicaciones de redes sociales, tales como Twitter, Instagram y Snapchat, tienen grandes cantidades de datos sin etiquetar. La comprensión del significado detrás de estos datos requiere algoritmos que clasifican los datos con base en los patrones o *clústeres* que encuentra.

Diferencias en el entrenamiento entre aprendizaje supervisado y aprendizaje no supervisado

La diferencia clave Aprendizaje automático supervisado y no supervisado es que el aprendizaje supervisado utiliza datos etiquetados, generalmente se usa para clasificar datos o hacer predicciones, puede tardar mucho el entrenamiento de los datos, además los resultados generados por los métodos de aprendizaje supervisados son más precisos y confiables. Las técnicas más usadas son de clasificación y de regresión. Mientras que el aprendizaje no supervisado utiliza datos no etiquetados, supervisado generalmente se usa para comprender las relaciones dentro de los conjuntos de datos, los resultados generados por los métodos de aprendizaje no supervisados no son muy precisos y confiables. Las técnicas más usadas son la reducción de dimensiones, el análisis de carritos de la compra, la estimación de densidad, y el agrupamiento. (TheBigDataUniversity, 2017).

Aprendizaje de refuerzo

El aprendizaje de refuerzo es un modelo de aprendizaje conductual. El algoritmo recibe retroalimentación del análisis de datos, conduciendo el usuario hacia el mejor resultado. El aprendizaje de refuerzo difiere de otros tipos de aprendizaje supervisado, porque el sistema no está entrenado con el conjunto de datos de ejemplo. Más bien, el sistema aprende a través de la prueba y el error. Por lo tanto, una secuencia de decisiones exitosas conduce al fortalecimiento del proceso, porque es el que resuelve el problema de manera más efectiva (Hurwitz & Kirsches, 2018).

Aprendizaje Profundo

El Aprendizaje Profundo (Deep Learning) es un subconjunto del Aprendizaje Automático, que a su vez es una rama de la Inteligencia Artificial (ver Figura 1). El aprendizaje profundo, está basado en redes neuronales de múltiples capas, que aplican algoritmos inspirados en cómo funciona el cerebro humano y aprenden de grandes cantidades de datos. Estos algoritmos aprenden de forma progresiva a través de tareas repetitivas que le permiten mejorar de manera gradual.

Se puede pensar en el aprendizaje profundo como una rama del aprendizaje automático que requiere un poder computacional robusto y de grandes conjuntos de datos. De esta forma, son capaces de aprender patrones ocultos dentro grandes cantidades de datos, para posteriormente ser capaces de realizar predicciones a partir de nuevos datos (IBM Watson Studio, 2020).

Redes Neuronales Artificiales

Las redes neuronales artificiales se inspiran en el comportamiento del cerebro humano, imitando de forma básica la manera en que las neuronas biológicas se comunican entre sí. De esta forma logran que los programas informáticos reconozcan patrones y resuelvan problemas comunes en los campos de la IA.

Las redes neuronales artificiales están compuestas por una capa de neuronas de entrada, una o más capas ocultas, y una capa de salida. Cada neurona se puede conectar a una o más neuronas, y puede tener un peso, un umbral y una función de activación asociada. Si la salida de cualquier neurona individual está por encima del valor de umbral especificado, dicha neurona se activa, aplica su función asociada, y envía ese dato a las demás neuronas con las que se conecta, comúnmente neuronas de una siguiente capa de la red (ver Figura 2).

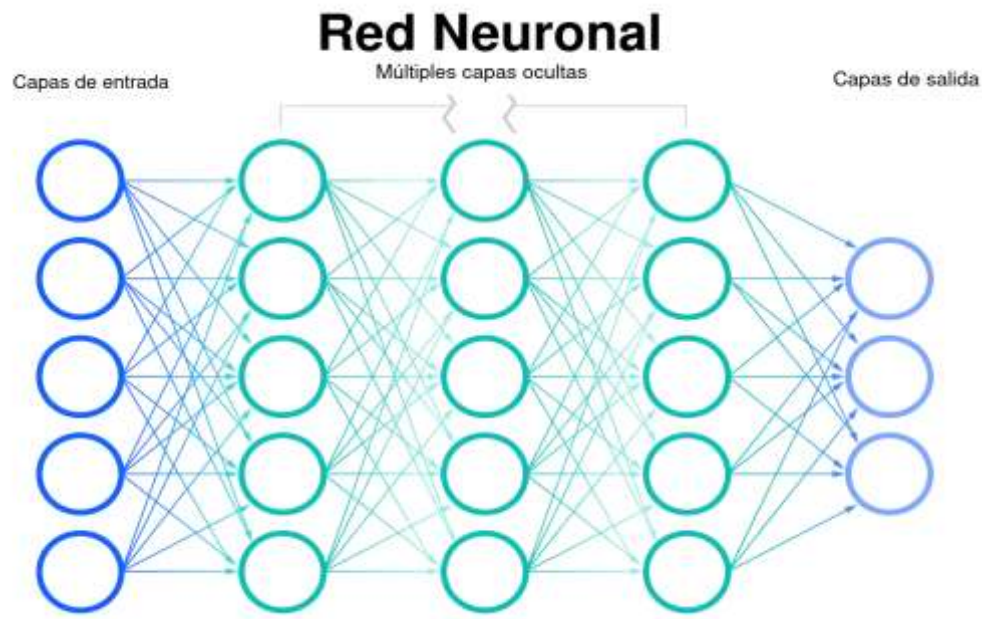


Figura 2: Arquitectura general de una Red Neuronal Artificial.

Las redes neuronales dependen de los datos de entrenamiento para aprender y mejorar su precisión con el tiempo. Sin embargo, una vez que estos algoritmos de aprendizaje se ajustan a la precisión, son potentes herramientas en la ciencia de computación e inteligencia artificial, permitiendo clasificar o agrupar datos a una alta velocidad. Por ejemplo, tareas de reconocimiento de voz o reconocimiento de imagen de grandes volúmenes de datos, pueden tardar minutos frente a horas en comparación con la identificación manual de expertos humanos (IBM Watson Studio, 2020).

Perceptrón

Un Perceptrón es un clasificador binario o discriminador lineal, el cual genera una predicción basándose en un algoritmo combinado con el peso de las entradas (ver Figura 3). El primer algoritmo que presentaba una red neuronal simple se llamó Perceptrón, creado por Frank Rosenblatt en 1958, basado en el trabajo realizado previamente por Santiago Ramón y Cajal y Charles Scott Sherrington, pioneros en el estudio del funcionamiento del cerebro humano (Ramírez, 2018).

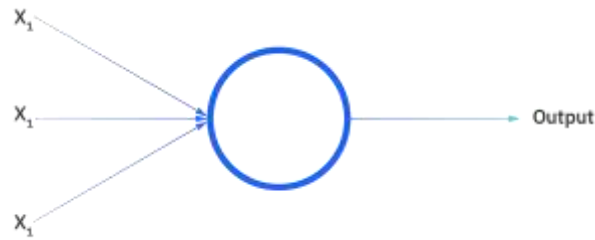


Figura 3: Arquitectura de un perceptrón.

Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales se utilizan regularmente para identificar patrones dentro de una imagen. Este es uno de los principales algoritmos que ha contribuido en el desarrollo y perfeccionamiento del campo de Visión por computadora.

Las redes convolucionales pueden contener varias capas ocultas (ver Figura 4), donde las primeras capas son capaces de reconocer patrones básicos como líneas y curvas, y así se van especializando en las siguientes capas, hasta poder reconocer formas complejas como siluetas o rostros. Tareas comunes de este tipo de redes son: detección o categorización de objetos, clasificación de escenas y clasificación de imágenes en general. (Silva & Freire, 2019).

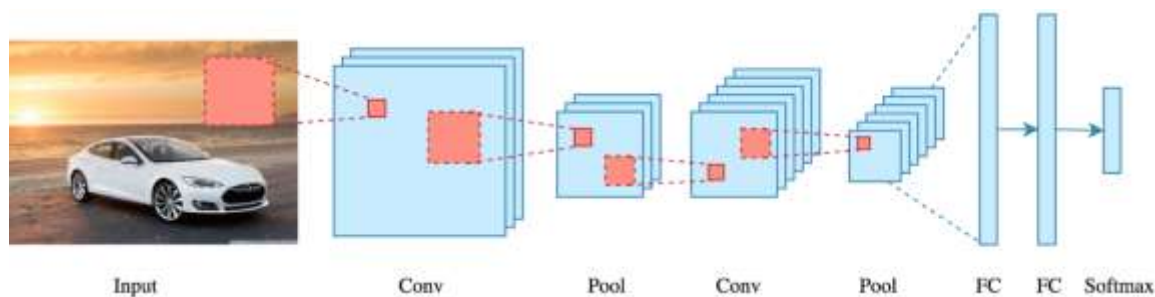
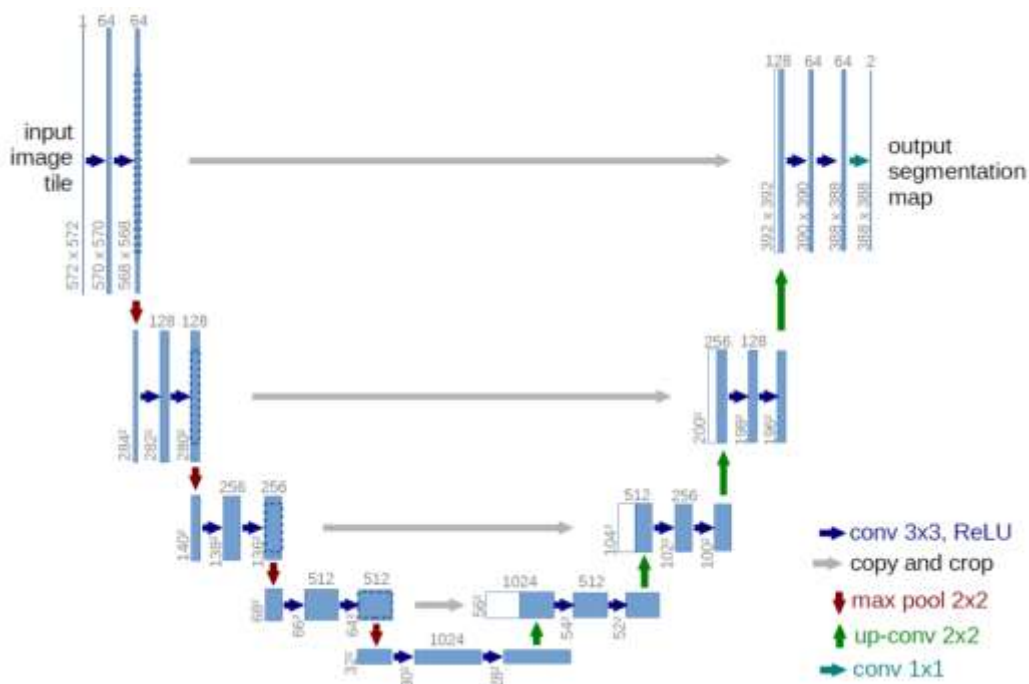


Figura 4: Estructura general de una red neuronal convolucional.

U-Net

U-NET es un modelo de red neuronal dedicado a tareas de visión artificial y más concretamente a problemas de segmentación semántica.

La arquitectura de U-NET consta de dos “vías” (ver Figura 5). La primera es la de la contracción, también llamada codificador. Se utiliza para captar el contexto de una imagen. En realidad, se trata de un conjunto de capas de convolución y de capas de “max pooling” que permiten crear un mapa de características de una imagen y reducir su tamaño para disminuir el número de parámetros de la red. La segunda vía es de expansión simétrica, también llamada decodificador. Permite una localización precisa de la clase de cada pixel de la imagen original, mediante la convolución transpuesta (DataScientest, 2022).



Visión artificial

La visión artificial es un campo de la IA que permite que las computadoras y los sistemas obtengan información significativa de imágenes digitales, videos y otras entradas visuales, y tomen acciones o hagan recomendaciones basadas en esa información. Si la IA permite que las computadoras piensen, la visión artificial les permite ver, observar y comprender. La visión artificial funciona de manera muy similar a la visión humana, excepto que los humanos tienen una ventaja. La vista humana tiene la ventaja de las experiencias y los contextos aprendidos para diferenciar entre los objetos, qué tan lejos están, si se están moviendo o si hay algo mal en una imagen (Marr, 2019).

Visión estereoscópica

La visión estereoscópica es el conjunto de técnicas que intentan recuperar información de profundidad a partir de dos o más vistas de una escena. Nuestros cerebros reciben imágenes similares de una escena tomadas desde dos puntos cercanos y en el mismo nivel horizontal debido a la forma en que están localizados los ojos (ver Figura 6). Dos objetos a distinta distancia del observador presentan posiciones relativas diferentes en sus imágenes. El cerebro es capaz de medir esta diferencia y de usarla para estimar la profundidad (Condori, 2013). Los seres humanos utilizamos dos imágenes para realizar este proceso. En este trabajo también se utiliza un sistema que adquiere dos imágenes para obtener información relacionada con la profundidad de la escena.

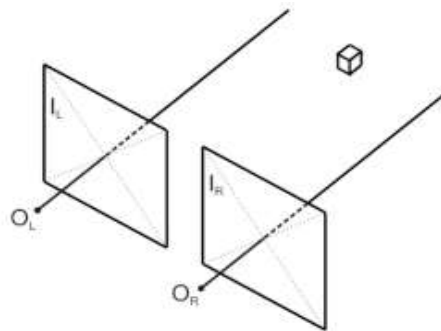


Figura 6: Campo de vista estereoscópico aplicado.

Procesamiento de imágenes

El procesamiento de imágenes está dado por un conjunto de operaciones llevadas a cabo sobre estas, a fin de realizar mediciones cuantitativas para poder describirlas; es decir, extraer ciertas características que permitan mejorar, ajustar, o detallar la imagen.

Para poder realizar mediciones sobre las imágenes se requiere que estén bien definidas sus características, los bordes bien delimitados, el color y el brillo uniformes (ver Figura 7). El tipo de características a realizar es un factor importante para determinar los pasos apropiados para su procesamiento (Ramos, 2003).

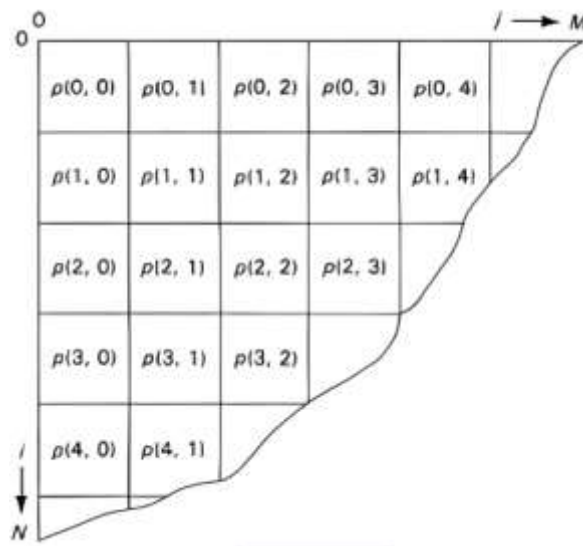


Figura 7: Índice de una matriz de píxeles.

Clasificación de objetos

Una manera muy sencilla de clasificar imágenes es compararla pixel a pixel con un conjunto de imágenes ya conocido y clasificarla como la clase de la imagen más parecida. Esta forma de actuar presenta claros problemas al comparar la imagen entera. Las imágenes se representarán por varias imágenes más pequeñas que definen las características locales extraídas. Para clasificar una imagen se comparan todas y cada una de sus características locales con todas y cada una de las características locales extraídas del conjunto de aprendizaje. Las imágenes se clasifican en la clase que mayor número de coincidencias tenga (Fu. K, et., 1987).

Segmentación Semántica

La Segmentación Semántica consiste en otorgar una etiqueta o categoría a cada píxel de una imagen. Los sistemas de segmentación semántica tienen como objetivo delimitar de forma precisa la categoría de los distintos objetos dentro de una imagen, a nivel de píxel, dando como resultado cualquier forma arbitraria presente (ver Figuras 8 y 9) (Tsung-Yi Li et., 2014).



Figura 8: Imagen original

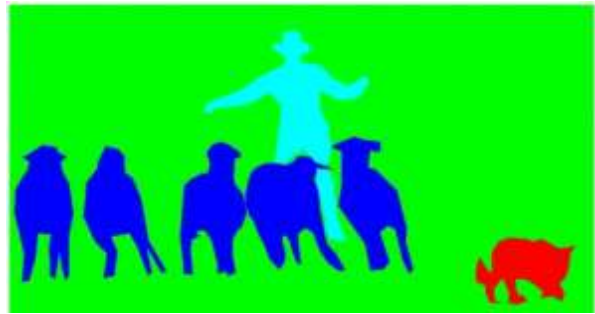


Figura 9: Imagen segmentada

Sistemas embebidos

Un sistema embebido es un sistema de computación basado en un microprocesador o un microcontrolador diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real (Barr, 2007). Existen plataformas desarrolladas por distintos fabricantes que proporcionan herramientas para el desarrollo y diseño de aplicaciones y prototipos con sistemas embebidos desde ambientes gráficos, algunos ejemplos de estas son: Arduino, Mbed, Raspberry Pi, Jetson Nano, etc.

Jetson Nano

La Jetson NANO de NVIDIA es una plataforma de desarrollo para implementar sistemas de inteligencia artificial. Incluye todos los periféricos necesarios para desarrollar un sistema embebido que utilice redes neuronales. Cuenta con un sistema operativo Linux base con el SDK de NVIDIA para desarrollar sistemas de IA en poco tiempo (ver Figura 10) (Explains, 2020).

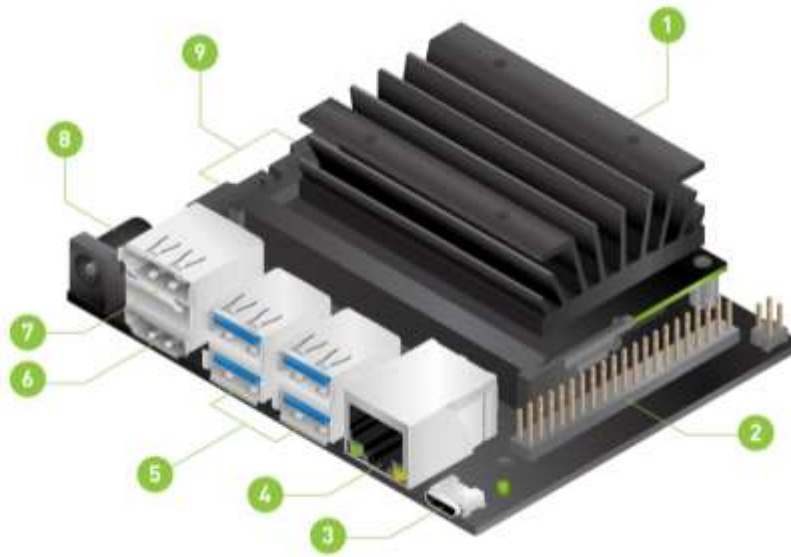


Figura 10: Tarjeta Jetson Nano.

Las características de la tarjeta son:

- GPU de 128 núcleos Maxwell TM GPU
- CPU de cuatro núcleos ARM A57
- Memoria de 4 GB LPDDR4 de 64 bits | 25.6 GB / s
- Adaptador de corriente DC 5V 4A
- USB 3.0 Tipo A y USB 2.0 Micro-B
- Dimensiones: 100 mm x 80 mm x 29 mm.

Pytorch

PyTorch es una biblioteca de aprendizaje automático de código abierto, utilizado para aplicaciones que implementan tareas de visión artificial y procesamiento de lenguajes naturales, desarrollado, principalmente, por el Laboratorio de Investigación de Inteligencia Artificial de Facebook (FAIR).

PyTorch proporciona dos características de alto nivel:

- Computación de Tensores.
- Redes Neuronales Profundas.

PyTorch define una clase Tensor (`torch.Tensor`), para almacenar y operar con matrices numéricas rectangulares, homogéneas y multidimensionales. Los Tensores de PyTorch son similares a los Vectores de NumPy, pero también se pueden operar en una GPU compatible con CUDA (Ketkar & Nikhil, 2017).

2.2 Definición del problema.

Para que un robot de asistencia realice de forma efectiva y segura su función encomendada, es esencial que cuente con la información necesaria y correcta en relación a su entorno. Esta información la adquiere, principalmente, a través de sensores y de sistemas de visión artificial. Uno de los desafíos más importantes en el campo de la visión artificial, es la detección e identificación de posición y orientación de los objetos a partir del análisis de imágenes y videos (Alikarami et al., 2017).

Por otro lado, la presencia de obstáculos, tanto estáticos como dinámicos, dificulta la navegación de los robots autónomos. Se han propuesto diversas estrategias para resolver este problema. Entre otras, se han aplicado sistemas difusos y sensores basados en información visual (Purwanto et al., 2017).

En este proyecto, se propone abordar la problemática de reconocimiento y determinación de la posición relativa de los objetos presentes en una escena, a través del uso de herramientas de la inteligencia artificial, como son las redes neuronales convolucionales.

2.3 Trabajo previo.

Los trabajos relacionados con la detección de objetos en imágenes pueden dividirse en diferentes categorías donde se utilizan estrategias de binarización OVA (Objetos Virtuales de Aprendizaje) y OVO (Uno contra uno) en la clasificación, detección o segmentación. La mayoría de los trabajos de reconocimiento de objetos, clasificación y segmentación de imágenes, solo utilizan modelos clásicos como SVM (Máquinas de vectores soporte), LDA (Análisis discriminante lineal) y kNN (K vecinos más próximos). En la clasificación de imágenes (Rocha, 2018) analizaron el enfoque OVA y OVO para reducir el espacio de características en 3 benchmarks de referencia conocidos, MNIST, ALOI (Biblioteca de imágenes de objetos de Ámsterdam y AUSLAM (Signo australiano lenguaje). Para la segmentación de imágenes (Yu M, Gong L, Et., 2017) compararon un clasificador individual basado en redes neuronales convolucionales contra un OVA y OVO basado en SVN, mostrando que las redes neuronales convolucionales logran un rendimiento mejor.

En los últimos años, se han propuesto diferentes sistemas para tareas de asistencia y cuidado personal. En (Miseikis et al., 2020) se muestra a Lio, una plataforma de robot móvil con un brazo multifuncional diseñado explícitamente para la interacción humano-robot y tareas de asistente de cuidado personal. El robot se ha implementado en varios centros de atención médica, donde funciona de forma autónoma, ayudando al personal y a los pacientes todos los días. Lio es intrínsecamente seguro al tener una cobertura total en material de cuero artificial suave, así como detección de colisiones, velocidad y fuerzas limitadas. Además, el robot tiene un controlador de movimiento compatible. Se utiliza una combinación de sensores visuales, de audio, láser, ultrasónicos y mecánicos para una navegación segura y una mejor comprensión del entorno. La configuración habilitada para ROS permite a los investigadores acceder a datos de sensores sin procesar y tener control directo del robot. La apariencia amigable de Lio ha dado

como resultado que el robot sea bien aceptado por el personal de atención médica y los pacientes. La operación totalmente autónoma es posible gracias a un motor de decisión flexible, navegación autónoma y recarga automática. Combinado con activadores de tareas programadas, esto permite que Lio funcione durante todo el día, con una duración de la batería de hasta 8 horas y recargándose durante los tiempos de inactividad. Una combinación de potentes unidades informáticas proporciona suficiente potencia de procesamiento para implementar inteligencia artificial y soluciones basadas en aprendizaje profundo a bordo del robot sin necesidad de enviar datos confidenciales a los servicios en la nube, lo que garantiza el cumplimiento de los requisitos de privacidad. Durante la pandemia de COVID-19, Lio se ajustó rápidamente para realizar funciones adicionales como desinfección y detección remota de temperatura corporal elevada. Cumple con la norma ISO13482: requisitos de seguridad para robots de cuidado personal, lo que significa que puede probarse e implementarse directamente en centros de atención.

En (Purwanto et al., 2017) El vehículo autónomo de interior (AIV) se utiliza para diversos fines, por lo que puede reducir la carga de trabajo humano. Este estudio tiene como objetivo desarrollar un sistema de navegación de corredor para AIV que utilice detección multipunto basada en visión. Se utilizan técnicas de detección multipunto en áreas de piso y pasillo para detectar áreas libres de obstáculos y estimar la dirección de AIV. La detección de área y la estimación de dirección se utilizan como información de entrada del algoritmo de navegación basado en un sistema de inferencia difuso para controlar el movimiento AIV. Los resultados de los experimentos muestran que AIV puede correr a lo largo del corredor con el mayor error medio de 6,49% con respecto a la línea central del corredor.

En (Mukherjee, 2020) se detalla un algoritmo de localización y detección de peatones basado en ROS que utiliza una cámara de visión estéreo ZED y Leddar M16, empleando darknet YOLOv2 para la localización, para producir resultados más rápidos y creíbles en la detección de objetos. Los datos de distancia se obtienen utilizando una nube de puntos de cámara estéreo y el Leddar M16, que luego se fusiona con ANFIS. YOLOv2 se entrenó con el

conjunto de datos Caltech Pedestrian en una GPU NVIDIA 940MX, con la interfaz del sensor realizada a través de ROS.

En (Farhadi, 2017) se presenta YOLO9000, un sistema de detección de objetos en tiempo real de última generación que puede detectar más de 9000 categorías de objetos. Primero, proponen varias mejoras al método de detección de YOLO, tanto novedosas como extraídas de trabajos anteriores. El modelo mejorado, YOLOv2, es lo último en tareas de detección estándar como PASCAL VOC y COCO. Usando un método de entrenamiento novedoso de múltiples escalas, el mismo modelo YOLOv2 puede ejecutarse en diferentes tamaños, lo que ofrece un equilibrio fácil entre velocidad y precisión. A 67 FPS, YOLOv2 obtiene 76,8 mAP en VOC 2007. A 40 FPS, YOLOv2 obtiene 78,6 mAP, superando a los métodos de última generación como Faster RCNN con ResNet y SSD mientras sigue funcionando significativamente más rápido. Finalmente proponen un método para entrenar conjuntamente en detección y clasificación de objetos. Con este método, entrenan a YOLO9000 simultáneamente en el conjunto de datos de detección de COCO y el conjunto de datos de clasificación de ImageNet. El entrenamiento conjunto permite que YOLO9000 prediga detecciones para clases de objetos que no tienen datos de detección etiquetados. Validaron este enfoque en la tarea de detección de ImageNet. YOLO9000 obtiene 19,7 mAP en el conjunto de validación de detección de ImageNet a pesar de tener solo datos de detección para 44 de las 200 clases. En las 156 clases que no están en COCO, YOLO9000 obtiene 16,0 mAP. YOLO9000 predice detecciones para más de 9000 categorías de objetos diferentes, todo en tiempo real.

En este trabajo se pretende probar la estrategia de segmentación semántica para el reconocimiento de los objetos, y probar una nueva técnica para estimar la distancia a la que se encuentran los objetos del robot.

Capítulo 3. Metodología aplicada.

El desarrollo de este proyecto se divide en dos etapas, la primera abarca la preparación del entorno de trabajo, que consiste en preparar las herramientas de trabajo: instalación del sistema operativo, controladores, librerías y software requerido como se muestra en el **Apéndice 1**. La segunda etapa consiste en la propuesta de sistema para el reconocimiento y clasificación de objetos, y el cálculo de la distancia relativa de estos.

3.1 Sistema propuesto

En este proyecto se propone un sistema de reconocimiento y clasificación de objetos, así como la estimación de la posición relativa de estos, con respecto a una cámara montada sobre un robot móvil. En la Figura 11 se muestra el diagrama general del sistema propuesto. El sistema tiene dos etapas de operación; la primera etapa abarca el preprocesamiento de las imágenes para entrenar el modelo y el entrenamiento del mismo. Estas dos tareas se realizan *offline*, en una computadora de escritorio. La segunda etapa corresponde al reconocimiento y clasificación de los objetos, así como la estimación de la posición relativa de estos. Estas tareas se realizan *online*, en un sistema Jetson Nano.

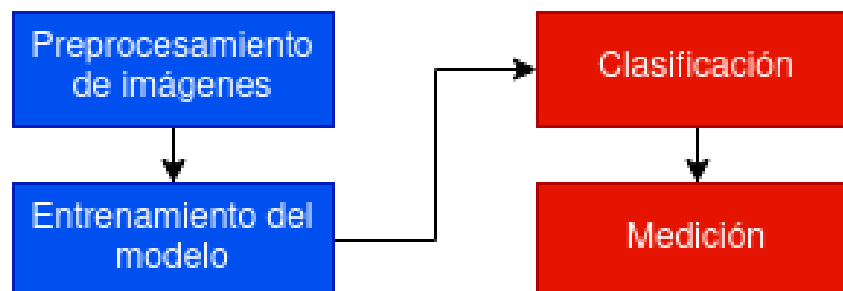


Figura 11: Diagrama general del funcionamiento del sistema.

3.2 Preparación de imágenes

La preparación de las imágenes para el entrenamiento del modelo, consistió en las siguientes tareas:

Se definió un conjunto de objetos de interés, a los cuales se les asignó un índice de clase y un color (ver Tabla 1). Estos objetos se encuentran dentro de un espacio real, utilizado como centro de trabajo. Se consideran como objetos de interés, aquellos a los cuales se pretende aproximar el robot en trabajos futuros. Las clases incluyen: conectores eléctricos, con la idea de que el robot se conecte de forma autónoma; celulares, platos y vasos para transportar; obstáculos comunes en ese entorno que el robot puede encontrar en su camino como herramienta, mesa, garrafón de agua, personas; otros obstáculos varios, poco comunes; y el resto de objetos como pared, piso, cortinas, etc., considerados como fondo.

Tabla 1: Definición de los objetos de interés.

Objeto	Color RGB	Índice
Fondo	(0, 0, 0)	0
Conector	(0, 128,128)	1
Celular	(128, 0, 0)	2
Plato	(0, 0, 128)	3
Vaso	(128, 128, 0)	4
Mesa	(0, 128, 0)	5
Personas	(192, 0, 0)	6
Herramienta	(128, 0, 128)	7
Garrafón de agua	(0,192,0)	8
Obstáculos	(0,0,192)	9

Se capturaron un total de 397 imágenes, que incluyen a los objetos de interés, a no más de 2.5 metros de distancia. Las imágenes se guardaron originalmente en formato JPG, con una

resolución de 640x480 píxeles. En la Tabla 2 se muestran las características básicas de la cámara utilizada. Para la captura de estas imágenes se configuró la cámara en 21fps, lo cual soporta un tamaño de las imágenes de hasta 8mp, sin embargo, al almacenar las imágenes se redujeron a un tamaño de 640x480 píxeles, el equivalente a un total de 307,200 píxeles.

Tabla 2: Especificaciones de la cámara.

Tipo de obturador	Rodante
Velocidad de fotogramas	21fps 8mp 30fps 1080p 120fps 720p
Ángulo de visión	75 grados (horizontal)

Se delimitó manualmente el contorno de los objetos de interés presentes en cada imagen, utilizando el software VIA VGG Image Anotator, Versión 2.0.11 1 (ver Figura 12), como se muestra en el **Apéndice 2**. Se generó un archivo JSON con la definición de los polígonos que representan el contorno correspondiente de cada objeto (ver Algoritmo 1). En la Figura 3 se muestra un ejemplo de una imagen segmentada y almacenada en formato PNG.

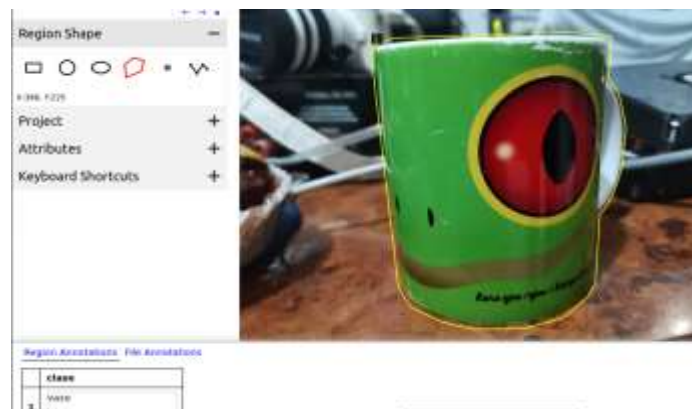


Figura 12: Delimitación del contorno de los objetos de interés.

Algoritmo 1: Generación de archivos PNG con imágenes segmentadas

```

entrada: Archivo JSON con los polígonos generados
salida : Archivos PNG con las imágenes segmentadas
for img in lista de imágenes en el archivo JSON do
    alto, ancho  $\leftarrow$  alto y ancho de img
    mascara  $\leftarrow$  genera nueva matriz tridimensional de (alto x ancho x 3)
    for poli in polígonos asociados a img en JSON do
        clase  $\leftarrow$  clase de objeto relacionado a poli
        (R, G, B)  $\leftarrow$  color predefinido para clase
        contorno  $\leftarrow$  crea matriz unidimensional vacía
        for (i, j) in lista de puntos de poli do
            agrega punto (i, j) a contorno
        end
        for i in rango(ancho) do
            for j in rango(alto) do
                // función pointPolygonTest
                if punto (i, j) se encuentra dentro del
                área de contorno then
                    mascara[j : i]  $\leftarrow$  (R, G, B) ;
                end
            end
        end
        almacena mascara como archivo PNG
    end
end
return

```

Mediante rutinas escritas en Python, a partir del archivo JSON, se generaron imágenes en formato PNG, con los objetos segmentados con su color correspondiente (ver Figura 13).



Figura 13: Segmentación de los objetos de interés.

Se normalizaron las componentes de color de las imágenes originales (JPG) y se almacenaron con formato Numpy. Además, se convirtieron las componentes de color RGB, de cada pixel de las imágenes segmentadas, a un valor único que corresponde a la clase del objeto al que pertenece el pixel, y se almacenó la matriz resultante en un archivo con formato Numpy (NPY) (ver Algoritmo 2).

Algoritmo 2: Generación de archivos Numpy

```
entrada: PATH de las carpetas de imágenes
salida : Archivos de imágenes con formato Numpy
PATH ORI ← PATH de las imágenes JPG
PATHS EG ← PATH de las imágenes PNG
for img ori in PATH ORI do
    |   img ori ← img ori/255,0
    |   almacena img ori con formato Numpy
end
COLOR ← [[0,0,0],[0,128,128],[128,0,0],[0,0,128],[128,128,0],[0,128,0],[192,0,0],[128,0,128],
[0,192,0], [0,0,192]]
etiqueta ← crea tensor de 2563 en ceros
for i, map in enumeración(COLOR) do
    |   etiqueta[(map[0]*256+map[1])*256+map[2]] ← i
end
for img seg in PATHS EG do
    |   indice ← ((img seg[:, :, 0] * 256 + img seg[:, :, 1]) * 256 + img seg[:, :, 2])
    |   img seg ← etiqueta[indice]
    |   // indice es una matriz bidimensional,
    |   // esta operación devuelve otra matriz
    |   // con la clase de objeto por pixel
    |   almacena img seg con formato Numpy
end
return
```

3.3 Entrenamiento del modelo de red neuronal

Con el objetivo de reconocer y clasificar los objetos en etapas posteriores, se implementó una estrategia de Segmentación Semántica, la cual consiste en clasificar las imágenes a nivel de pixel. Para esto, se diseñó y entrenó un modelo de red neuronal, que combina capas convolucionales y convolucionales transpuestas.

La arquitectura U-Net, consta de dos bloques principales, el primer bloque, llamado *downsampling* o camino de contracción, funciona como extractor de características de las imágenes de entrada y disminuye la resolución de salida. Este bloque aporta a la red la capacidad de clasificación de los objetos. El segundo bloque, nombrado como *upsampling* o camino de expansión, es aproximadamente simétrico al *downsampling*, resultando la arquitectura con forma de U. Este bloque, tiene como función propagar la información de las características extraídas, a capas de más alta resolución y aporta la capacidad de localización de los objetos clasificados. Cuando la resolución de salida corresponde a las dimensiones de las imágenes de entrada, es posible clasificar los objetos a nivel de pixel (Ronneberger et al., 2015).

En este proyecto, se utiliza el modelo ResNet18 (He et al., 2016) como bloque *downsampling*, al cual se le eliminaron las dos últimas capas, dado que estas se utilizan cuando el objetivo es clasificar a nivel de la imagen completa. Para realizar la función del bloque *upsampling*, al igual como se propone en (Ronneberger et al., 2015), se agregaron cuatro capas convolucionales transpuestas, combinadas con capas convolucionales comunes (ver Tabla 3).

Además de esta configuración, se evaluaron diferentes combinaciones, buscando un equilibrio entre precisión y el tiempo de inferencia sobre la Jetson. Un incremento del número de capas o de la cantidad de canales por capa, mejora la calidad del reconocimiento, pero incrementa el tiempo de inferencia y viceversa. Con la configuración elegida, se logra un 90% de precisión en el reconocimiento, con un tiempo de inferencia aproximado de 0.006 segundos. Por ejemplo, reducir a la mitad el número de capas de ambos bloques (Down y upsampling), el tiempo de inferencia se reduce a un segundo, pero con resultados de reconocimiento por debajo del 80 %.

Tabla 3: Arquitectura de la red neuronal utilizada.

Capas	Canales	Dimensión	K	Stride	Padding
Imagen	3	320x240			
Conv	64	160x120	7x7	2x2	3x3
Conv	64	160x120	3x3	1x1	1x1
Conv	64	160x120	3x3	1x1	1x1
Conv	64	160x120	3x3	1x1	1x1
Conv	64	160x120	3x3	1x1	1x1
Conv	128	80x60	3x3	2x2	1x1
Conv	128	80x60	3x3	1x1	1x1
Conv	128	80x60	3x3	1x1	1x1
Conv	128	80x60	3x3	1x1	1x1
Conv	256	40x30	3x3	2x2	1x1
Conv	256	40x30	3x3	1x1	1x1
Conv	256	40x30	3x3	1x1	1x1
Conv	256	40x30	3x3	1x1	1x1
Conv	512	20x15	3x3	2x2	1x1
Conv	512	20x15	3x3	1x1	1x1
Conv	512	20x15	3x3	1x1	1x1
Conv	512	20x15	3x3	1x1	1x1
Conv Trans	256	40x30	2x2	1x1	2x2
Conv	256	40x30	3x3	1x1	1x1
Conv Trans	128	80x60	2x2	1x1	2x2
Conv	128	80x60	3x3	1x1	1x1
Conv Trans	64	160x120	2x2	1x1	2x2
Conv	64	160x120	3x3	1x1	1x1
Conv Trans	64	320x240	2x2	1x1	2x2
Conv	10	320x240	3x3	1x1	1x1

En la Tabla 4 se muestran los parámetros del entrenamiento. Al igual que en el proceso para obtener la configuración de la arquitectura de la red, se evaluaron diferentes combinaciones de los parámetros, buscando disminuir el valor de la función de pérdida sobre los datos de entrenamiento y, principalmente, sobre los datos de prueba, sin incrementar de forma significativa el número de épocas requerido para obtener estos valores. Por ejemplo, aumentar la tasa de aprendizaje o el factor de decaimiento, mejora la velocidad del entrenamiento (menor número de épocas), pero incrementa la función de pérdida. Incrementar el número de épocas, con una configuración fija de los demás parámetros, reduce la función de pérdida sobre los datos de entrenamiento, pero sobre los datos de prueba, la función llega a un punto de inflexión, donde a partir de ese punto, se incrementa con cada época, esto es, el modelo se sobreentrena.

Tabla 4: Parámetros del entrenamiento.

Tamaño de lote	10
Número de épocas	100
Tasa de aprendizaje	0.0001
Factor de decaimiento	0.001
Función de pérdida	Entropía Cruzada
Función de optimización	Adam

El entrenamiento de la red se implementó en Python versión 3.9.7, utilizando las librerías Pytorch versión 1.9.0 y Torchvision versión 0.10.0. Se utilizaron 350 imágenes para entrenamiento y 47 para pruebas. En la Tabla 5 se muestran las características del equipo de cómputo donde se ejecutó el entrenamiento. La carga de los conjuntos de datos y el algoritmo de entrenamiento, es similar a como se explica en (Sensio, 2020)

Tabla 5: Características del equipo de cómputo para el entrenamiento.

Procesador	Intel Core i7-4790 3.60GHz x 8
Tarjeta gráfica	NVIDIA GeForce GTX TITAN X
Memoria RAM	16 GB
Disco duro	SSD 240 GB
Sistema operativo	Ubuntu 20.04 LTS

En la Figura 14 se muestra la estadística del entrenamiento. El valor final de la función de pérdida, a nivel de pixel, sobre los datos de entrenamiento, es de 0.00641 (color azul) Esta métrica está asociada con la cantidad de píxeles mal clasificados por la red. Para esto, el sistema compara la segmentación inferida por la red y la segmentación real que se introduce como dato. Un valor cero indica que no hay píxeles mal clasificados y el factor IoU de 0.87202 (color naranja). Esta métrica corresponde a la relación entre el área de píxeles bien clasificados (coincidencias entre la segmentación real y la inferida), y el área total que abarcan ambas segmentaciones. Un valor uno indica una coincidencia exacta entre ambas segmentaciones. Sobre los datos de prueba, la función de pérdida concluyó con un valor de 0.07024 (color verde) y el factor IoU de 0.76395 (color rojo). Los datos de prueba se utilizan durante el entrenamiento, únicamente para ver el comportamiento del modelo. Es decir, el conocimiento del error sobre estos datos, no se aplica en la mejora del modelo, como sí sucede con el error sobre los datos de entrenamiento. Por lo tanto, esta información es un parámetro más real de cómo se comportará el modelo con nuevos datos.

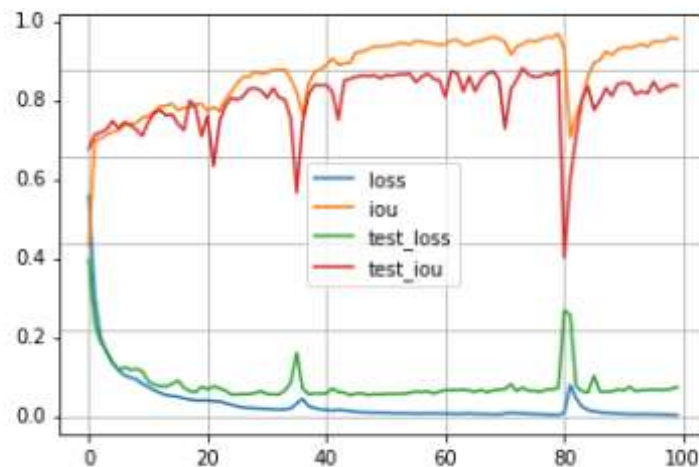


Figura 14: Estadística del entrenamiento.

3.4 Reconocimiento y clasificación

El proceso de reconocimiento y clasificación de los objetos se implementó en lenguaje Python, sobre un sistema Jetson Nano, equipado con un módulo de cámara. Para esto, se almacenó previamente el modelo de la red entrenada en formato Pytorch (.pt), y se cargó sobre la Jetson Nano.

El proceso inicia con la captura de una imagen a través de la cámara, y se pasa a la red para su evaluación. La red devuelve una matriz tridimensional con la predicción de la clase a la que pertenece cada pixel (ver Algoritmo 3).

Algoritmo 3: Reconocimiento de objetos sobre la Jetson Nano

```
entrada: Modelo de la red neuronal entrenada
salida : imágenes con segmentación inferida
carga del modelo de la red neuronal
activa el modelo en modo predicción
configura y activa la cámara
repeat
    imagen ← frame del buffer de la cámara
    imagen ← imagen en formato Torch
    imagen ← imagen con transferencia de la tercera dimensión hacia la primera dimensión
    // Se transfieren los canales de color
    // como primera dimensión de la imagen
    imagen ← imagen con nueva dimensión insertada como primera dimensión
    // Se agrega una dimensión dado que la red procesa por lote de imágenes
    imagen ← imagen.float()/255,0
    salida ← predicción de la imagen con el modelo
    salida ← primera imagen de la dimensión cero de salida
    // Se toma la primera (única) imagen del lote devuelto por la red
    salida ← argumento máximo de la primera
    dimensión de salida
    // Se reduce de tensor tridimensional a bidimensional con la clase inferida en cada pixel
    despliega o almacena salida con formato PNG
until pulsa tecla < ESC >;
return
```

Las dimensiones del tensor devuelto por la red son $(1 \times 10 \times 240 \times 320)$. La dimensión con subíndice cero corresponde al número de imágenes del lote procesado (se procesa una a la vez), la dimensión uno corresponde a la cantidad de clases por clasificar, las últimas dos dimensiones, corresponden al alto y ancho de la imagen. El valor de cada celda indica la probabilidad estimada por la red, con valor entre 0 y 1, de que el pixel que se encuentra en la fila y columna a la que pertenece la celda, sea de la clase con valor igual al subíndice de la dimensión uno de la celda. A este tensor se le elimina la dimensión cero, quedando ahora el número de clases como nueva dimensión cero.

Por ejemplo, en la Figura 15 se muestra un tensor con la predicción para tres clases (imagen superior). Se presentan únicamente los valores para la última columna (columna 319 de la dimensión dos). En color amarillo se muestran las celdas con el valor de probabilidad de que el pixel en la fila y columna correspondiente sea de la clase cero. En color verde, las celdas con la probabilidad de que el pixel sea de la clase uno, y en color naranja las celdas con la probabilidad de que el pixel sea de la clase dos.

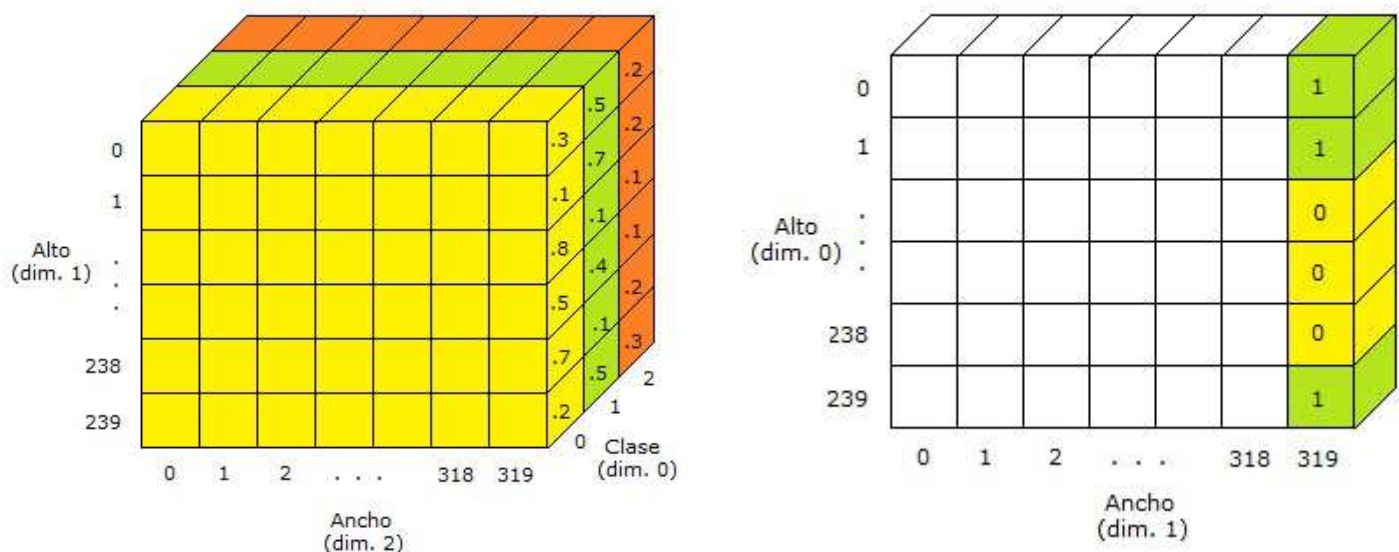


Figura 15: Ejemplo de tensor de probabilidades para tres clases (superior) y del tensor con predicción final de clases (inferior).

Para obtener la clase de cada pixel, se aplica la función `torch.argmax(tensor prob,axis=0)`, sobre el tensor tridimensional, resultando un tensor bidimensional, donde el valor de cada celda corresponde a la clase inferida del pixel (1). En la imagen anterior en la Figura 14, se muestran las clases asignadas para la última columna, a partir del tensor de probabilidades de la imagen superior. Por ejemplo, el pixel de la fila cero y columna 319, tiene una probabilidad de 0.30 de pertenecer a la clase cero, 0.50 a la clase uno y 0.20 a la clase dos. Por lo que el valor asignado (clase con máxima probabilidad) es la clase uno.

En la Figura 16 se muestra un ejemplo de una imagen utilizada como entrada a la red. Y en la Figura 17 se muestra la imagen segmentada a partir de la inferencia realizada por la red.

$$\text{clase pixel}(i,j) = k, \text{ tal que } \text{tensor}(k,i,j) = \max\{\text{tensor}(m,i,j) \mid 0 \leq m < \text{len}(\text{clases})\} \quad (1)$$



Figura 16: Imagen original de entrada a la red.

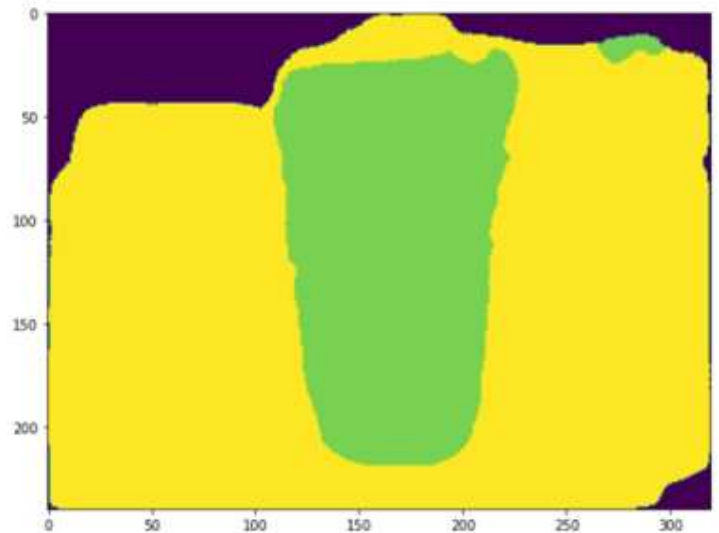


Figura 17: Segmentación inferida por la red neuronal.

3.5 Estimación de la posición relativa

Una vez identificados y segmentados los objetos en la imagen, el siguiente paso es estimar su posición relativa con respecto al robot. Para esto, se aplicó una técnica de visión estereoscópica, para calcular la distancia a la que se encuentran los objetos del robot, a partir de dos imágenes tomadas en posiciones diferentes de la cámara. Esta estrategia, asume que, a mayor distancia de los objetos con respecto a la cámara, menor será su desplazamiento, al comparar su posición en ambas imágenes. Y de forma correspondiente, a menor distancia de los objetos, mayor será su desplazamiento.

Para obtener la relación, entre el desplazamiento de los objetos en las imágenes y la distancia a la que se encuentran de la cámara, se capturaron ocho imágenes de tres puntos de control, posicionando la cámara a 50, 80, 100 y 150 cm de estos. Para cada una de estas distancias, se capturaron dos imágenes, colocando la cámara a 6 cm de distancia entre una y otra toma (ver Figura 18).

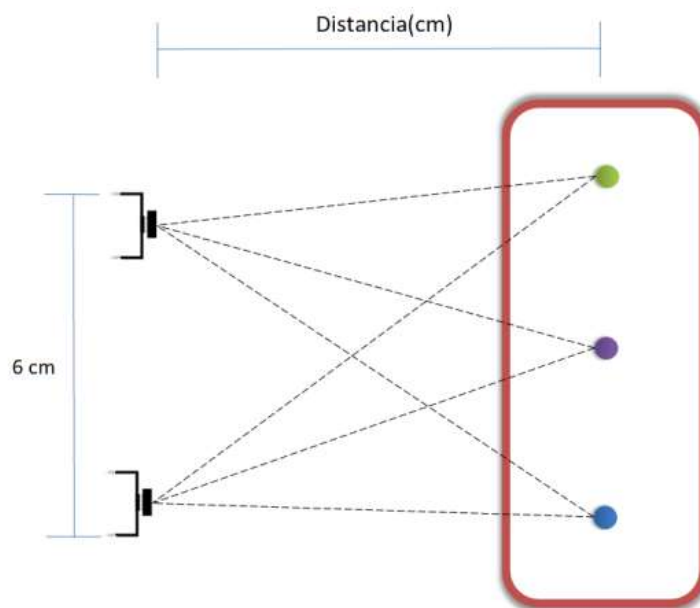


Figura 18: Captura de puntos de control desde dos posiciones distintas.

En la Tabla 6 se muestran las posiciones de los puntos de control en cada una de las imágenes capturadas. Las posiciones están dadas en píxeles sobre el eje X. Así mismo, se muestra el desplazamiento de los puntos entre las dos imágenes. Como desplazamiento asociado a cada una de estas distancias, se tomó el promedio de los tres puntos de control.

Tabla 6: Desplazamiento de los puntos de control.

		Posición en el eje X (píxeles)					
		Control uno		Control dos		Control tres	
		Pos.	Desp	Pos.	Desp	Pos.	Desp
50 cm	Origen	320	73	197	74	76	72
	6 cm	247		123		4	
80 cm	Origen	320	47	239	48	160	48
	6 cm	273		191		112	
110 cm	Origen	320	38	254	39	190	39
	6 cm	282		215		151	
150 cm	Origen	320	26	276	27	232	26
	6 cm	294		294		206	

Una vez obtenidos los desplazamientos asociados a estas cuatro distancias, se aplicó análisis de regresión con la librería `scipy.optimize` de Python, para obtener la función que mejor describa esta relación. El método `curve fit` aplica el algoritmo Levenberg-Marquardt (Levenberg, 1944), (Marquardt, 1963), para resolver el problema de ajuste de curvas por mínimos cuadrados (Scipy, 2022). Esto es, dado un conjunto de m pares (x_i, y_i) , aplica una combinación de los algoritmos Gauss-Newton y Descenso del Gradiente, para encontrar los parámetros β del modelo de la curva $f(x, \beta)$, que minimicen la suma de los cuadrados de las diferencias (2).

$$\operatorname{argmin}_{\beta} \sum_{i=1}^m [y_i - f(x_i, \beta)]^2 \quad (2)$$

Para esto, se probaron diferentes funciones (ver Algoritmo 4), lineal (ver Figura 19), exponencial (ver Figura 20), e hiperbólica (ver Figura 21). Finalmente, la función hiperbólica dada en (3), resultó ajustarse mejor a los datos. Esta función hiperbólica, de la forma $y = k/x$, asume como variable independiente x , el desplazamiento de los objetos entre ambas imágenes, dado en pixeles; y calcula, como variable dependiente y , la distancia a la que se encuentran los objetos de la cámara, dada en centímetros.

$$\text{distancia(cm)} = 3836,15/\text{desplazamiento(px)} \quad (3)$$

Algoritmo 4: Análisis de Regresión

```
entrada: Distancias de los puntos de control y desplazamientos asociados
salida : Ecuaciones de las curvas que mejor se ajustan
define func _lin(x,m, b):
    return m · x + b
define func exp(x, a, b):
    return a · eb·x
define func hip(x, k):
    return k/x
x ← arreglo de distancias
y ← arreglo de desplazamientos
linOpt ← curve_fit( func _lin, x, y)
m, b ← paramOpt
genera y muestra gráfica lineal
expOpt ← curve_fit( func exp, x, y)
a, b ← paramOpt
genera y muestra gráfica exponencial
hipOpt ← curve_fit( func hip, x, y)
k ← paramOpt
genera y muestra gráfica hiperbólica
return
```

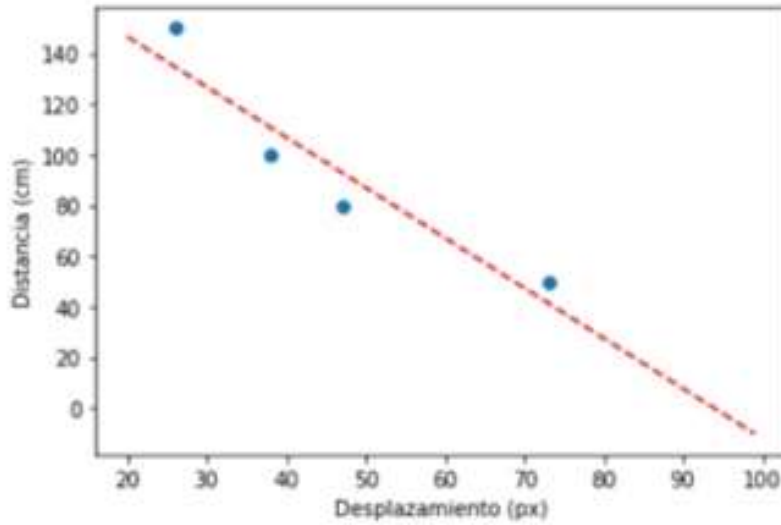


Figura 19: Regresión lineal.

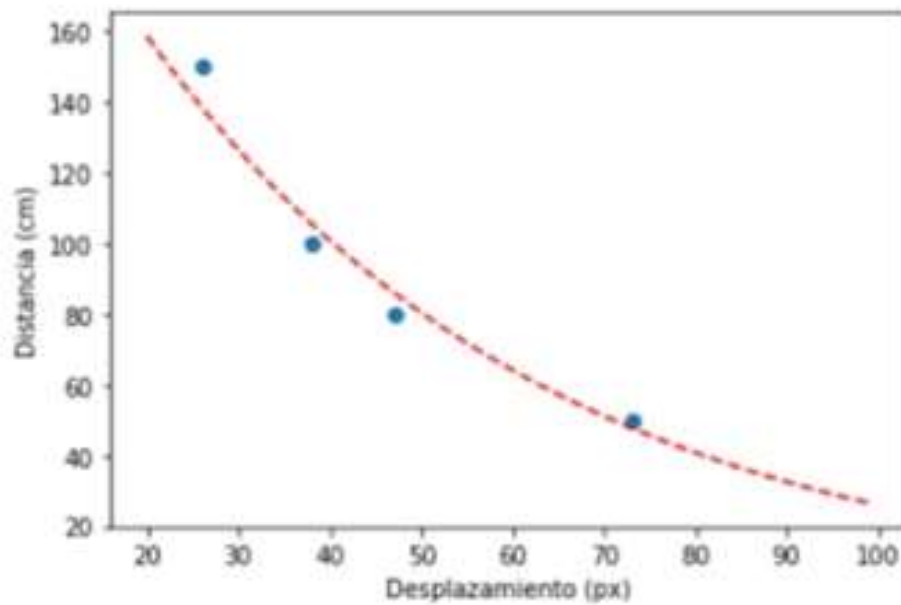


Figura 20: Regresión exponencial.

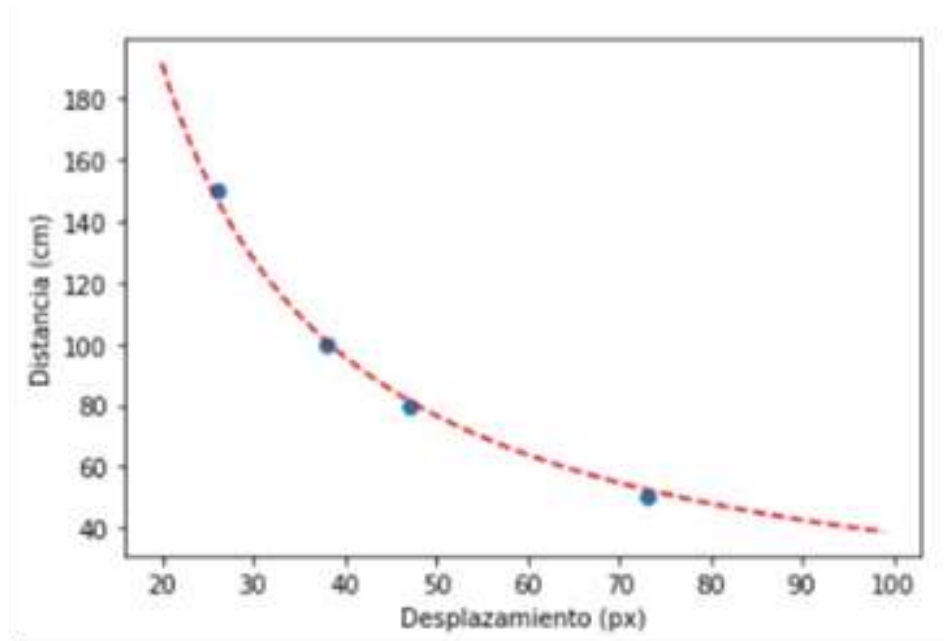


Figura 21: Regresión hiperbólica.

Con la función (3), se puede estimar la distancia relativa de los objetos con respecto al robot, a partir del desplazamiento de estos en las imágenes tomadas desde dos puntos diferentes.

Para esto, una vez segmentadas las imágenes, se obtiene el contorno y el centroide de los objetos en ambas imágenes, utilizando la librería OpenCV de Python (ver Algoritmo 5). En la Figura 34 se muestra un ejemplo de la generación del contorno y centroide para un mismo objeto, en las dos imágenes capturadas.

El valor del desplazamiento se obtiene con el valor absoluto de la diferencia de la componente X de ambos centroides. En el ejemplo de la Figura 33, la componente X del centroide del objeto en la imagen superior es 227, y en la imagen inferior es de 176, resultando un desplazamiento de 51 píxeles. Aplicando la función (3), la distancia inferida, entre el robot y el objeto, es de 71 cm, lo cual es cercano a la distancia real, que es de 70 cm.

Algoritmo 5: Cálculo de Distancia Relativa

entrada: Modelo de la red neuronal entrenada, clase del objeto de interés
salida : Distancia relativa estimada del objeto de interés
carga del modelo de la red neuronal
activa el modelo en modo predicción
 configura y activa la cámara uno
 configura y activa la cámara dos
repeat
 imagen1 \leftarrow frame del buffer de la cámara uno
 imagen2 \leftarrow frame del buffer de la cámara dos
 salida1 \leftarrow Proceso de reconocimiento con *imagen1*
 // reconocimiento con Algoritmo 3
 salida2 \leftarrow Proceso de reconocimiento con *imagen2*
 salida1 \leftarrow filtrado de pixeles de la clase de interés de *salida1*
 salida2 \leftarrow filtrado de pixeles de la clase de interés de *salida2*
 contorno1 \leftarrow identificación de contorno en *salida1*
 // Function *f* indContours de OpenCV
 contorno2 \leftarrow identificación de contorno en *salida2*
 M 1 \leftarrow cálculo
 M 2 \leftarrow cálculo de momentos del contorno2
 centr 1x \leftarrow M 11,0/M 10,0
 // Función moments de OpenCV
 centr 2x \leftarrow M 21,0/M 20,0
 desplazamiento \leftarrow abs(*centr 2x* – *centr 1x*)
 distancia \leftarrow 3836,15/*desplazamiento*
 despliega *distancia*
until pulsa tecla < ESC >;
return

Un problema detectado al aplicar esta estrategia de estimación de distancias, es que si en una o en ambas imágenes, el objeto de interés se encuentra en un extremo, parcialmente fuera del campo de visión de la cámara, no es probable que el centroide corresponda al mismo punto de referencia sobre el objeto en ambas imágenes. Es decir, no se estaría midiendo el desplazamiento del mismo punto (o puntos cercanos), por lo tanto, no sería un indicador confiable de cuanto se desplaza el objeto. Por ejemplo, en la Figura 22, tanto en la imagen de la izquierda como la derecha, se obtiene el centroide aproximadamente en el centro de la taza completa. En cambio, el centroide de la taza de la Figura 22, se encuentra cargado al lado derecho de la taza (si esta se viera completa), por lo que el desplazamiento de este centroide,

con respecto a cualquiera de los dos centroides de la Figura 23, no representaría una buena aproximación del desplazamiento de la taza.

La propuesta para estos casos, es utilizar las coordenadas del borde del objeto, que se encuentra del lado contrario al extremo por donde sale parcialmente de la imagen. Esto es, si el objeto sale parcialmente por el lado izquierdo de la imagen, se toma la componente X máxima (contorno derecho) del objeto (circulo amarillo en la Figura 22). Si el objeto sale parcialmente por el extremo derecho, se toma la componente X mínima. Los resultados de aplicar esta estrategia, se presentan en la siguiente sección.

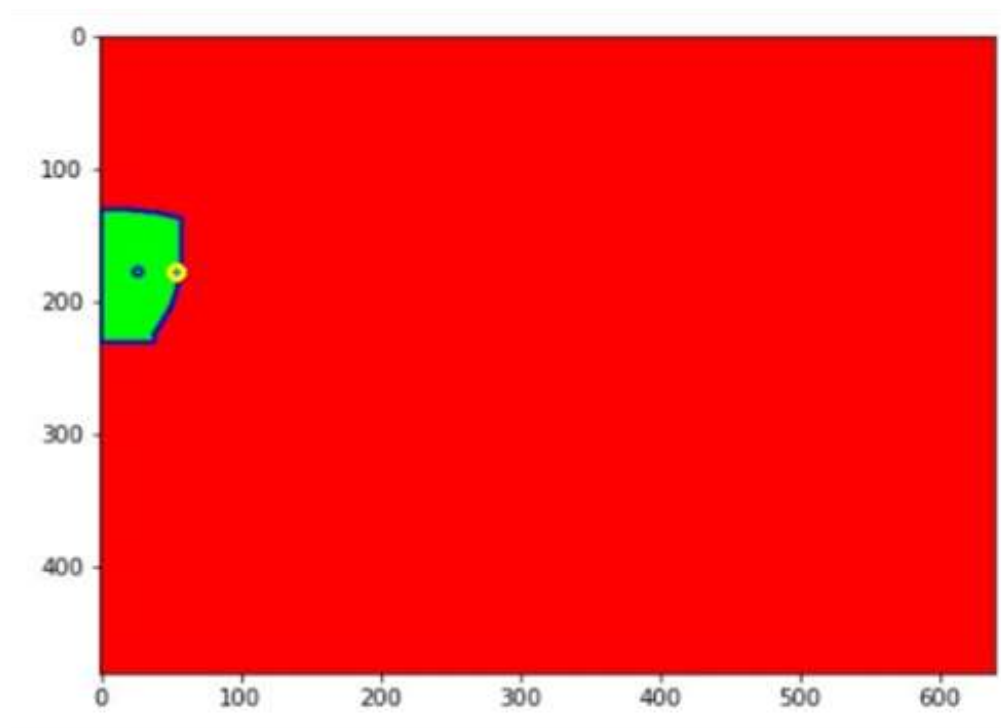


Figura 22: Objeto parcialmente fuera del campo de visión.

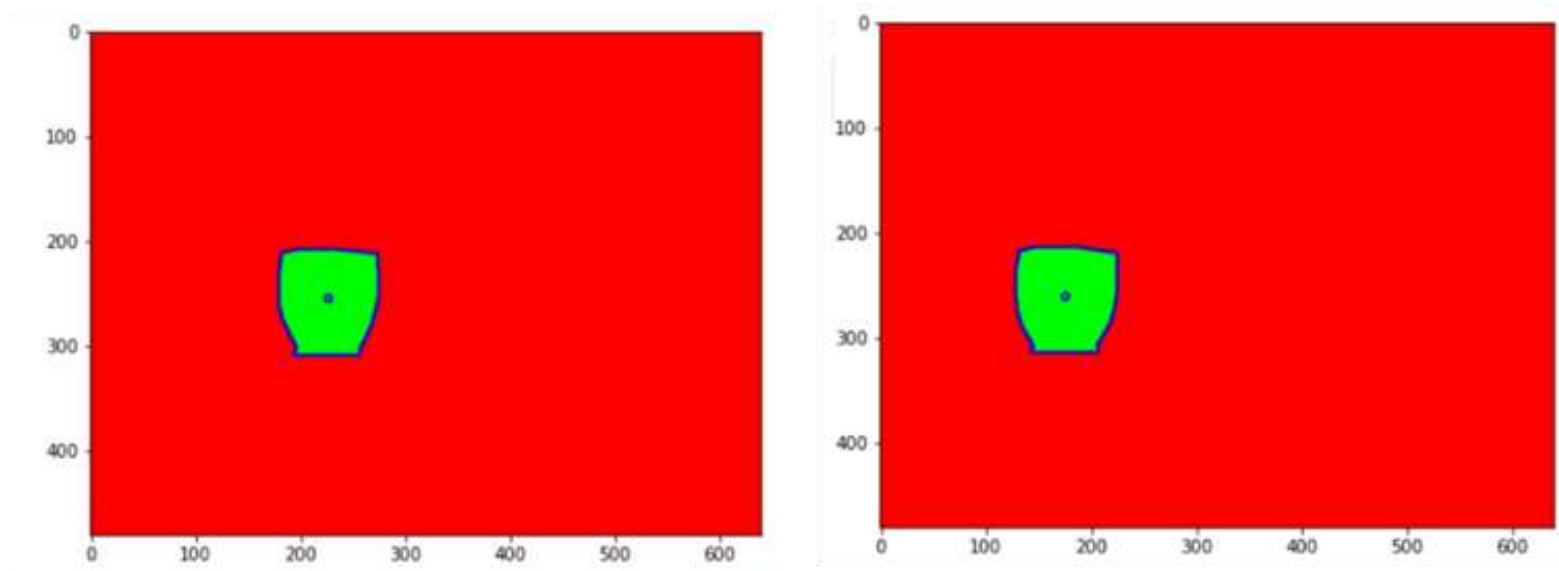


Figura 23: Ejemplo de la generación de contornos y obtención de centroides.

Capítulo 4. Resultados y Conclusiones

4.1 Pruebas de reconocimiento y clasificación

Las pruebas consistieron en aplicar el sistema de reconocimiento y clasificación, sobre 47 imágenes que contienen un total de 127 ocurrencias de los objetos de interés. En la Figura 24 se muestran ejemplos de las pruebas realizadas. En la primera columna se muestran las imágenes originales, en la segunda columna se muestran las imágenes con la segmentación definida de forma manual (considerada como real), y en la tercera columna se muestran las imágenes con la segmentación inferida por la red.

Se puede observar que la segmentación inferida es similar a la segmentación real. Por ejemplo, en la Figura 24-A, en ambas imágenes segmentadas, se muestran en color amarillo los píxeles que se etiquetaron como parte de la mesa, o como equipo electrónico sobre esta. Se etiquetaron los píxeles del celular en color azul turquesa, y en color violeta los píxeles del fondo, que incluye piso, pared, cortinas y otros objetos no considerados de interés. Los colores asignados durante las pruebas a las diferentes clases de objetos, no corresponden

necesariamente con el color asignado en la preparación de las muestras, sin embargo, el valor de la etiqueta asignada sí corresponde.

En algunos casos se detectaron falsos positivos, por ejemplo, en la Figura 24-A se etiquetó el brillo sobre el borde del celular, como parte de un plato. Un caso similar ocurrió en el reconocimiento del tomacorriente de la Figura 24-E. Se detectaron, también, casos de falsos negativos, por ejemplo, en la Figura 24-C no se reconoció una de las patas de la mesa. Para mejorar el reconocimiento, y disminuir estos casos de errores, se propone reentrenar la red con una mayor cantidad de imágenes de muestra, considerando mayor variación en la orientación de las tomas, diferentes condiciones de brillo, diferentes distancias, y generar la segmentación manual de forma más exacta.

En la Tabla 7 se muestra la estadística de los resultados de las pruebas. El número de objetos reconocidos y clasificados con la clase correcta es de 116 (positivos correctos), la cantidad de objetos identificados con una clase no correcta es de 12 (falsos positivos), y el número de objetos presentes en las imágenes, no identificados por el sistema, es de 11 (falsos negativos). En esta estadística no se incluye el fondo como objeto de interés.

Tabla 7: Pruebas de reconocimiento y clasificación.

Positivos correctos	116
Falsos positivos	12
Falsos negativos	11
Precisión	90.60%
Sensibilidad	91.30%



Figura 24: Ejemplos de pruebas de reconocimiento y clasificación.

Para un análisis de la predicción al nivel de clases, en la Figura 25, se muestra el mapa de calor de los resultados de la clasificación. En este caso, si se incluye el fondo como un objeto más, para mostrar los casos en que un objeto de interés se reconoció como fondo y viceversa.

Dado que en las imágenes de prueba resultó un bajo número de garrafrones y de obstáculos, estos no se incluyen en el mapa de calor.

Se observa que uno de los errores más persistentes de la red, es en el reconocimiento de los conectores, los cuales confunde como parte de una mesa. Sin embargo, similar al caso de la Figura 24-E, en los cuatro casos reportados en el mapa, la confusión es parcial, es decir, reconoce el conector, pero parte de este, lo confunde como parte de una mesa. Lo mismo sucede con el celular de Figura 23-A, identifica correctamente el celular, pero el borde con brillo de este, lo confunde como parte de un plato.

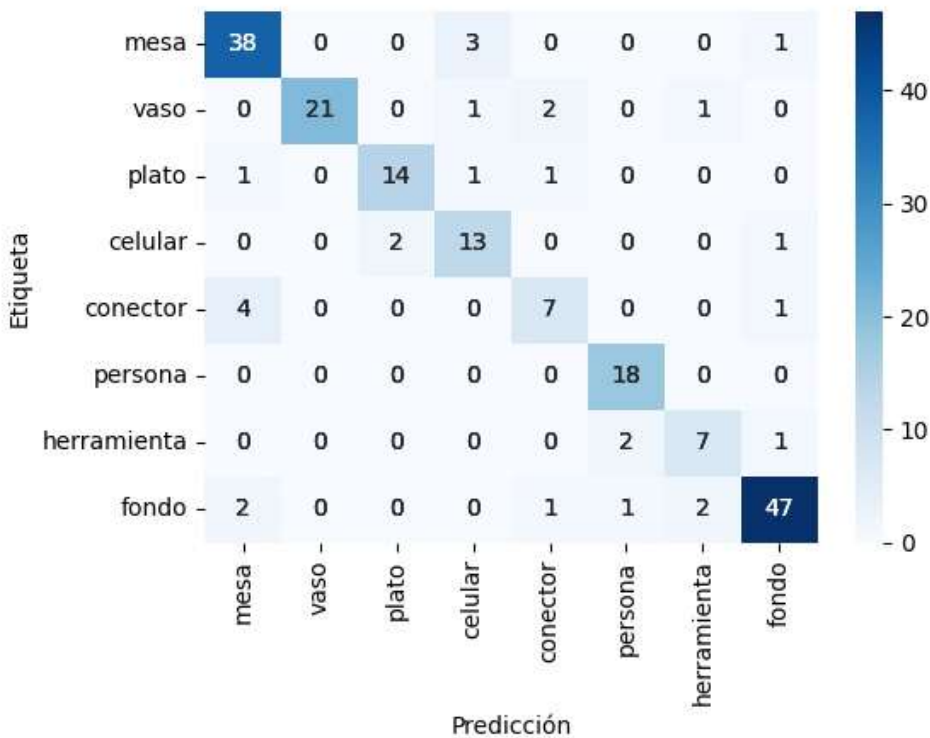


Figura 25: Mapa de calor de las pruebas de clasificación.

4.2 Pruebas de estimación de distancias relativas

Para estas pruebas, se colocaron objetos en diferentes posiciones con distancias conocidas, y se aplicó la estrategia propuesta. Por cada posición de los objetos, se capturaron dos imágenes con las cámaras separadas entre sí a una distancia de 6 cm. Una vez obtenidas las imágenes, se generó el contorno y centroide de los objetos de prueba, se calculó el desplazamiento de los centroides, se estimó la distancia de los objetos respecto a las cámaras y se obtuvo el error de la estimación.

Por ejemplo, en la Figura 26 se muestran las imágenes que corresponden a la prueba número uno reportada en la Tabla 8. Ambas imágenes (superior e inferior) se tomaron con la cámara a una distancia de 150 cm de la taza. Una vez segmentadas las imágenes, el centroide de la taza de la imagen superior resultó en la coordenada $x = 155$, y el centroide de la taza de la imagen inferior en $x = 130$, que corresponde a un desplazamiento del centroide de 25 cm. Aplicando la función (3) se obtiene una distancia estimada de 153 cm, y un error respecto a la distancia real de 3 cm.



Figura 26: imágenes de la prueba número uno de estimación de distancias

En la quinta columna de la Tabla 8, se muestra el error de estimación, al considerar únicamente los centroides para el cálculo del desplazamiento. Se puede observar que el error para las pruebas 8 y 10 es bastante elevado. Esto es debido a que en estos casos, el objeto se sale parcialmente de la imagen, por lo que los centroides calculados no corresponden al mismo punto de referencia del objeto en ambas imágenes. En la columna ocho se muestra el error para estas dos pruebas, con el cálculo del desplazamiento ajustado con base al borde izquierdo del objeto. De esta forma, el error final promedio en la estimación de las distancias, es de 4.6 cm, lo cual representa un 95,2% de precisión, con una raíz del error cuadrático medio (RMSE) de 6.1 cm (ver Tabla 9).

Tabla 8: Pruebas de estimación de la posición relativa de los objetos.

Prueba	Dist. Real	Desplaz. (centr.)	Dist. Estim.	Error (cm)	Desplaz. (borde)	Dist. Estim.	Error (cm)
1	150	25	153	3			
2	150	23	166	16			
3	120	31	123	3			
4	120	31	123	3			
5	100	36	106	6			
6	100	38	101	1			
7	70	51	75	5			
8	70	31	124	54	53	68	2
9	50	72	53	3			
10	30	69	56	25	111	26	4
media = 12.3					5.06		
desv. = 15.6					4.16		

Tabla 9: Resultados de las pruebas de estimación de distancias.

MSE	37,4
RMSE	6,1
Distancia real promedio	96 cm
Error promedio	4,6 cm
Desviación estándar	4,2 cm
Precisión de la estimación	95,2%

En la Figura 27 se muestra gráficamente la distancia de los objetos inferida por el sistema (componente Y de los puntos naranjas), a partir del desplazamiento de los centroides. Se

muestra también la distancia real de los objetos (componente Y de los puntos azules). La componente X de cualquier punto (azul o naranja), corresponde al desplazamiento.

En la Figura 28 se muestran las distancias, una vez que se ajustó el desplazamiento de los objetos que salen parcialmente de la imagen (pruebas 8 y 10). Para estos objetos, se utilizó el desplazamiento de sus bordes izquierdos, en lugar de sus centroides. Se puede observar que las distancias estimadas se ajustan mejor a las distancias reales.

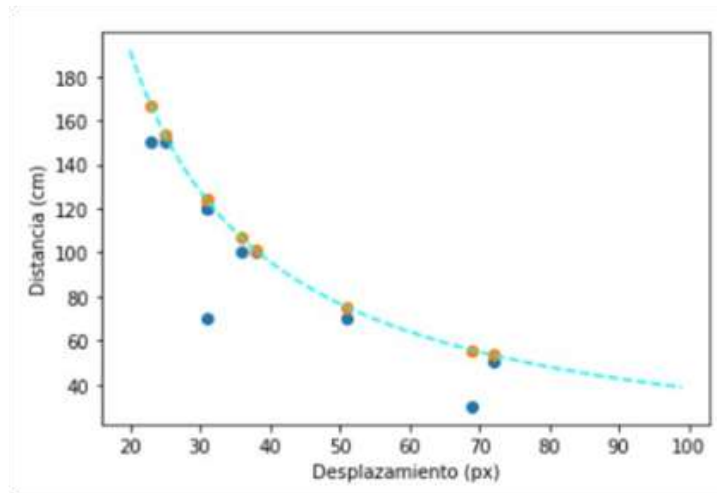


Figura 27: Estimación de las distancias con base a centroides.

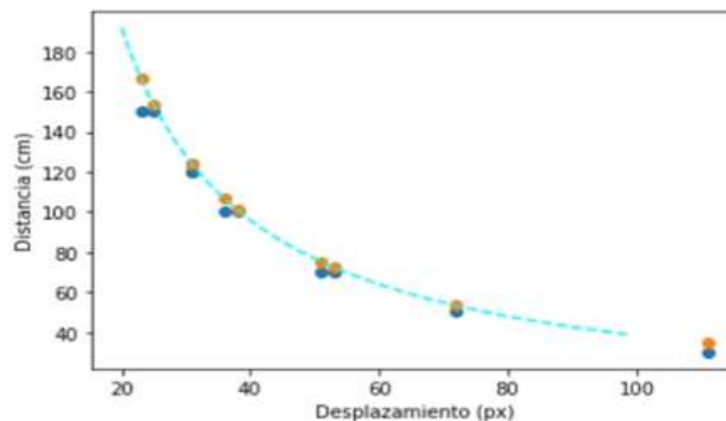


Figura 28: Estimación de las distancias con base a centroides y bordes.

4.3 Análisis e interpretación de resultados

Los resultados de los experimentos del método propuesto muestran un 90.6% de precisión en el reconocimiento. Esto es, de 127 ocurrencias de objetos de interés presentes en las

47 imágenes de prueba, 115 se identificaron de manera correcta y 12 con error. En cuanto al cálculo de la distancia relativa a la que se encuentran los objetos de la cámara, los experimentos muestran una tasa de error medio de 5 cm.

Como primer acercamiento en la atención de la problemática expuesta, consideramos que los resultados son satisfactorios. Sin embargo, existen varias oportunidades de mejora (como se muestra en la sección de Recomendaciones), tanto en el proceso de reconocimiento, como en la estimación de la distancia de los objetos. Por otro lado, por cuestión de tiempo y por un daño que sufrió la tarjeta Jetson Nano durante el desarrollo de los experimentos, queda pendiente la realización de pruebas del sistema sobre el robot en movimiento, lo cual se propone como trabajo futuro para un próximo proyecto de residencias.

4.4 Conclusiones

En este trabajo se propone un sistema para el reconocimiento y clasificación de objetos, basado en redes neuronales convolucionales, así como la estimación de la posición relativa de estos con respecto a una cámara montada sobre un robot móvil, aplicando una técnica de visión estereoscópica. El objetivo de contar con este sistema, es integrarlo, en una siguiente etapa del proyecto, sobre un robot de cuidados y asistencia básica.

Se ha logrado en buena medida los objetivos planteados de Diseñar e implementar algoritmos para la automatización del procesamiento de las imágenes mismos que fueron de ayuda en el pre-procesamiento de los conjuntos de datos de entrenamiento, se logró diseñar e implementar un sistema para el cálculo de la distancia relativa de los objetos a la cámara objetivos alcanzados el cual alcanzo una buena precisión además se logró implementar un sistema para el reconocimiento de objetos a través de redes neuronales artificiales lo cual se logró realizar completar satisfactoriamente.

Los experimentos realizados arrojaron resultados satisfactorios, sin embargo, consideramos que es posible mejorar la precisión del reconocimiento de los objetos, a través de un mayor y más variado conjunto de muestras de entrenamientos.

4.5 Recomendaciones

Con el objetivo de lograr una mejor precisión en el reconocimiento de los objetos, se recomienda lo siguiente:

- Realizar la segmentación manual de los objetos, utilizada para el entrenamiento de la red, de forma más precisa. Es decir, segmentar de forma más próxima al contorno real de los objetos. Se consideró que esto permitirá un mejor reconocimiento y clasificación de los objetos, cuando estos se encuentren en un entorno diferente.
- Incrementar la cantidad y variedad de las imágenes de entrenamiento y validación. Incluyendo un mayor número de imágenes de reflejos, diferentes horarios del día, etc.
- Probar con diferentes arquitecturas y parámetros de la red neuronal.

En cuanto al cálculo de la distancia relativa de los objetos, se recomienda:

- Realizar el experimento con las cámaras bien calibradas, o de preferencia con cámaras de visión estereoscópica con doble lente, para lograr mayor precisión en los cálculos.

Apéndice 1

Preparación del entorno de trabajo

Instalación y configuración de Ubuntu.

Inicia con la instalación de un disco sólido de 512 GB, en un equipo de cómputo que cuenta con una tarjeta gráfica Nvidia GeForce GTX TITAN X instalada (ver Figura 29)



Figura 29: Instalación del disco sólido.

Para la instalación de Ubuntu 20.04 se descarga la imagen del sistema operativo desde la página: <http://www.releases.ubuntu.com/20.04/>. En la Figura 30, se muestran las opciones de descarga. Se recomienda la opción “Desktop Image”.



Figura 30: Página para descargar el sistema operativo Ubuntu.

Posteriormente se ejecutan los siguientes pasos:

- Formatear una memoria USB para utilizarse como arranque del sistema.
- Se enciende la computadora con la USB puesta y se accede a la BIOS.
- Se selecciona la opción “iniciar con Boot Menu”
- Se selecciona la memoria que contiene la imagen del sistema operativo a instalar (ver Figura 31).

Reconocimiento y seguimiento de objetos en entornos controlados

Se selecciona el GRUB, es decir, que sistema operativo se va a instalar, en este caso Ubuntu.

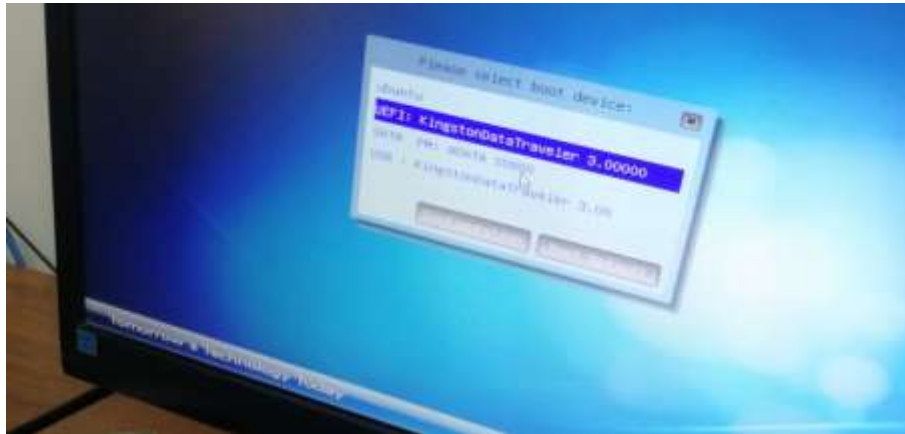


Figura 31: Menú de selección de arranque.

A continuación inicia la instalación, indicando que se están descomprimiendo los archivos para la instalación (ver Figura 32).



Figura 32: Descompresión de archivos necesarios para la instalación.

Cuando la instalación finaliza, muestra un menú donde se selecciona el idioma de instalación, es este caso español, y se elige si se desea probar o instalar Ubuntu. En este caso se indicó en la opción “Instalar Ubuntu” (ver Figura 33).

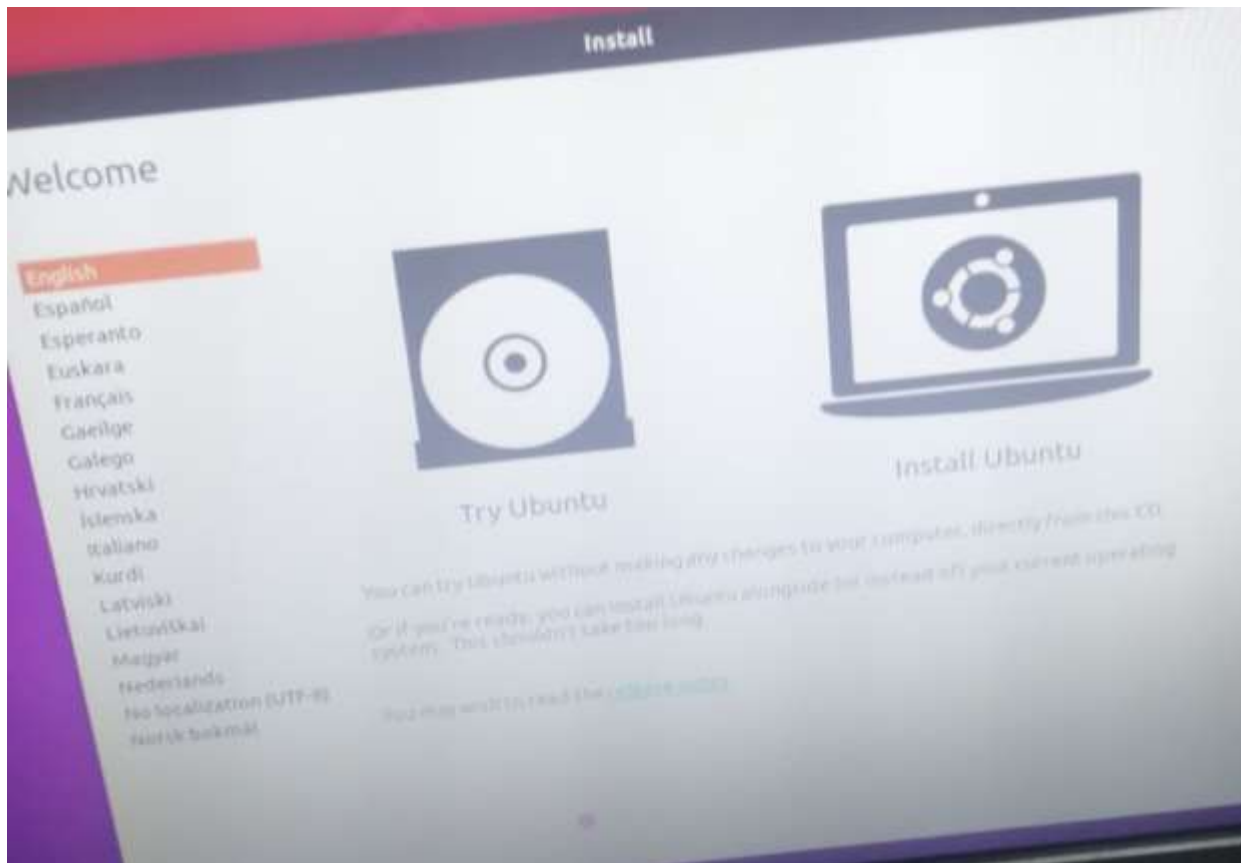


Figura 33: Menú de prueba o instalación de Ubuntu.

Reconocimiento y seguimiento de objetos en entornos controlados

El siguiente paso es seleccionar la configuración del teclado. En este caso la selección es “Español latinoamericano” (ver Figura 34).

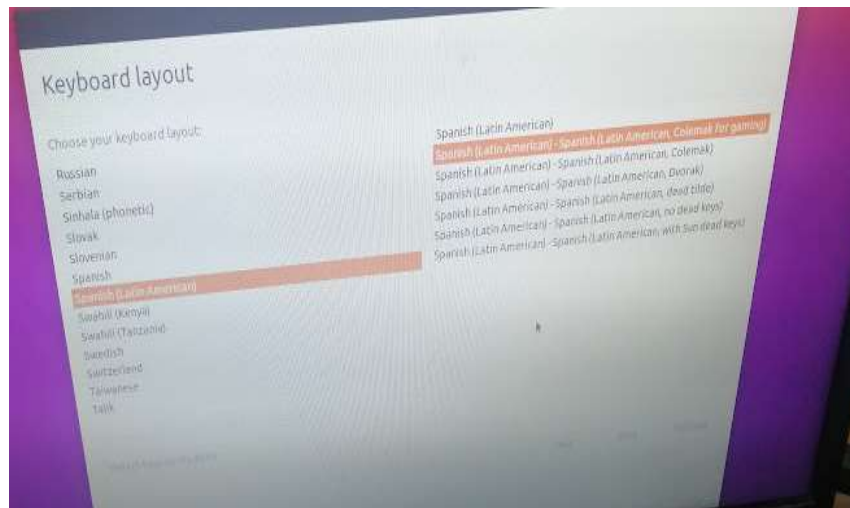


Figura 34: Menú de selección de teclado.

En el siguiente menú se selecciona el tipo de instalación, en este caso la selección es “instalación normal” (ver Figura 35).

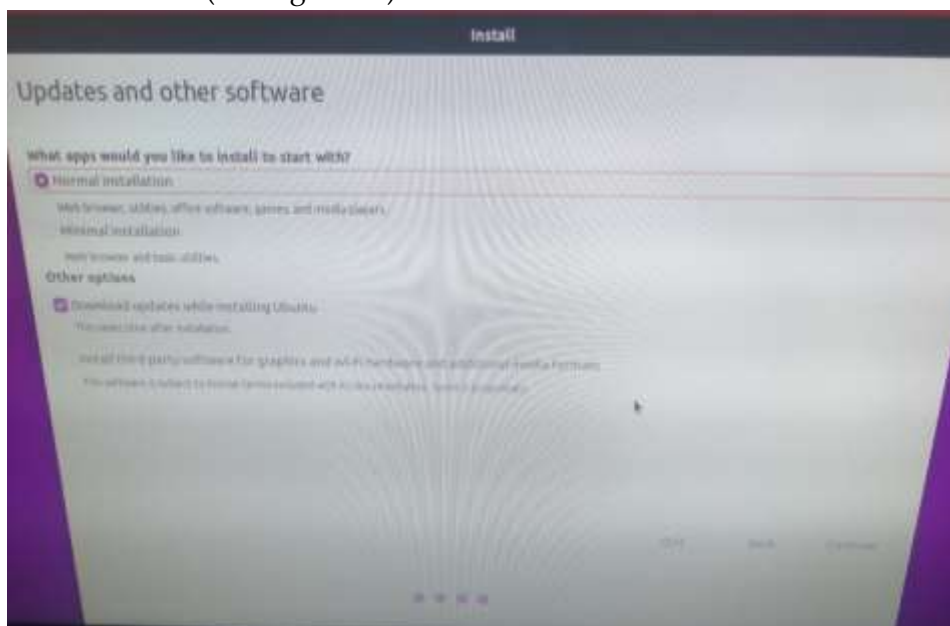


Figura 35: Menú de selección de tipo de instalación.

A continuación se indica la partición en el disco sólido, donde se aloja el sistema operativo y se guarda la información de configuración (ver Figura 36).

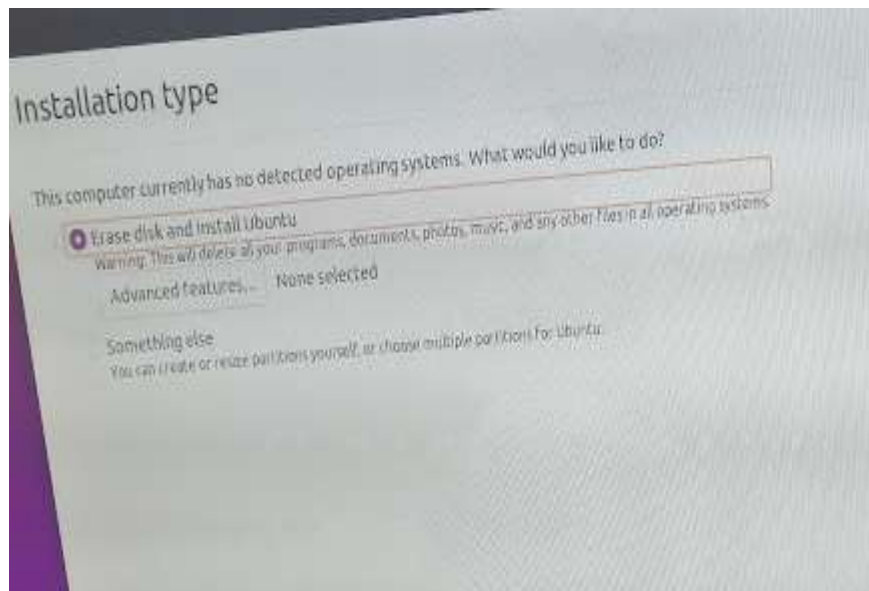


Figura 36: Menú de selección de disco para instalación.

Posteriormente, se establece la región para la zona horaria (ver Figura 37).

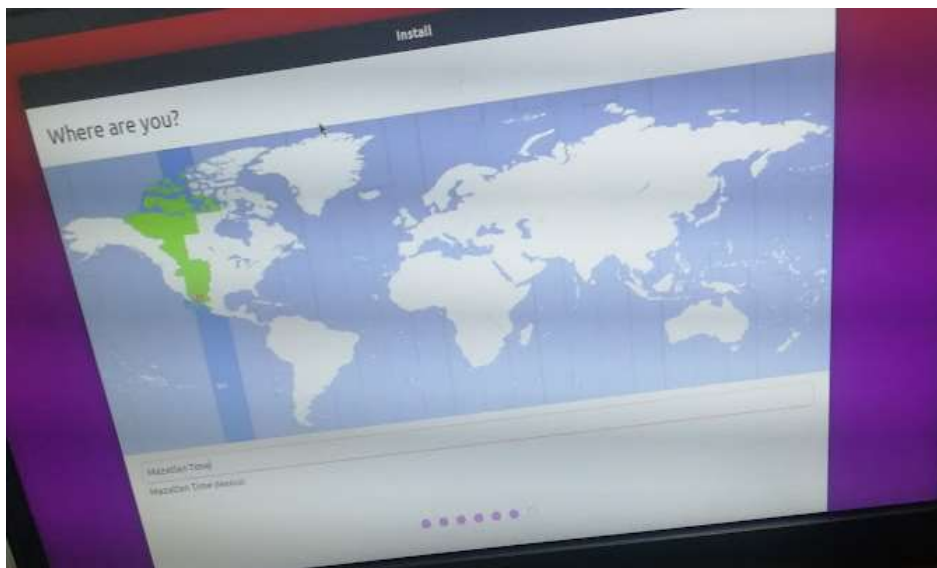


Figura 37: Menú de selección de zona horaria.

Reconocimiento y seguimiento de objetos en entornos controlados

Como último paso, se registra el nombre de usuario y contraseña para acceso al sistema (ver Figura 38).

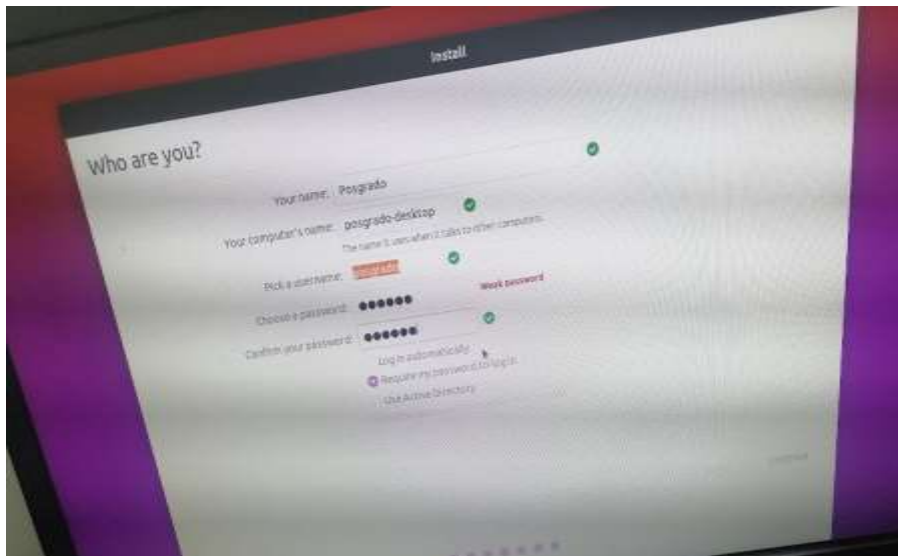


Figura 38: Formulario donde se registra el nombre de usuario y contraseña.

Finalmente, el sistema muestra una pantalla donde indica que la instalación y configuración de Ubuntu 20.04 ha concluido (ver Figura 39).

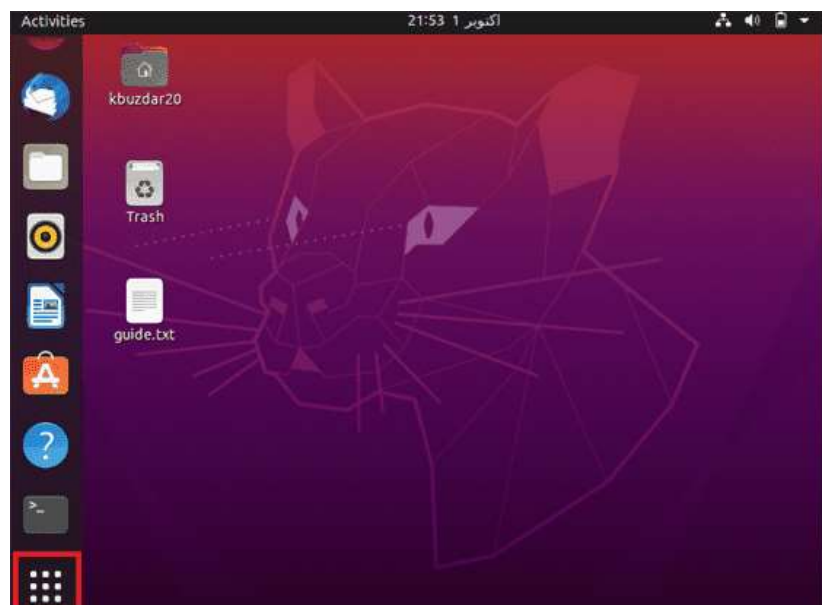


Figura 39: Inicio de Ubuntu.

Instalación de Drivers Nvidia

Como primer paso se actualiza la lista de librerías de Linux desde la terminal:

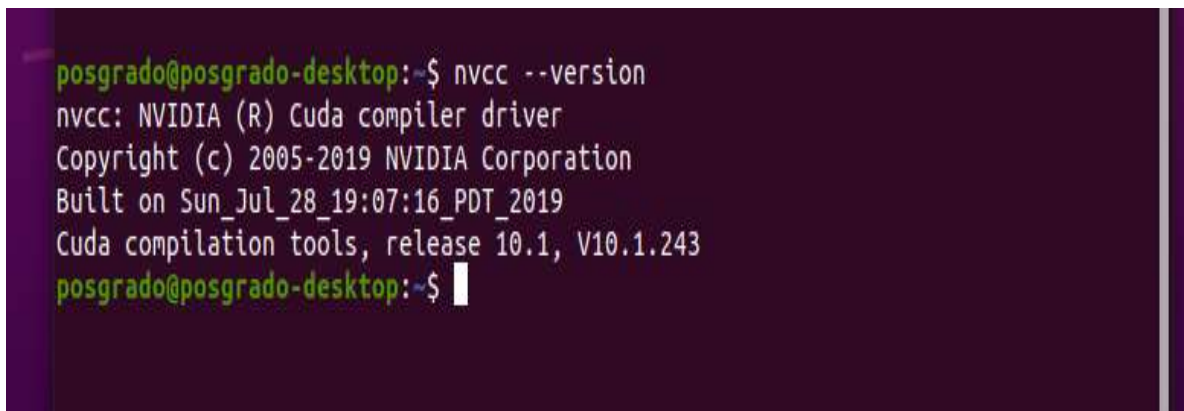
```
sudo apt update
```

Una vez actualizada la lista de librerías, se instalan los Drivers para la herramienta de desarrollo CUDA:

```
sudo apt install nvidia-cuda-toolkit
```

Se verifica que los Drivers se hayan instalado correctamente (ver Figura 40):

```
nvcc --version
```

A terminal window with a dark purple background. The prompt is 'posgrado@posgrado-desktop:~\$'. The command 'nvcc --version' has been entered, and the output is displayed in white text: 'nvcc: NVIDIA (R) Cuda compiler driver', 'Copyright (c) 2005-2019 NVIDIA Corporation', 'Built on Sun_Jul_28_19:07:16_PDT_2019', and 'Cuda compilation tools, release 10.1, V10.1.243'. The prompt is repeated at the bottom: 'posgrado@posgrado-desktop:~\$' followed by a cursor.

```
posgrado@posgrado-desktop:~$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Sun_Jul_28_19:07:16_PDT_2019
Cuda compilation tools, release 10.1, V10.1.243
posgrado@posgrado-desktop:~$
```

Figura 40: Resultado de consultar la versión de drivers de CUDA instalados

A continuación, se ejecuta el comando para mostrar la versión recomendada de los Drivers de la tarjeta Nvidia, dependiendo la tarjeta gráfica (ver Figura 21):

```
ubuntu-drivers devices
```

Posteriormente, se instalan los Drivers de la tarjeta, con la versión recomendada (ver Figura 41):

```
sudo apt install nvidia-driver-470
```



```
posgrado@posgrado-desktop:~$ ubuntu-drivers devices
WARNING:root:_pkg_get_support nvidia-driver-390: package has invalid Support Legacyheader, cannot determine support level
== /sys/devices/pci0000:00/0000:00:01.0/0000:01:00.0 ==
modalias : pci:v000010DEd00001189sv000019DAsd00002255bc03sc00i00
vendor   : NVIDIA Corporation
model    : GK104 [GeForce GTX 670]
driver   : nvidia-driver-418-server - distro non-free
driver   : nvidia-340 - distro non-free
driver   : nvidia-driver-390 - distro non-free
driver   : nvidia-driver-450-server - distro non-free
driver   : nvidia-driver-470 - distro non-free recommended
driver   : nvidia-driver-470-server - distro non-free
driver   : xserver-xorg-video-nouveau - distro free builtin

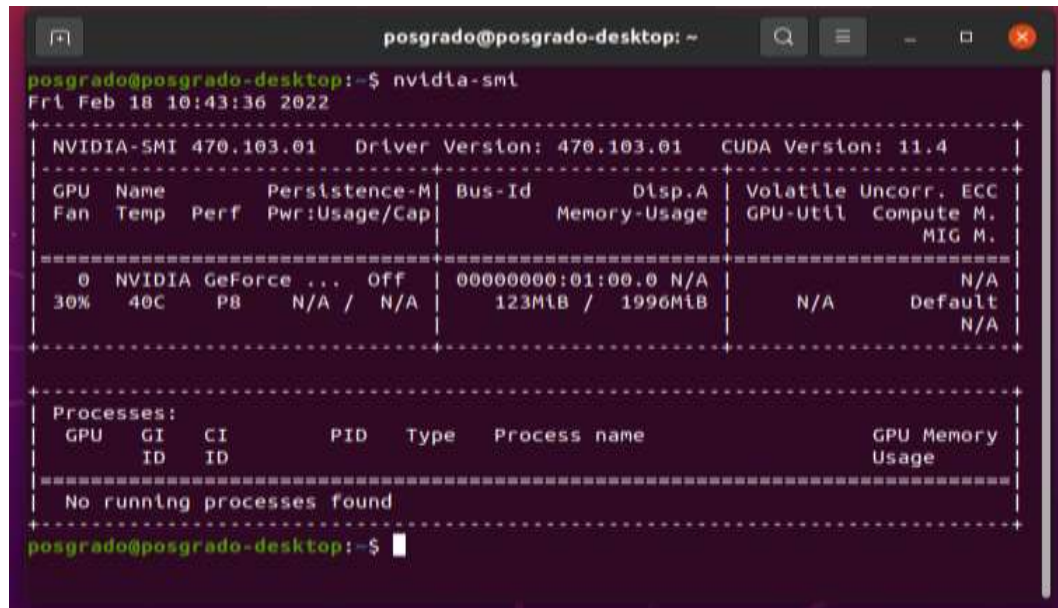
posgrado@posgrado-desktop:~$ cd /usr
posgrado@posgrado-desktop:/usr$ cd
posgrado@posgrado-desktop:~$ sudo apt install nvidia-driver-470
```

Figura 41: Ejemplo de instalación de drivers recomendados.

Reconocimiento y seguimiento de objetos en entornos controlados

Se verifica que la instalación se realizó de manera de la manera correcta (ver Figura 42).
Se recomienda reiniciar la computadora antes de realizar este paso.

`nvidia-smi`

A terminal window titled 'posgrado@posgrado-desktop: ~' showing the output of the 'nvidia-smi' command. The output displays NVIDIA-SMI 470.103.01, Driver Version: 470.103.01, and CUDA Version: 11.4. It also shows a table of GPU information for a single NVIDIA GeForce card, including fan speed (30%), temperature (40C), power usage (P8), and memory usage (123MiB / 1996MiB). Below this, it shows a table of running processes, but states 'No running processes found'.

```
posgrado@posgrado-desktop:~$ nvidia-smi
Fri Feb 18 10:43:36 2022

+-----+
| NVIDIA-SMI 470.103.01   Driver Version: 470.103.01   CUDA Version: 11.4   |
+-----+-----+
| GPU  Name            Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|=====+=====+
| 0.  NVIDIA GeForce ...  Off      | 00000000:01:00.0 N/A |         N/A         |
| 30%   40C    P8      N/A /  N/A   | 123MiB / 1996MiB |      N/A      Default |
+-----+-----+

+-----+
| Processes:                                                       GPU Memory |
|  GPU   GI    CI          PID    Type   Process name                  Usage   |
|=====+=====+
| No running processes found                                         |
+-----+

posgrado@posgrado-desktop:~$
```

Imagen 42: Drivers y versiones instalados.

Instalación de Python

A continuación se instala Python:

```
sudo apt install python3-pip
```

Se verifica la versión de Python instalada:

```
python3 --version
```

Instalación de Pytorch y Torchvisión

Se instala Pytorch y Torchvisión con soporte para CUDA:

```
pip3 install torch==1.8.2 torchvision==0.9.2 torchaudio==0.8.2 --extra-index-url https://download.pytorch.org/whl/lts/1.8/cu102
```

Instalación de Visual Studio Code

A continuación se instala Visual Studio Code como ambiente de desarrollo, ya que ofrece un amplio soporte para trabajar con varios lenguajes de programación, plataformas y sistemas operativos.

```
sudo snap install --classic code
```

Instalación de librerías

Finalmente, se instala un conjunto de librerías utilizadas para el desarrollo del proyecto:

```
pip install numpy
```

```
pip install opencv-python
```

```
pip install scipy
```

```
pip install matplotlib
```

```
pip install tqdm
```

```
pip install Pandas
```

Apéndice 2

Pre-procesamiento mediante VIA, VGG Image Anotattor

Es el software usado para la segmentación en este proyecto, es el VGG Image Anotattor. Este software crea anotaciones, es decir, guarda los puntos de las regiones definidas para cada imagen, indicando la clase de objeto a la que pertenece, generando un archivo JSON que contiene todos los polígonos que representan a cada objeto. Se puede acceder desde la liga: <https://annotate.officialstatistics.org/>

En la Figura 43 Se muestra la pantalla de inicio con las instrucciones.

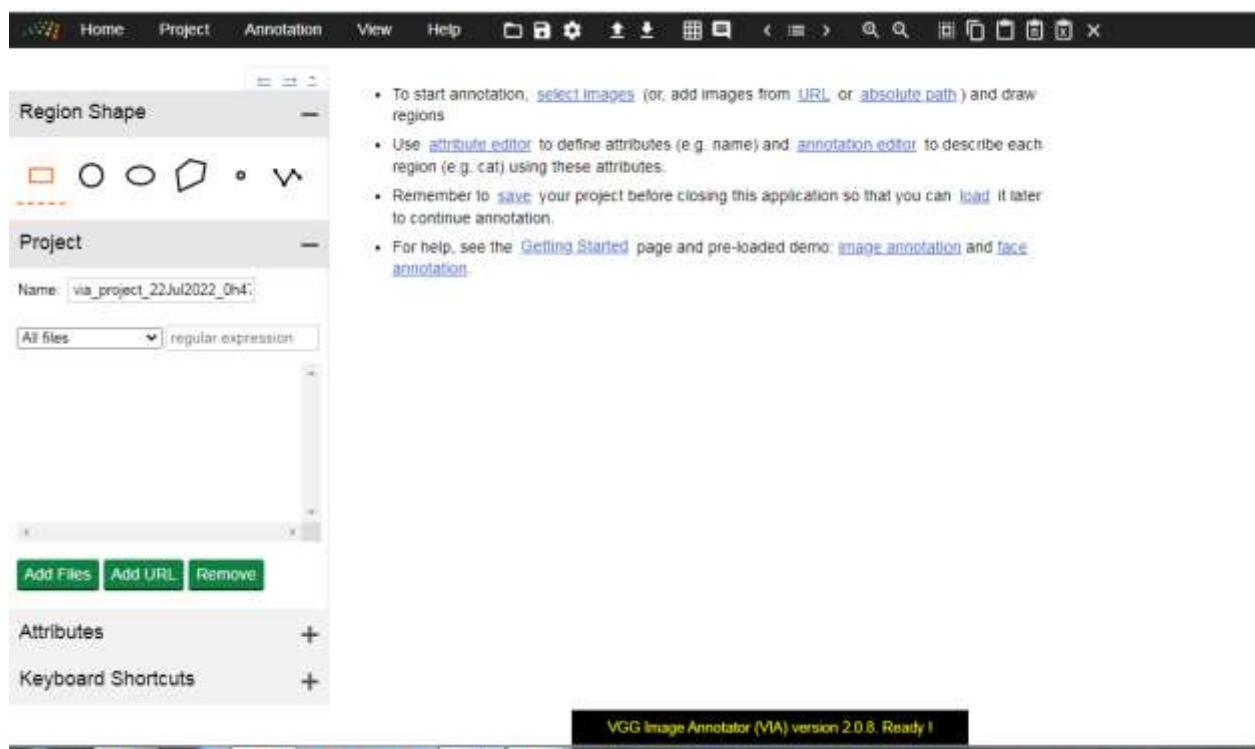


Figura 43: Página principal del software VIA

Primero, se carga el conjunto de imágenes al software, en “Project-add local files”. Una vez cargadas las imágenes se muestra la lista de estas en el menú inferior de la izquierda (ver Figura 44). Donde se asigna también el nombre en la opción “name”. Este nombre es el que se asigna al archivo JSON al finalizar.

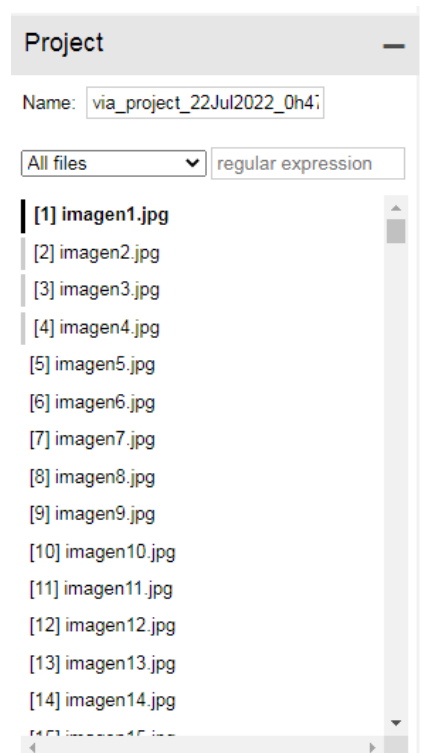


Figura 44: Conjunto de imágenes cargadas.

A continuación se debe seleccionar una imagen, y seleccionar la opción “Polygon” (ver Figura 45).

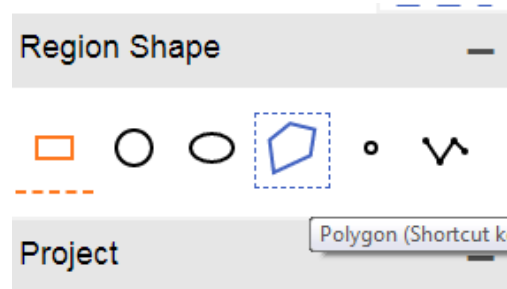


Figura 45: Tipos de seleccionadores disponibles.

Se selecciona el objeto de interés por los bordes, ligeramente por fuera del margen del objeto (ver Figura 46).

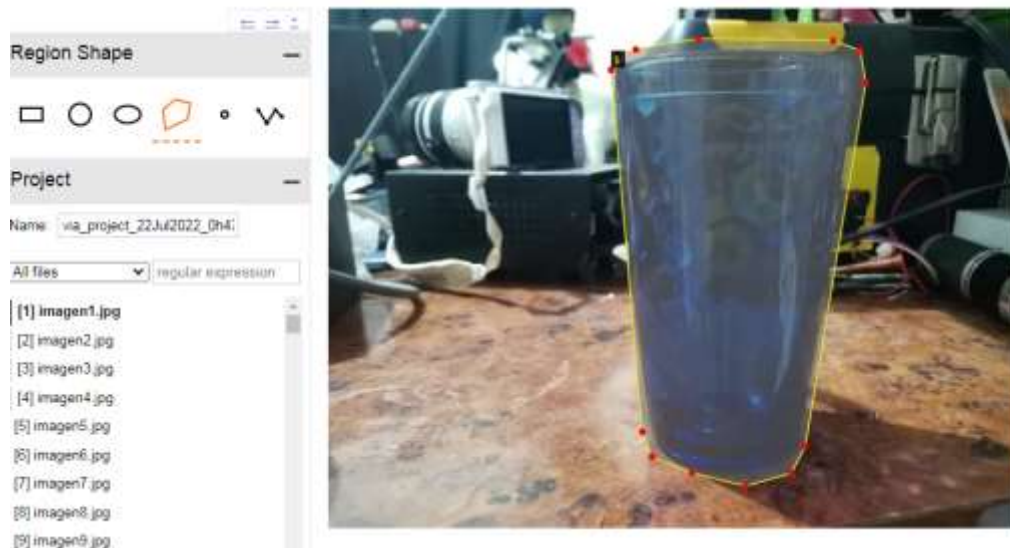


Figura 46: Ejemplo de selección de objeto en VIA.

Después de aplicar la selección del objeto de interés se indica el nombre de clase a la cual el objeto pertenece de la siguiente manera (ver Figura 47). Esto en “Attributes-RegionAttributes”, en donde se indica el nombre “clase”, importante de hacer. Finalmente se agrega, y se selecciona la opción. “Toggle Annotation Editor”.

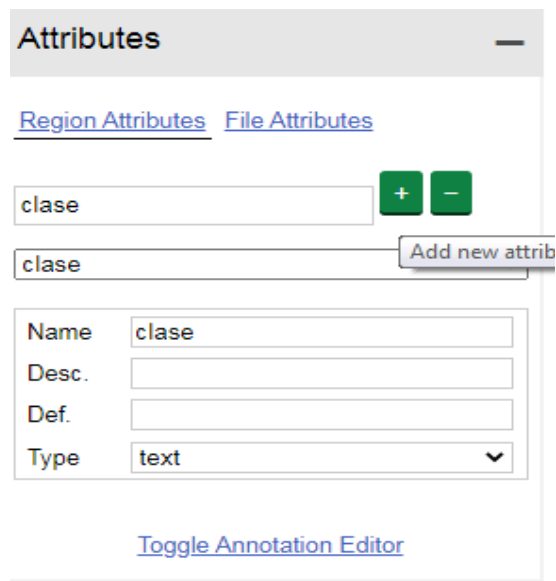


Figura 47: Menú de configuración del anotador de clases

Finalmente se agrega el nombre a cada clase que está representado por un índice (ver Figura

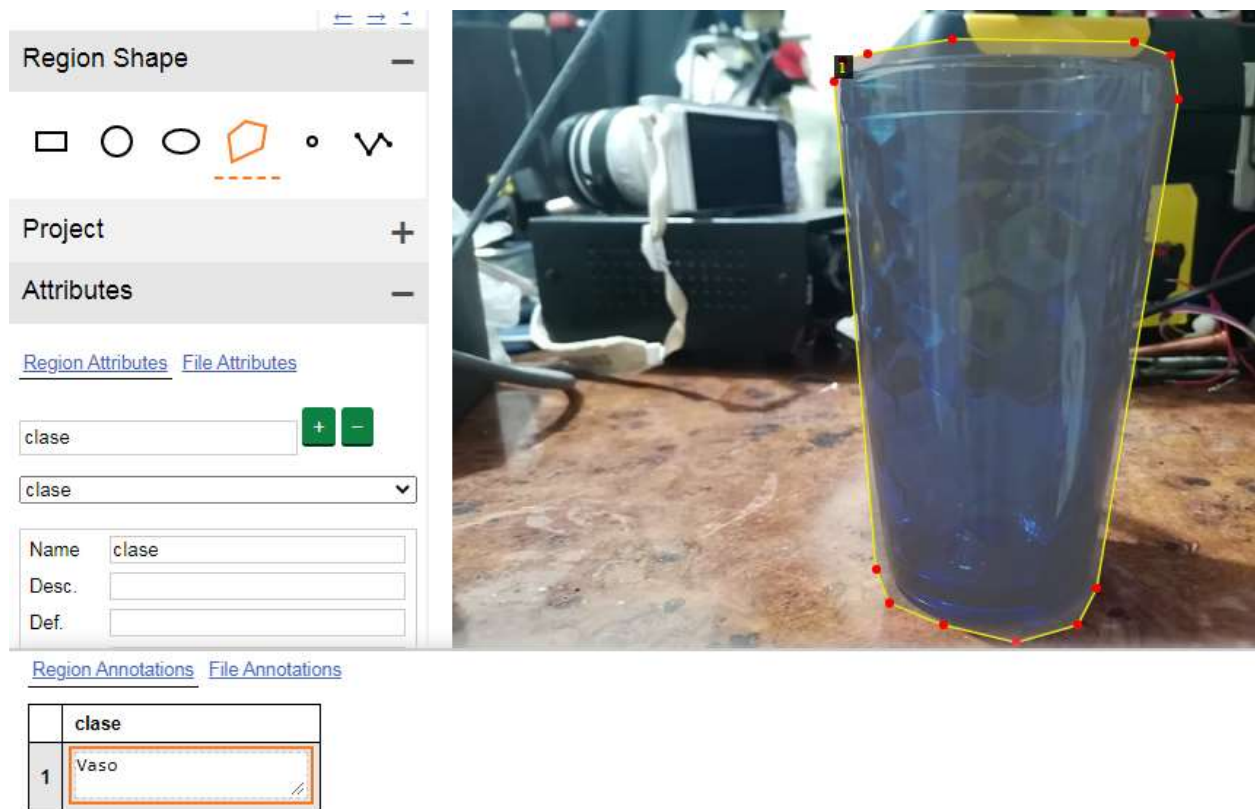


Figura 48: Ejemplo de objeto seleccionado, índice y clase perteneciente en VIA

Se repite este proceso de segmentación en cada una de las imágenes. Si en algún momento no se puede continuar o hacer la segmentación de todas las imágenes, la aplicación cuenta con la opción de cargar “Project-Save” para guardar el progreso y posteriormente cargar el archivo Json para continuar con la edición desde “Project-Load”.

Al finalizar la segmentación y almacenarla en “Project-Save” se genera el archivo JSON que se utiliza posteriormente para la generación de los archivos de máscaras.

Apéndice 3

Explicación de código para replicar estos experimentos

En esta sección se explica el proceso para replicar los experimentos, y se muestra la liga al repositorio del proyecto donde se encuentran los códigos documentados, referencias usadas y documentos de avances de la investigación.

Una vez preparado el entorno de trabajo, y segmentadas las imágenes en VIA, se crean los conjuntos de datos para iniciar el entrenamiento.

Posteriormente, se pueden descargar los archivos de script de la siguiente liga dentro del repositorio (<https://github.com/EloyFtw/Residencia-profesional/>) (ver Figura 49).

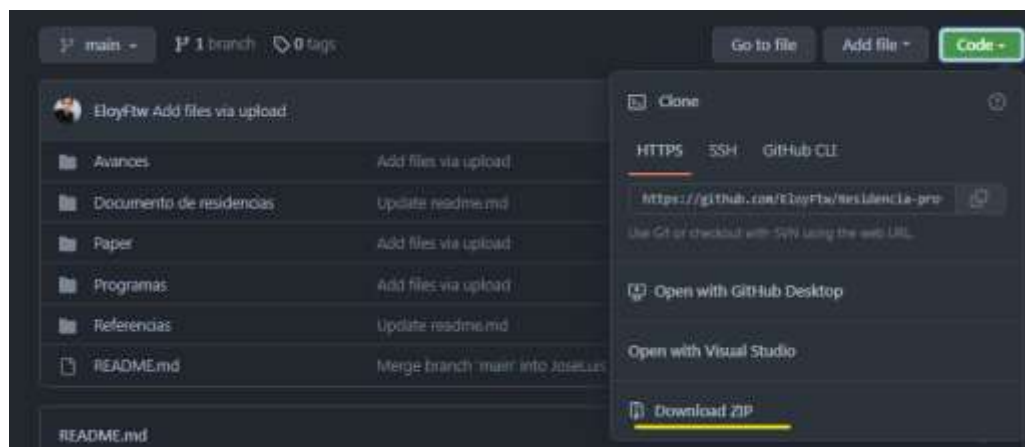


Figura 49: Forma de descargar los archivos del repositorio.

Dentro de la carpeta de archivos que se encuentra en el directorio: Residencia-profesional/Programas/ReplicarPruebas/ se crea la estructura de directorio que se muestra a continuación, que es donde se replica el experimento.

Carpeta con los archivos

→DataSet

→ ImágenesJPG

→MascarasPNG

→ ImágenesNpy

→MascarasNpy

Creación de archivos de máscaras en formato *png

El primer paso es renombrar el archivo JSON generado desde VIA con un nombre específico, en este caso *MRD.json*. A continuación se copian las imágenes que se van a usar para el entrenamiento al directorio DataSet/ImagenesJPG/.

En el script del archivo “*Crear Mascara.py*” (ver Figura 50) se define el color RGB que va a tener cada objeto indicando el nombre y color. Debe coincidir con el nombre asignado a la clase indicada en VIA.

```
if clase==regions['region_attributes']['clase']:
    polygons = regions['shape_attributes']

    if clase=='mesa': R,V,A= [128,0,0]
    elif clase=='vaso': R,V,A= [0,0,128]
    elif clase=='tasa': R,V,A= [128,128,0]
    elif clase=='plato': R,V,A= [128,0,128]
    elif clase=='celular': R,V,A= [128,128,128]
    else : R,V,A= 0,0,0
```

Figura 50: Definición de colores de los objetos

Posteriormente se define el nombre de la ruta de las imágenes, la ruta de donde se va a guardar las máscaras y el directorio del archivo Json generado anteriormente (ver Figura 51).

```
IMAGE_FOLDER = "./DataSet/ImagenesJPG/"
MASK_FOLDER = "./DataSet/MascarasPNG/"
PATH_ANNOTATION_JSON = './MRD.json'
```

Figura 51: Directorios usados en la creación de máscaras *png

Por último en este archivo se define el orden en que los objetos se van a colorear para definir el orden en que se va a segmenta cada objeto (ver Figura 52).

```
for i in range(len(regions)): colorear('mesa', regions[i])
for i in range(len(regions)): colorear('plato', regions[i])
for i in range(len(regions)): colorear('vaso', regions[i])
for i in range(len(regions)): colorear('tasa', regions[i])
for i in range(len(regions)): colorear('celular', regions[i])
```

Figura 52: Orden en que se va a colorear los objetos.

El script se ejecuta una vez configurados estos parámetros, y al finalizar, la carpeta máscaras debe contener el mismo número de elementos que en la carpeta imágenes.

Creación del conjunto de datos en formato Npy

En el script del archivo “preparacionImg-1.ipynb”(ver Figura 53), se indica la relación de colores de acuerdo a como se definió en el script del archivo “Crear Mascara.py”, se indica el índice al cual va a pertenecer cada objeto.

```
import torch

COLORMAP = [[0, 0, 0], [128, 128, 128], [128, 0, 0], [0, 0, 128], [128, 128, 0], [0, 128, 0]]

CLASSES = ['fondo', 'conector', 'celular', 'plato', 'vaso', 'mesa',

def colormap2label():
    colorlabel = torch.zeros(256 ** 3, dtype=torch.long)
    for i, colormap in enumerate(COLORMAP):
        colorlabel[(colormap[0] * 256 + colormap[1]) * 256 + colormap[2]] = i
    return colorlabel
```

Figura 53: Definición de índices para cada objeto

En caso de no tener el suficiente poder computacional para realizar el entrenamiento o la inferencia, es recomendable ejecutar el siguiente script (ver Figura 54) que ajusta el tamaño de las imágenes y máscaras a una dimensión de 240x320.

```
for archivo in archivos:
    imagen=torchvision.io.read_image('./Imagenes/ImagenesR/'+archivo+'.jpg')
    imagen=torchvision.transforms.functional.resize(imagen,(240,320))
    torchvision.io.write_jpeg(imagen,'./ImagenesEscaladas/'+archivo+'.jpg')

for archivo in archivos:
    mascara=torchvision.io.read_image('./Mascara/'+archivo+'.png',torchvisio
    mascara=torchvision.transforms.functional.resize(mascara,(240,320))
    print(mascara)
    for f in range(0,240):
        for c in range(0,320):
            if mascara[0][f][c]<64:
                mascara[0][f][c]=0
            elif mascara[0][f][c]<160:
                mascara[0][f][c]=128
```

Figura 54 Código para redimensionar las imágenes y mascaras

Finalmente, se generan los archivos en formato NPY en el directorio indicado anteriormente (ver Figura 55), estos archivos contienen los índices asociados a los colores y la normalización de los pixeles de cada una de las imágenes y máscaras.

```
for archivo in archivos:
    imagen=torchvision.io.read_image('./DataSet/ImagenesJPG/'+archivo+'.jpg')
    imagen=imagen.float()/255
    np.save('./DataSet/ImagenesNpy/'+archivo,imagen.numpy())

colorlabel = colormap2label()
for archivo in archivos:
    mascara=torchvision.io.read_image('./DataSet/MascarasPNG/'+archivo+'.png',tc
    mascara=label_indices(mascara,colorlabel)
    np.save('./DataSet/MascarasNpy/'+archivo, mascara.numpy())
```

Figura 55: Código para genera los conjuntos de datos en formato NPY.

Entrenamiento del modelo

El archivo de script “PyTorch Segmentación Semántica.ipynb” contiene el código para la configuración y el entrenamiento del modelo de red neuronal.

Primero se selecciona la ruta donde se encuentran los archivos *NPY (ver Figura 56).

```
import torch
import os
from pathlib import Path

# El alto y ancho de las imágenes debe ser divisible por 16 para que concierne
# convolucionales y convolucionales transpuestas.
path = Path('./DataSet/')
imgs = [path/'ImagenesNpy'/'i' for i in os.listdir(path/'ImagenesNpy/')]
masks = [path/'MascarasNpy'/'i' for i in os.listdir(path/'MascarasNpy/')]

len(imgs), len(masks)
```

Figura 56: Configuración de rutas de archivos *Npy en entrenamiento.

Se indica el número de clases (Cantidad de objetos diferentes) que va a reconocer el modelo.

```
class Dataset(torch.utils.data.Dataset):
    def __init__(self, X, y, n_classes=10):
        self.X = X
        self.y = y
        self.n_classes = n_classes

    def __len__(self):
        return len(self.X)

    def __getitem__(self, ix):
        img = np.load(self.X[ix])
        mask = np.load(self.y[ix])
        #img = torch.tensor(img).unsqueeze(0) Para el caso donde se trabaja con imágenes de 2D
        img = torch.tensor(img) # Para el caso donde se trabaja con imágenes de 3D
        mask = (np.arange(self.n_classes) == mask[...,None])
        return img, torch.from_numpy(mask).permute(2,0,1)
```

Figura 57: Configuración de número de clases en entrenamiento.

En esta sección de código se especifica la cantidad de imágenes que serán para entrenamiento y la cantidad de imágenes para pruebas. Se elige una cantidad para pruebas de aproximadamente el 25% del total de imágenes (ver Figura 58).

```
dataset = {
    'train': Dataset(imgs[:-50], masks[:-50]),
    'test': Dataset(imgs[-50:], masks[-50:])
}

len(dataset['train']), len(dataset['test'])
```

Figura 58: Configuración de cantidad de imágenes de prueba e imágenes de entrenamiento.

La siguiente sección de código es parte de la configuración del entrenamiento de la red, Se indica el número de clases, el cual debe coincidir con la cantidad de clases definidas previamente (ver Figura 59).

```
class UNetResnet(torch.nn.Module):
    def __init__(self, n_classes=10, in_ch=3):
        super().__init__()

        self.encoder = torchvision.models.resnet18(pretrained=True) # Para entrenamiento
        #self.encoder = torchvision.models.resnet18(pretrained=False) # Para predicción únicamente
        if in_ch != 3:
            self.encoder.conv1 = torch.nn.Conv2d(in_ch, 64, kernel_size=7, stride=2, padding=3, bias=False)

        self.deconv1 = deconv(512,256)
        self.deconv2 = deconv(256,128)
        self.deconv3 = deconv(128,64)
        self.out = out_conv(64, 64, n_classes)
```

Figura 59: Configuración de la red neuronal.

Por último, se asigna la cantidad de épocas de entrenamiento, la cual debe ser proporcional a la cantidad total de imágenes de prueba y entrenamiento (ver Figura 60). Una gran cantidad de épocas de entrenamiento puede sobre-entrenar el modelo especializándose en las imágenes de entrenamiento. Caso contrario, muchas imágenes y pocas épocas, pueden hacer que el modelo tenga dificultad para realizar un buen aprendizaje.

```
model = UNetResnet()  
#print(model)  
hist = fit(model, dataloader, epochs=100)
```

Figura 60: Configuración de las épocas de entrenamiento.

Configurar la cámara

Para configurar la cámara se usan el script “cam.py”, este configura la captura de imágenes (ver Figura 61). Se indica el número de índice en donde está conectada la cámara. Además se indica la ruta donde se guardarán las imágenes capturadas.

```
! cap = cv2.VideoCapture(1)  
! cap1 = cv2.VideoCapture(0)  
! print("Inicio")  
!  
! def tomarFoto(ix):  
!     leído, frame = cap.read()  
!     leído1, frame2 = cap1.read()  
!     print("Tomada")  
!     if leído == True and leído1 == True:  
!         cv2.imwrite("./fotos/foto"+str(ix)+".png", frame)  
!         cv2.imwrite("./fotos/foto"+str((ix+1))+".png", frame2)  
!         print("Foto guardada correctamente")  
!     else:  
!         print("Error al acceder a la cámara")  
!
```

Figura 61: Código para captura de imágenes.

Referencias

- *Arquitectura U-Net.* (26 Abril, 2022). © 2022 DataScientest. Recuperado de: <https://datascientest.com/es/u-net-lo-que-tenes-que-saber>.
- *Bernard Marr.* (8 de Abril, 2019). Forbes Media LLC. All Rights. 7 Amazing Examples Of Computer And Machine Vision In Practice Recuperado de: <https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/?sh=5f2b1c671018>
- *Condori Arias, E. F.* (2013). Reconocimiento y clasificación de objetos usando inteligencia artificial basada en SVM y visión estereoscópica.
- *D. Purwanto, M. Rivai and H. Soebhakti,* "Vision-based multi-point sensing for corridor navigation of Autonomous Indoor Vehicle," 2017 International Conference on Electrical Engineering and Computer Science (ICECOS), 2017, pp. 67-70, doi: 10.1109/ICECOS.2017.8167168.
- *Diddeniya, S. I. A. P., Adikari, A. M. S. B., Gunasinghe, H. N., Silva, P. R. S. D., Ganegoda, N. C., and Wanniarachchi, W. K. I. L.* (2018). Vision based office assistant robot system for indoor office environment. 3rd International Conference on Information Technology Research (ICITR).
- *Emani, S., Soman, K., Sajith, V., and Adarsh, S.* (2019). Obstacle Detection and Distance Estimation for Autonomous Electric Vehicle Using Stereo Vision and DNN. In: Wang, J., Reddy, G., Prasad, V., Reddy, V. (eds) Soft Computing and Signal Processing. Advances in Intelligent Systems and Computing, 898:639–648
- *Ferreira, L.* (2013). Localization and navigation in an autonomous vehicle. Universidade de Aveiro Departamento de Electrónica, Telecomunicaciones e Informática.
- *G. Yang, S. Wang and J. Yang,* "Desire-Driven Reasoning for Personal Care Robots," in IEEE Access, vol. 7, pp. 75203-75212, 2019, doi: 10.1109/ACCESS.2019.2921112.
- *Gary Explains.* (17 mayo, 2020). Jetson Nano. 330Ohms Blog. ¿Que puedo hacer con un Jetson nano?. Recuperado de: <https://blog.330ohms.com/2020/05/17/que-puedo-hacer-con-una-jetson-nano/>

- H. Alikarami, F. Yaghmaee and M. J. Fadaeieslam, "Sparse representation and convolutional neural networks for 3D human pose estimation," 2017 3rd Iranian Conference on Intelligent Systems and Signal Processing (ICSPIS), 2017, pp. 188-192, doi: 10.1109/ICSPIS.2017.8311614.
- J. Mišeikis et al., "Lio-A Personal Robot Assistant for Human-Robot Interaction and Care Applications," in *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5339-5346, Oct. 2020, doi: 10.1109/LRA.2020.3007462.
- Jishnu, K., Indu, V., Ananthakrishnan, K., Amith, K., Reddy, P., and Pramod, S. (2020). Voice controlled personal assistant robot for elderly people. *Proceedings of the Fifth International Conference on*
- John McCarthy. (2004). Computer Science Department. What is artificial intelligence? pp. 2, Stanford, CA 94305. Recuperado de: https://borghese.di.unimi.it/Teaching/AdvancedIntelligentSystems/Old/IntelligentSystems_2008_2009/Old/IntelligentSystems_2005_2006/Documents/Symbolic/04_McCarthy_whatissai.pdf
- Judith Hurwitz, & Daniel Kirsch, (2018), *Machinne Learning for Dummies*, IBM Limited Version, Published by John Wiley & Sons, Inc. 111 River St. Hoboken, NJ 07030-5774, p. 4.
- Ketkar, Nikhil (2017). «Introduction to PyTorch». *Deep Learning with Python*. Apress, Berkeley, CA. pp. 195-208. ISBN 9781484227657. doi:10.1007/978-1-4842-2766-4_12.
- Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168.
- M. Turing (1950) *Computing Machinery and Intelligence*. *Mind* 49: 433-460. Recuperado de: <https://www.csee.umbc.edu/courses/471/papers/turing.pdf>
- Michael Barr. (25 de junio de 2013). *Embedded Systems Glossary*. Netrino Technical Library. Recuperado de: <http://www.netrino.com/Embedded-Systems/Glossary>
- Mukherjee, A., Adarsh, S., and Ramachandran, K. I. (2020). Ros-based pedestrian detection and distance estimation algorithm using stereo vision, Leddar and cnn.

- *Rahul and Nair, B. B. (2018). Camera-Based Object Detection, Identification and Distance Estimation. 2018 2nd International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE), pages 203–205*
- *Ramos Rivas Melva, (18 Diciembre, 2003). Tesis Sistema de pre-procesamiento de imágenes electrocardiográficas en telemedicina. Departamento de Ingeniería en Sistemas Computacionales. Escuela de Ingeniería, Universidad de las Américas Puebla., Cap. 3.*
- *Ramírez, Fran (20 de julio de 2018). Historia de la IA: Frank Rosenblatt y el Mark I Perceptrón, el primer ordenador fabricado específicamente para crear redes neuronales en 1957.*
- *Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. <https://doi.org/10.1109/CVPR.2017.690>.*
- *Rocha A . y Goldstein S.K. (2014), Multiclass from binary: Expanding one-versus-all, one-versus-one and ecoc-based approaches. IEEE Transactions on Neural Networks and Learnings Systems 25(2): 289-302.*
- *Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. Medical Image Computing and Computer-Assisted Intervention, 9351.*
- *Sarahí Silva, Estefanía Freire. (23 Nov. 2019). Intro a las redes neuronales convolucionales. BootCamp AI. Recuperado de: <https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8>*
- *Scipy (15 julio 2022). [scipy.optimize.curve_fit](https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html). API reference. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html.*
- *Sensio, J. (3 octubre 2020). Vision Artificial - Segmentation Semantic. Sensio Inteligencia Artificial. [https://juansensio.com/blog/050 cv segmentación](https://juansensio.com/blog/050%20cv%20segmentaci%C3%B3n).*
- *TheBigDataUniversity. Aprendizaje automático: aprendizaje supervisado y no supervisado, clase cognitiva, 13 de marzo de 2017. Recuperado de: <https://www.youtube.com/watch?v=cjf6yaYE86U&list=PL-XeOa5hMEYz7xMckkUL8w2EKzM3TDrON&index=5>*
- *Tsung-Yi Li, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C. Lawrence Zitnick. (2014). Microsoft COCO: Common Objects in Context , Publicado, 2014. 740-755 ECCV*

2014. Springer International. In *European conference on computer vision*, pp. 740–755. Springer. Recuperado de: <https://www.microsoft.com/en-us/research/wp-content/uploads/2014/09/LinECCV14coco.pdf>

- Yu M., Gong L., y Kollias S. (2017). *Computer vision baseed fall detection by a convolutional neural network*. In *Proccedings of the 19th ACM international Conference on multimodal interaction*, pp. 416/420. ACM.
- ¿Qué es deep learning?. (1 Mayo, 2020). IBM Corporation. IBM Cloud Education. Recuperado de: <https://www.ibm.com/mx-es/cloud/deep-learning>.
- ¿Qué es Machine Learning?. (15 Julio, 2020). IBM Corporation. IBM Cloud Education. Recuperado de: <https://www.ibm.com/mx-es/analytics/machine-learning>.
- ¿Qué son las redes neuronales?. (17 Agosto 2020). IBM Corporation. IBM Cloud Education. Recuperado de: <https://www.ibm.com/mx-es/cloud/learn/neural-networks>.

Anexo B.

Ponencia en el noveno Congreso Internacional de Robótica (CIRC) el 11 de Mayo del 2022 con el proyecto *Reconocimiento y triangulación de objetos en entornos controlados*.





DESDE 2013
<https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive>
Padi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI



Publicación Semestral Padi Vol. X No. 00 (2022) 1-13

Reconocimiento y estimación de distancia relativa de objetos en entornos controlados Recognition and estimation of relative distance of objects in controlled environments

Eloy Clemente *, Enrique Luna *, José Luis Gómez *, Isaac Villa *

*Tecnológico Nacional de México Campus La Paz, Boulevard Forjadores de Baja California Sur No. 4720 C.P. 23080

Resumen

En este proyecto, se presenta un sistema de reconocimiento y clasificación de objetos que se encuentran en su entorno, para un robot de asistencia, así como la estimación de la distancia relativa de estos con respecto al robot. Para el reconocimiento y clasificación de los objetos, se aplicaron técnicas de visión artificial basadas en herramientas de segmentación semántica, como son redes neuronales convolucionales. Para la estimación de la distancia relativa de los objetos, una vez identificados, se implementó una técnica de visión estereoscópica. Los resultados de los experimentos muestran un 90.6 % de precisión en el reconocimiento y clasificación, y un error medio de 4.6 cm al estimar la distancia de los objetos.

Palabras Clave: Visión artificial, Redes neuronales convolucionales, Segmentación semántica, Visión estereoscópica, Inteligencia artificial.

Abstract

This project presents a system for recognition and classification of objects that are in its environment, for an assistance robot, as well as the estimation of their relative distance with respect to the robot. For the recognition and classification of objects, we apply artificial vision techniques based on semantic segmentation tools, such as convolutional neural networks. For the estimation of the relative distance of the objects, once identified, a stereoscopic vision technique was implemented. The results of the experiments show a 90.6 % accuracy in recognition and classification, and an average error of 4.6 cm when estimating the distance of the objects.

Keywords: Computer vision, Convolutional neural networks, Semantic segmentation, Stereoscopic vision, Artificial intelligence.

Anexo A.

Carta de aceptación



LA PAZ, B.C.S. 09 DE MARZO 2022

INSTITUTO TECNOLÓGICO DE LA PAZ

Dr. Mario Cortés Larringa

CON ATENCIÓN A:

Lic. Paola Cristina Torres Ortega

JEFA DE DIVISIÓN DE ESTUDIOS PROFESIONALES

Por medio de la presente hago constar que el **C. ELOY ANTONIO CLEMENTE ROSAS** con número de control **17310793** de la carrera **INGENIERIA EN SISTEMAS COMPUTACIONALES**, ha sido aceptado en nuestra empresa, para realizar sus residencias profesionales, mediante el desarrollo del proyecto denominado **RECONOCIMIENTO Y SEGUIMIENTO DE OBJETOS EN ENTORNOS CONTROLADOS**

Sin más por el momento quedo de usted para cualquier aclaración o duda.

ATENTAMENTE

MSC. José Luis Gómez Torres

RFC: GOTL8806187S6



Instituto Tecnológico de La Paz
Departamento de Gestión Tecnológica y Vinculación

La Paz, B.C.S., 24/ENERO/2022

No. de Oficio: DGTV/051/2022

ASUNTO: PRESENTACIÓN DEL ESTUDIANTE
Y AGRADECIMIENTO

C. MSC. JOSÉ LUIS GÓMEZ TORRES
PROPIETARIO DE OLIN ROBOTICS
PRESENTE

El Instituto Tecnológico de La Paz, tiene a bien presentar a sus finas atenciones al (la) C. ELOY ANTONIO CLEMENTE ROSAS con número de control 17310793 de la carrera de INGENIERÍA EN SISTEMAS COMPUTACIONALES quien desea desarrollar el proyecto de Residencias Profesionales, denominado "RECONOCIMIENTO Y SEGUIMIENTO DE OBJETOS EN ENTORNOS CONTROLADOS." cubriendo un total de 500 horas, en un período de cuatro a seis meses.

Es importante hacer de su conocimiento que todos los estudiantes que se encuentran inscritos en esta institución cuentan con un seguro contra accidentes personales con la empresa THONA SEGUROS, según póliza AP-TEC-021-01 e inscripción en el IMSS.

Así mismo, hacemos patente nuestro sincero agradecimiento por su buena disposición y colaboración para que nuestros estudiantes, aun estando en proceso de formación, desarrollen un proyecto de trabajo profesional, donde puedan aplicar el conocimiento y el trabajo en el campo de acción en el que se desenvolverán como futuros profesionistas.

Al vernos favorecidos con su participación en nuestro objetivo, sólo nos resta manifestarle la seguridad de nuestra más atenta y distinguida consideración.

ATENTAMENTE

Excelencia en Educación Tecnológica

L.C. JESÚS ISRAEL GALLO GASTELUM
JEFE DEL DEPARTAMENTO DE GESTIÓN
TECNOLÓGICA Y VINCULACIÓN.

ccp. Archivo
JIGG/lgs



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LA PAZ
GESTIÓN TECNOLÓGICA Y VINCULACIÓN

[Handwritten signature]
Recibo




Boulevard Forjadores de B.C.S. #4720, Col. 8 de Octubre 1ra Sección, C.P. 23080, La Paz,
B.C.S., Tel. (612) 12 10424 e-mail: ilza.gs@lapaz.tecnm.mx | lapaz.tecnm.mx




2022 Flores
Magón

Carta convenio.



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de La Paz
Departamento de Gestión Tecnológica y Vinculación

ACUERDO DE TRABAJO QUE CELEBRAN POR UNA PARTE EL **TECNOLÓGICO NACIONAL DE MÉXICO / INSTITUTO TECNOLÓGICO DE LA PAZ**, REPRESENTADO POR LA SUBDIRECCIÓN DE PLANEACIÓN Y VINCULACIÓN, POR ACUERDO DEL DIRECTOR ING. JESÚS DAVID ESTRADA RUÍZ, POR LA OTRA "OLIN ROBOTICS", REPRESENTADO(A) EN ESTE ACTO POR EL (LA) C. MSC. JOSÉ LUIS GÓMEZ TORRES, POR LA OTRA EL (LA) C. ELOY ANTONIO CLEMENTE ROSAS, A QUIENES EN LO SUCESIVO SE LES DENOMINARÁ "EL TECNOLÓGICO", "LA EMPRESA" Y "EL ESTUDIANTE" RESPECTIVAMENTE, SUJETÁNDOSE A LAS SIGUIENTES DECLARACIONES Y CLÁUSULAS:

I.- DECLARACIONES:

Declara: **"EL TECNOLÓGICO"**


- 1.1. Ser una Institución dedicada a impartir educación superior tecnológica de acuerdo a las necesidades que demanden los sectores productivos regionales y nacionales, adscrita al Tecnológico Nacional de México dependiente de la Secretaría de Educación Pública y dotado de plena capacidad para suscribir el presente acuerdo de trabajo.
- 1.2. Que derivado de la Reforma de la Educación Superior Tecnológica, se incorporó a los planes de estudio de las carreras que se imparten en el Sistema Nacional de Educación Superior Tecnológica, el programa de Residencias Profesionales, y dado el carácter curricular de este programa se hace necesario fortalecer la vinculación entre la educación superior y los sectores productivos de bienes y servicios.
- 1.3. Que tiene establecido su domicilio en Boulevard Forjadores de Baja California Sur No. 4720 y que su R.F.C. es **TNM140723GFA**.

Declara **"LA EMPRESA"**.




- 1.4. Que su misión es: PERMITIR QUE LAS PERSONAS CON RETOS FÍSICOS PUEDAN CONSTRUIR SUS PROPIAS SOLUCIONES PARA AUTONOMÍA USANDO LA ROBÓTICA, LA PROGRAMACIÓN Y LA ELECTRÓNICA COMO HERRAMIENTAS.
- 1.5. Que su domicilio se encuentra establecido en: Calle Sepia No. 370 entre Art. 16 y Magenta, Col. Fraccionamiento Arcoiris III, C.P. 23088 en la Ciudad de La Paz, Baja California Sur, que su R.F.C. es **GOTL8806187S6**.

Declara **"EL ESTUDIANTE"**


- 1.6. Ser estudiante regular de **"EL TECNOLÓGICO"** inscrito en la carrera de **INGENIERÍA EN SISTEMAS COMPUTACIONALES**, con número de control **17310793**.
- 1.7. Haber acreditado como mínimo el **80%** del total de los créditos del Plan de Estudios de la carrera a que se hace referencia en la declaración 1.6.
- 1.8. Que desea realizar su Residencia Profesional en **"LA EMPRESA"**
- 1.9. Que el proyecto denominado **"RECONOCIMIENTO Y SEGUIMIENTO DE OBJETOS EN ENTORNOS CONTROLADOS."** es producto de la investigación preliminar realizada y que previo aval de la Academia de **SISTEMAS Y COMPUTACIÓN**, ha sido autorizado para su desarrollo por el Departamento de **SISTEMAS Y COMPUTACIÓN**.



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LA PAZ
GESTIÓN TECNOLÓGICA Y VINCULACIÓN



Boulevard Forjadores de B.C.S. #4720, Col. B de Octubre 3ra Sección, C.P. 23080, La Paz,
B.C.S., Tel. (612) 12 10424 e-mail: liza.gs@lapaz.tecnm.mx | lapaz.tecnm.mx



2022 **Ricardo Flores**
ALMA MATER

**EDUCACIÓN**
SECRETARÍA DE EDUCACIÓN PÚBLICA

**TECNOLÓGICO NACIONAL DE MÉXICO**

Instituto Tecnológico de La Paz
Departamento de Gestión Tecnológica y Vinculación

II.- CLÁUSULAS:

PRIMERA. "LA EMPRESA" Se compromete a brindar todas las facilidades necesarias tanto al asesor del proyecto como a "EL ESTUDIANTE" para el desarrollo del proyecto de Residencias Profesionales.

SEGUNDA. "EL ESTUDIANTE" Se compromete a cumplir en tiempo y forma con todas las condiciones, características y requisitos que se mencionan en el proyecto a que se hace referencia en la declaración 1.9 y que es parte integrante del presente Acuerdo de Trabajo.

TERCERA. "EL ESTUDIANTE" Se compromete a sujetarse al Manual de Procedimientos para la Planeación, Operación y Acreditación de las Residencias Profesionales vigente, **cumpliendo con un total de 500 horas en un periodo de 4 a 6 meses, a partir de la fecha en que se suscribe el presente convenio.**

CUARTA. "EL ESTUDIANTE" cuenta con un seguro contra accidentes que pudieran ocurrir con motivo del desarrollo del proyecto que se menciona en la declaración 1.9 para lo cual "EL TECNOLÓGICO" a través del Tecnológico Nacional de México, tiene contratado seguro contra accidentes personales con la empresa THONA SEGUROS, según póliza AP-TEC-021-01 e inscripción en el IMSS.

Así mismo "EL TECNOLÓGICO" se compromete a verificar que "EL ESTUDIANTE" tenga actualizado sus servicios médicos.

QUINTA. Con motivo del desarrollo de algún proyecto de Residencias Profesionales y debido al carácter académico que tienen "EL ESTUDIANTE" acepta que su participación no originará ninguna relación laboral ni de seguridad social con "LA EMPRESA".

SEXTA. Los apoyos que "LA EMPRESA" pueda otorgar a "EL ESTUDIANTE" serán de acuerdo a su normatividad y disponibilidad presupuestal.

SÉPTIMA. "EL ESTUDIANTE" y "EL TECNOLÓGICO" se comprometen con "LA EMPRESA" a resguardar la confidencialidad de la información que surja con motivo de la implementación del presente acuerdo y que se aplique única y exclusivamente al desarrollo del proyecto a que se refiere la declaración 1.9.

OCTAVA: una vez Leído que fue el presente Acuerdo y enteradas las partes de su contenido, alcance y fuerza legal, se firma al margen y al calce en la ciudad de La Paz, Baja California Sur a los veinticuatro días del mes de enero del año dos mil veintidos.

**LA EMPRESA**
MSC. JOSÉ LUIS GÓMEZ TORRES
PROPIETARIO DEL OLIN ROBOTICS

**EL ESTUDIANTE**
ELOY ANTONIO CLEMENTE ROSAS
NO. DE CONTROL 17310793
INGENIERÍA EN SISTEMAS COMPUTACIONALES

**EL TECNOLÓGICO**
M.C. JORGE LUIS ALBARRÁN PARRA
SUBDIRECTOR DE PLANEACIÓN Y VINCULACIÓN


TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LA PAZ
GESTIÓN TECNOLÓGICA Y VINCULACIÓN
Boulevard Forjadores de B.C.S. #4720, Col. B de Octubre 1ra Sección, C.P. 23000, La Paz,
B.C.S. Tel: (612) 12 10424 e-mail: liza.gs@lapaz.tecnim.mx | lapaz.tecnim.mx**2022** *Flóres*
año de Maestría



**TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LA PAZ**

EVALUACIÓN Y SEGUIMIENTO DE RESIDENCIA PROFESIONAL POR COMPETENCIAS

Nombre del Residente: Eloy Antonio Clemente Rosas Número de control: 17310793
 Nombre del proyecto: Reconocimiento y seguimiento de objetos en entornos controlados
 Programa Educativo: Ingeniería en Sistemas Computacionales
 Periodo de realización de la Residencia Profesional: 24/01/2022 al 22/07/2022
 Calificación Final (promedio de ambas evaluaciones):

En qué medida el residente cumple con lo siguiente			
Evaluación por el asesor externo	Criterios a evaluar	Valor	Evaluación
	Asiste puntualmente en el horario establecido	5	5
	Trabaja en equipo y se comunica de forma efectiva (oral y escrita)	10	9
	Tiene iniciativa para colaborar	5	5
	Propone mejoras al proyecto	10	9
	Cumple con los objetivos correspondientes al proyecto	15	15
	Es ordenado y cumple satisfactoriamente con las actividades encomendadas en los tiempos establecidos del cronograma	15	15
	Demuestra liderazgo en su actuar	10	10
	Demuestra conocimiento en el área de su especialidad	20	20
	Demuestra un comportamiento ético (es disciplinado, acata órdenes, respeta a sus compañeros de trabajo, entre otros)	10	10
Calificación total		100	98

Observaciones:

		14/02/22
José Luis Gómez Torres Nombre y firma del asesor externo	Trámite sello Sello de la empresa, organismo o dependencia	Fecha de Evaluación

En qué medida el residente cumple con lo siguiente			
Evaluación por el asesor interno	Criterios a evaluar	Valor	Evaluación
	Asistió puntualmente a las reuniones de asesoría	10	10
	Demuestra conocimiento en el área de su especialidad	20	20
	Trabaja en equipo y se comunica de forma efectiva (oral y escrita)	15	13
	Es dedicado y proactivo en las actividades encomendadas	20	20
	Es ordenado y cumple satisfactoriamente con las actividades encomendadas en los tiempos establecidos en el cronograma	20	20
	Propone mejoras al proyecto	15	15
Calificación total		100	98

Observaciones:

		15/02/22
Jorge Enrique Luna Taylor Nombre y firma del asesor interno	SECRETARÍA DE EDUCACIÓN PÚBLICA TECNOLÓGICO NACIONAL DE MÉXICO INSTITUTO TECNOLÓGICO DE LA PAZ SEGUIMIENTO DE SISTEMAS Y COMPUTACIÓN	Fecha de Evaluación

ITLP-AC-PO-006-08

Toda copia en PAPEL es un "Documento No Controlado" a excepción del original

Rev. 01



**TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LA PAZ**

EVALUACIÓN Y SEGUIMIENTO DE RESIDENCIA PROFESIONAL POR COMPETENCIAS

Nombre del Residente: Eloy Antonio Clemente Rosas Número de control: 17310793
 Nombre del proyecto: Reconocimiento y seguimiento de objetos en entornos controlados
 Programa Educativo: Ingeniería en Sistemas Computacionales
 Periodo de realización de la Residencia Profesional: 24/01/2022 al 22/07/2022
 Calificación Final (promedio de ambas evaluaciones):

En qué medida el residente cumple con lo siguiente			
Criterios a evaluar		Valor	Evaluación
Evaluación por el asesor externo	Asiste puntualmente en el horario establecido	5	5
	Trabaja en equipo y se comunica de forma efectiva (oral y escrita)	10	8
	Tiene iniciativa para colaborar	5	5
	Propone mejoras al proyecto	10	10
	Cumple con los objetivos correspondientes al proyecto	15	15
	Es ordenado y cumple satisfactoriamente con las actividades encomendadas en los tiempos establecidos del cronograma	15	15
	Demuestra liderazgo en su actuar	10	10
	Demuestra conocimiento en el área de su especialidad	20	20
	Demuestra un comportamiento ético (es disciplinado, acata órdenes, respeta a sus compañeros de trabajo, entre otros)	10	10
Calificación total		100	98

Observaciones:

		15/04/2022
José Luis Gómez Torres Nombre y firma del asesor externo	Trámite sello Sello de la empresa, organismo o dependencia	Fecha de Evaluación

En qué medida el residente cumple con lo siguiente			
Criterios a evaluar		Valor	Evaluación
Evaluación por el asesor interno	Asistió puntualmente a las reuniones de asesoría	10	10
	Demuestra conocimiento en el área de su especialidad	20	20
	Trabaja en equipo y se comunica de forma efectiva (oral y escrita)	15	13
	Es dedicado y proactivo en las actividades encomendadas	20	20
	Es ordenado y cumple satisfactoriamente con las actividades encomendadas en los tiempos establecidos en el cronograma	20	20
	Propone mejoras al proyecto	15	15
Calificación total		100	98

Observaciones:

		18/04/2022
Jorge Enrique Luna Taylor Nombre y firma del asesor interno	SECRETARÍA DE EDUCACIÓN PÚBLICA TECNOLÓGICO NACIONAL DE MÉXICO INSTITUTO TECNOLÓGICO DE LA PAZ DEPARTAMENTO DE SISTEMAS COMPUTACIONALES Sello de la institución	Fecha de Evaluación



**TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LA PAZ**

EVALUACIÓN DE REPORTE DE RESIDENCIA PROFESIONAL POR COMPETENCIAS

Nombre del Residente: Eloy Antonio Clemente Rosas Número de control: 17310793
 Nombre del proyecto: Reconocimiento y seguimiento de objetos en entornos controlados
 Programa Educativo: Ingeniería en Sistemas Computacionales
 Período de realización de la Residencia Profesional: 24/01/2022 al 22/07/2022
 Calificación Final (promedio de ambas evaluaciones):

En qué medida el residente cumple con lo siguiente		
Evaluación por el asesor externo	Criterios a evaluar	Valor
	Portada	2
	Agradecimientos	2
	Resumen	2
	Índice	2
	Introducción	2
	Problemas a resolver, priorizándolos	5
	Objetivos	5
	Justificación	
	Marco teórico (fundamentos teóricos)	10
	Procedimiento y descripción de las actividades realizadas	5
	Resultados, planos, gráficas, prototipos, manuales, programas, análisis estadísticos, modelos matemáticos, simulaciones, normativas, regulaciones y restricciones, entre otros. Solo para proyectos que por su naturaleza lo requieran: estudio de mercado, estudio técnico y estudio económico.	45
	Conclusiones, recomendaciones y experiencia profesional adquirida	15
	Competencias desarrolladas y/o aplicadas	3
	Fuentes de información	2
Calificación total		100

Observaciones:

		25/07/2022
José Luis Gómez Torres Nombre y firma del asesor externo	Sello de la empresa, organismo o dependencia	Fecha de Evaluación

En qué medida el residente cumple con lo siguiente		
Evaluación por el asesor interno	Criterios a evaluar	Valor
	Portada	2
	Agradecimientos	2
	Resumen	2
	Índice	2
	Introducción	2
	Problemas a resolver, priorizándolos	5
	Objetivos	5
	Justificación	
	Marco teórico (fundamentos teóricos)	10
	Procedimiento y descripción de las actividades realizadas	5
	Resultados, planos, gráficas, prototipos, manuales, programas, análisis estadísticos, modelos matemáticos, simulaciones, normativas, regulaciones y restricciones, entre otros. Solo para proyectos que por su naturaleza lo requieran: estudio de mercado, estudio técnico y estudio económico.	45
	Conclusiones, recomendaciones y experiencia profesional adquirida	15
	Competencias desarrolladas y/o aplicadas	3
	Fuentes de información	2
Calificación total		100

Observaciones:

		28/07/2022
Jorge Enrique Luna Taylor Nombre y firma del asesor interno	SECRETARÍA DE EDUCACIÓN PÚBLICA TECNOLÓGICO NACIONAL DE MÉXICO INSTITUTO TECNOLÓGICO DE LA PAZ DEPARTAMENTO DE SISTEMAS Sello de la institución	Fecha de Evaluación

ITLP-AC-PO-006-09

Toda copia en PAPEL es un "Documento No Controlado" a excepción del original

Rev. 01