Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

**MÁSTER UNIVERSITARIO EN INGENIERÍA DE CONTROL, AUTOMATIZACIÓN Y ROBÓTICA**

# TRABAJO FIN DE MÁSTER

## *DESARROLLO DE ALGORITMOS DE APRENDIZAJE PROFUNDO BASADOS EN VISIÓN ARTIFICIAL PARA VEHÍCULOS AUTÓNOMOS DE INTERIORES*

**Estudiante** *Azurmendi Marquinez, Iker*
**Director/Directora** *Zulueta Guerrero, Ekaitz*
**Departamento** *Ingeniería de Sistemas y Automática*
**Curso académico** *2021-2022*

*Bilbao, 20 de junio de 2022*

# Resumen

El aprendizaje profundo (Deep Learning, DL) ha supuesto un importante avance para la industria de los vehículos autónomos. El desarrollo de las Redes Neuronales Profundas (DNN), y en particular de las Redes Neuronales Convolucionales (CNN), ha permitido mejorar las técnicas basadas en visión por ordenador, haciendo más útil la información obtenida de las cámaras. Igualmente, la visión destaca entre los demás sentidos por su riqueza y practicidad, además de ser barata y fácil de usar. Por esta razón, recientemente, se han realizado estudios sobre el uso de técnicas de aprendizaje profundo basadas en imágenes para controlar sistemas de conducción autónoma. Sin embargo, aunque, la conducción autónoma puede implicar que se refiera a la conducción automatizada de vehículos de exteriores, no hay que olvidar que existe la navegación autónoma de interiores. En este estudio, se propone un sistema de navegación completamente autónoma para vehículos de interior empleando técnicas de visión artificial y aprendizaje profundo. El algoritmo de navegación está formado por los módulos de seguimiento de trayectorias, detección de obstáculos y localización del vehículo. Por consiguiente, el robot debe ser capaz de seguir una trayectoria con un control automático de la dirección y la velocidad, a la vez que detecta obstáculos y se localiza en el entorno de navegación. Por último, se evalúan los tres algoritmos para determinar su viabilidad para ser implementados en un Vehículo de Guiado Automático (AGV) industrial.



Palabras Clave: Visión Artificial, Aprendizaje Profundo, Red Neuronal Convolucional, Navegación autónoma de interiores, Seguimiento de líneas, Detección de obstáculos, Seguimiento del vehículo.



# Abstract

Deep Learning (DL) has provided a significant breakthrough for the autonomous vehicle industry. The development of Deep Neural Networks (DNN), and particularly Convolutional Neural Network (CNNs), have enabled the improvement of computer vision-based techniques, making the information gathered from cameras more useful. Vision stands out among the other senses because of its richness and practicality, as well as its cheapness and simplicity of use. For this reason, recently, studies have been carried out on the use of image-based deep learning to control autonomous driving systems. However, while autonomous driving may imply that it refers to the automated driving of outdoor vehicles, it should not be forgotten that autonomous indoor navigation exists. This study proposes a fully autonomous navigation system for indoor vehicles using computer vision and deep learning techniques. The navigation algorithm consists of the path following, obstacle detection and vehicle localization modules. Therefore, the robot must be able to follow a trajectory with automatic control of direction and speed, while it detects obstacles and locates itself in the navigation environment. Furthermore, the three algorithms will be evaluated to determine their feasibility for implementation in an industrial Automated Guided Vehicle (AGV).



Keywords: Computer Vision, Deep Learning, Convolutional Neural Network, Autonomous Indoor Navigation, Line following, Obstacle Detection, Vehicle Tracking.

# Laburpena

Ikaskuntza Sakona (Deep Learning, DL) aurrerapen garrantzitsua izan da ibilgailu autonomoen industriarentzat. Sare Neuronal Sakonak (DNN) eta, bereziki, Sare Neuronal Konboluzionalak (CNN), garatzeak ahalbidetu dute ordenagailu bidezko ikusmenean oinarritutako teknikak hobetzea, kameretatik lortutako informazioa baliagarriagoa bihurtuz. Ikusmena, merkea eta erabiltzeko erraza izateaz gain, beste zentzumenen artean, bere aberastasun eta praktikotasunagatik nabarmentzen da. Horregatik, azkenaldian, ikaskuntza sakonaren erabilerari buruzko azterlanak egin dira, irudietan oinarrituta, gidatze autonomoko sistemak kontrolatzeko. Hala ere, gidatze autonomoak kanpoko ibilgailuen gidatze automatizatuari erreferentzia egitea ekar dezakeen arren, ez da ahaztu behar barneko nabigazio autonomoa ere dagoela. Ikerketa honetan, barruko ibilgailuentzako nabigazio sistema guztiz autonomo bat proposatzen da, ikusmen automatikoa eta ikaskuntza sakoneko teknikak erabiliz. Nabigazio algoritmoa, ibilbidearen jarraipena egiteko, oztopoak detektatzeko eta ibilgailuak mapan kokatzeko metodo ezberdinek osatuko dute. Ondorioz, robotak gai izan behar du norabidea eta abiadura automatikoki kontrolatuz ibilbide bat egiteko, aldi berean, oztopoak hautemanez eta nabigazio-ingurunean kokatuz. Gainera, hiru algoritmoak ebaluatuko dira, Ibilgailu Gidatu Automatiko (AGV) industrial batean inplementatzeko duten bideragarritasuna zehazteko.

Hitz gakoak: Ikusmen Artifiziala, Ikaskuntza Sakona, Sare Neuronal Konboluzionala, Barne Nabigazio Autonomoa, Lerroen jarraipena, Oztopoen detekzioa, Ibilgailuen jarraipena.

# Índice

# Lista de tablas

# Lista de ilustraciones

# Acrónimos

| | | |
|---|---|---|
| **IA** | Artificial intelligence | Inteligencia Artificial |
| **ML** | Machine Learning | Aprendizaje Automático |
| **DL** | Deep Learning | Aprendizaje Profundo |
| **NN** | Neural Network | Red Neuronal |
| **ANN** | Artificial Neural Network | Red Neuronal Artificial |
| **CNN** | Convolutional Neural Network | Red Neuronal Convolucional |
| **DNN** | Deep Neural Network | Red Neuronal Profunda |
| **RNN** | Recurrent Neural Network | Red Neuronal Recurrente |
| **FFNN** | Feed Forward Neural Network | Red Neuronal prealimentada |
| **BN** | Batch Normalization | Normalización Por Lotes |
| **ILSVRC** | Large Scale Visual Recognition Challenge | Reto de reconocimiento visual a gran escala |
| **VGG** | Visual Geometry Group | Grupo de geometría visual |
| **AV** | Autonomous Vehicles | Vehículos Autónomos |
| **GPS** | Global Positioning System | Sistema de Posicionamiento Global |
| **LiDAR** | Light Detection and Ranging | Detección y alcance de la luz |
| **AGV** | Automated Guided Vehicle | Vehículo de guiado automático |
| **UAV** | Unmanned Aerial Vehicle | Vehículo Aéreo No Tripulado |
| **ARM** | Autonomous Mobile Robots | Robots Móviles Autónomos |
| **SLAM** | Simultaneous Localization and Mapping | Localización y Mapeo Simultáneo |
| **RCNN** | Regions with Convolutional Neural Networks | Regiones con redes neuronales convolucionales |
| **YOLO** | You Only Look Once | Sólo se mira una vez |
| **SMC** | Sliding Mode Control | Control en modo deslizante |
| **GNSS** | Global Navigation Satellite System | Sistema global de navegación por satélite |

| | | |
|---|---|---|
| **VIO** | Visual Inertial Odometry | Odometría visual-inercial |
| **EKF** | Extended Kalman Filter | Filtro de Kalman Extendido |
| **AR** | Augmented Reality | Realidad aumentada |
| **TAM** | Tracking and Mapping | Mapeo y seguimiento |
| **ADS** | Automation Device Specification | Especificación del dispositivo de automatización |
| **PN** | Petri Nets | Redes Petri |
| **RFID** | Radio Frequency Identification | Identificación por radiofrecuencia |
| **HW** | Hardware | Hardware |
| **FC** | Fully Connected | Totalmente conectadas |
| **ResNet** | Residual Network | Red Residual |

# Introducción y contexto 1

En los últimos años, la Inteligencia Artificial (IA) está experimentando un crecimiento monumental en la reducción de la brecha entre las capacidades de los seres humanos y las máquinas. Tanto es así que la IA ha evolucionado hasta convertirse en una poderosa herramienta que permite a las máquinas pensar y actuar como seres humanos. Por esta razón, tanto los investigadores como los entusiastas trabajan en numerosos aspectos de este campo para conseguir cosas increíbles. Uno de esos ámbitos es la visión por ordenador [1].

El objetivo de este campo es permitir a las máquinas ver el mundo como lo hacen los humanos, percibirlo de manera similar e incluso utilizar el conocimiento para una multitud de tareas como el reconocimiento de imágenes y vídeos, el análisis y la clasificación de imágenes, la recreación de medios, los sistemas de recomendación o el procesamiento del lenguaje natural, entre otros. Los avances en visión por ordenador se han construido y perfeccionado con el tiempo. Sin embargo, hoy en día, las técnicas de Aprendizaje Profundo (DL) son las más utilizadas para la visión artificial, principalmente sobre un algoritmo en particular: la Red Neuronal Convolucional (CNN), debido a que proporcionan una mejora espectacular del rendimiento en comparación con los algoritmos tradicionales de procesamiento de imágenes.

Por otro lado, la última década ha sido testigo de un progreso cada vez más rápido en la tecnología de los vehículos de autoconducción, respaldado principalmente por los avances en las nuevas técnicas de visión por ordenador. Sin embargo, aunque, la conducción autónoma puede implicar que se refiera a la conducción automatizada de vehículos de exteriores, no hay que olvidar que existe la navegación autónoma de interiores. A pesar de que esta última no recibe tanta atención como la primera, permite estudiar, comprender y diseñar técnicas de navegación a aficionados con recursos limitados.

En este trabajo se pretende desarrollar un sistema de conducción autónoma para vehículos de interiores mediante el uso exclusivo de algoritmos de visión artificial. Para ello, se tendrán en cuenta tanto algoritmos más complejos basados en aprendizaje profundo como métodos más sencillos de código abierto.

El proyecto se ha desarrollado en la Escuela de Ingeniería de Vitoria-Gasteiz junto con el director del proyecto, el Dr. Ekaitz Zulueta Guerrero. El vehículo empleado para llevar a cabo las pruebas es un prototipo de Vehículo de Guiado Automático (AGV) industrial, el cual ha sido diseñado y desarrollado por alumnos de la escuela a lo largo de los últimos años. El AGV empleado se puede observar en la **Figura 1**.

De acuerdo con 6river los AGVs [2] “son sistemas de manipulación de materiales o portadores de carga que se desplazan de forma autónoma por un almacén, un centro de distribución o una

planta de fabricación, sin un operario o conductor a bordo". Es decir, un AGV es un vehículo que transporta productos de A hasta B sin necesidad de un operador humano. En un sentido más amplio, los AGV también incluyen sistemas que se utilizan para tareas de servicio, por ejemplo, manipulación, supervisión o limpieza.

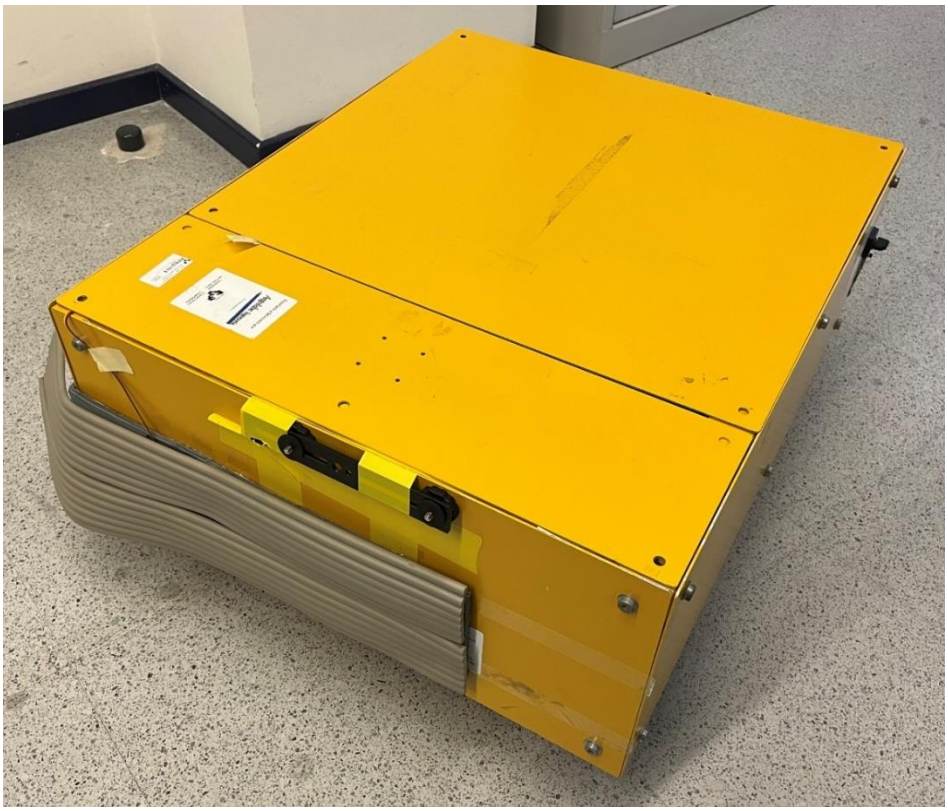


**Figura 1.** AGV empleado en el proyecto.

Para el desarrollo de este proyecto se han empleado los programas Matlab, para desarrollar todos los algoritmos de aprendizaje profundo y visión artificial, así como TwinCAT v3 para mover el vehículo, debido a que las ruedas del AGV se controlan desde un autómata de Beckhoff. Asimismo, se han adquirido dos webcams, se han impreso marcadores fiduciarios para realizar el algoritmo de localización y se ha dibujado una línea amarilla en el suelo con cinta adhesiva como trayectoria que debe seguir el AGV.

Por último, se presenta la distribución de los apartados del trabajo de fin de máster. En primer lugar, en el apartado de *Alcance y Objetivos* se habla del problema que se quiere resolver y las metas que se deben alcanzar para obtener un resultado satisfactorio. En segundo lugar, se realiza un análisis del *Estado del arte*, en el que, tras introducir brevemente el aprendizaje profundo y las redes neuronales convolucionales, se hace un estudio de los antecedentes bibliográficos de los últimos años acerca de los vehículos autónomos de interior. Después, en el apartado *Desarrollo de la solución* se explican los algoritmos que se han realizado en este trabajo y las técnicas empleadas para su desarrollo e implementación en un vehículo autónomo de interior. Asimismo, los resultados obtenidos y ejemplos reales de la puesta en marcha de los algoritmos se expondrán en la sección de *Análisis de los resultados*. Finalmente, los capítulos de *Conclusiones* y *Bibliografía* darán pasos a los *Anexos*, en los que, por ejemplo, se mostrará el código empleado y como usarlo.

# Alcance y objetivos 2

## 2.1 Alcance

El alcance de este proyecto es el desarrollo, pruebas e implementación de un algoritmo de navegación para vehículos autónomos de interior. La idea es lograr un sistema de navegación autónoma que sea flexible (que pueda emplearse en diferentes entornos y vehículos) y que permita guiar vehículos autónomos de manera fiable en un entorno industrial.

La solución propuesta es un algoritmo de navegación que puede dividirse en tres partes principales: el seguimiento de trayectorias, la evasión de obstáculos y la localización del vehículo.

El seguimiento de trayectorias es el aspecto más importante del algoritmo de navegación, ya que permite mover el vehículo por una trayectoria definida. Esta trayectoria consiste en una línea hecha con cinta adhesiva que tiene un color diferente al del suelo, y que puede ser modificada fácilmente por el usuario. Los otros dos algoritmos dan un valor añadido al sistema de navegación: la detección de obstáculos ofrece un nivel de seguridad extra al vehículo y a su entorno (personas, objetos, etc.), mientras que la localización permite seguir y mapear la posición del vehículo en tiempo real.

Este sistema pretende ser implementado en un AGV disponible en la Escuela de Ingeniería de Vitoria-Gasteiz para que pueda ser empleado en un futuro en un entorno industrial.

Los sistemas de AGV tienen sus comienzos en la década de 1950. Este primer vehículo precisaba de cable que, enterrado en el suelo de la fábrica, creaba un campo magnético que servía de guía al vehículo [3]. Debido a su fiabilidad y rendimiento, poco a poco, se han ido incorporando al mundo industrial a la vez que han crecido sus aplicaciones y sofisticación.

En los últimos años, además, se han desarrollado robots AGV dotados de tecnología de IA [4]. Dentro este campo, la visión por ordenador está desempeñando un papel fundamental en el desarrollo de modelos de aprendizaje automático, y aprendizaje profundo, basados en la percepción visual, no solo para AGVs sino para todo tipo de vehículos autónomos [5]. Por ejemplo, la visión por ordenador se está empleando actualmente en este ámbito para tareas como el reconocimiento de objetos, la detección de carril o la creación de mapas, entre otros.

Por esta razón, se desean estudiar estas nuevas técnicas de inteligencia artificial (sobre todo de aprendizaje profundo) y visión artificial para tratar de desarrollar un sistema de navegación completamente autónoma de bajo coste, que como se ha comentado anteriormente, sea flexible, seguro y funcione correctamente. Para ello, se considera que alguno de los aspectos más importantes para tener en cuenta, son la recogida de suficientes datos de entrenamiento y

su etiquetado, para obtener unos resultados precisos, así como lograr que estos algoritmos se ejecuten lo suficientemente rápido en el hardware disponible.

## 2.2 Objetivos

El objetivo principal de este proyecto es el desarrollo de un sistema de navegación para un vehículo autónomo de interior empleando exclusivamente algoritmos de visión artificial. Para lograr este objetivo, se ha considerado necesario desarrollar los siguientes objetivos parciales:

a) Un sistema de seguimiento de líneas que permita al vehículo moverse por una trayectoria predefinida por el usuario. Para que el algoritmo "sigue líneas" funcione correctamente debe realizar las siguientes tareas:
    a. Detectar el carril por donde tiene que circular el vehículo.
    b. Obtener un punto de referencia al que debe dirigirse el vehículo.
    c. Calcular unas consignas de control para el movimiento de las ruedas.
b) Un algoritmo de detección de personas que sirva como un elemento de seguridad para robots móviles. Su funcionamiento es el siguiente: en caso de que el vehículo detecte un obstáculo, se detendrá y esperará a que este desaparezca.
c) Un método de seguimiento del vehículo que, por un lado, permita al usuario observar donde está el vehículo en cada momento, y, por otro lado, ayude al vehículo a localizarse para, por ejemplo, realizar una tarea específica llegado a cierto lugar. El algoritmo de localización consistirá en:
    a. Identificar un punto de referencia en el entorno.
    b. Estimar la posición del vehículo.
    c. Mapear el recorrido del vehículo.

Por otro lado, para lograr los objetivos marcados anteriormente, se ha considerado necesario realizar las siguientes tareas:

1) Analizar y comprender el AGV, y el código de control correspondiente, disponible en la Escuela de Ingeniería de Vitoria-Gasteiz para identificar sus posibilidades y limitaciones.
2) Realizar un estudio de los antecedentes bibliográficos de los últimos años acerca de sistemas de navegación autónomos para vehículos de interior.
3) Desarrollar un sistema de navegación que permita al AGV moverse de forma completamente autónoma y que sea flexible para diferentes entornos y vehículos.
4) Realizar las pruebas necesarias para determinar si el sistema desarrollado cumple o no con los objetivos propuestos.

# Estado del arte 3

## 3.1 Deep Learning

El aprendizaje automático (ML) y la inteligencia artificial (AI) se están convirtiendo en técnicas dominantes de resolución de problemas en muchos ámbitos de la industria y la investigación, sobre todo por el reciente éxito del aprendizaje profundo (DL). El mundo crece a un ritmo exponencial y también lo hace el tamaño de los datos creados, capturados, copiados y consumidos en todo el mundo. Estos datos son cada vez más significativos y relevantes desde el punto de vista contextual, lo que permite abrir nuevos caminos para este tipo de técnicas [6].

De acuerdo con Mitchell (1997) [7], el ML es la ciencia que "se ocupa de la cuestión de cómo construir programas informáticos que mejoren automáticamente con la experiencia". Así pues, tanto la IA como el ML tratan de construir programas informáticos inteligentes, y el DL, al ser un caso de ML, no es una excepción (véase **Figura 2**) [6].

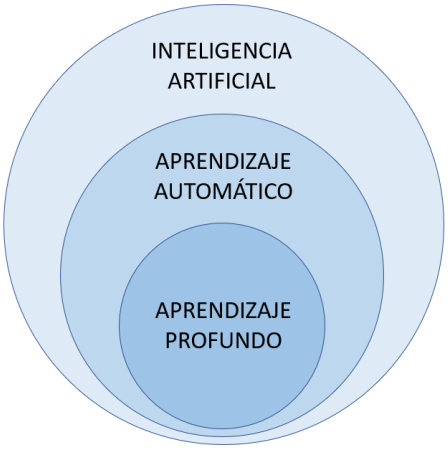



**Figura 2.** Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo.

El DL [8], [9] es un tipo de ML que es esencialmente una red neuronal (NN) con tres o más capas. Estas redes neuronales intentan simular el comportamiento del cerebro humano permitiéndole "aprender" de grandes cantidades de datos. En los últimos años, el aprendizaje profundo ha logrado avances significativos en dominios como el reconocimiento o clasificación de imágenes y videos [10], los sistemas autónomos y robótica [11], el análisis de texto y el procesamiento del lenguaje natural [12] o la medicina [13], entre muchos otros [14].

El DL, en lugar de organizar datos para que se ejecuten a través de ecuaciones predefinidas, configura parámetros básicos acerca de los datos y entrena al ordenador para que aprenda por su cuenta reconociendo patrones mediante el uso de muchas capas de procesamiento. En otras palabras, se sustituye la complejidad de escribir y programar algoritmos, que cubren todas las

eventualidades, por la dificultad de encontrar un esquema general apropiado para los algoritmos. Además, será necesario procesar adecuadamente los datos, ya que, para el correcto entrenamiento de un modelo de DL, se requieren muchos datos y, normalmente, grandes conjuntos de entrenamiento etiquetados.

Hay muchos tipos de modelos de DL, entre los que la red neuronal profunda (DNN) [15], la red neuronal convolucional (CNN) [16] y la red neuronal recurrente (RNN) [17], [18] son tres estructuras básicas. La DNN es un tipo de red neuronal prealimentada (FFNN) con múltiples capas ocultas, en la que los datos fluyen desde la capa de entrada a la de salida sin necesidad de que se produzca un bucle de retorno. La CNN, que suele aplicarse al análisis de imágenes, utiliza una variante de los perceptrones multicapa y está diseñada para requerir un preprocesamiento mínimo. Finalmente, la RNN es un tipo de ANN que tiene una memoria interna, lo que le permite exhibir comportamientos temporales. Se utiliza con datos secuenciales o series temporales.

La historia de las redes neuronales se remonta se remonta a la década de 1940 [19], y la intención original era simular el sistema cerebral humano para resolver problemas generales de aprendizaje de una manera principista. Se popularizó en las décadas de 1980 y de 1990 con la propuesta del algoritmo de retropropagación de Rumelhart et al. [17]. Sin embargo, debido al sobreajuste del entrenamiento, la falta de datos de entrenamiento a gran escala, la limitada capacidad de computación y la insignificancia de su rendimiento en comparación con otras herramientas de aprendizaje automático, las redes neuronales pasaron de moda a principios de la década del 2000 [20].

El aprendizaje profundo se ha hecho popular desde 2006 [21], con un gran avance en el reconocimiento del habla y la clasificación de imágenes. La recuperación del aprendizaje profundo puede atribuirse a los siguientes factores [20]:

1) La aparición de grandes cantidades de datos de entrenamiento anotados a gran escala y de acceso abierto.
2) El rápido desarrollo de sistemas de cálculo paralelo de alto rendimiento, como los clústeres de GPU, y el progreso de hardware específico para algoritmos de IA (ASIC).
3) Los avances significativos en el diseño de estructuras de red y estrategias de entrenamiento. Por ejemplo, con el preentrenamiento no supervisado y las capas de guiado por autoencoder [22] o la máquina de Boltzmann restringida [23], se proporciona una buena inicialización. Con el dropout y el aumento de datos, se ha mejorado el problema del sobreajuste en el entrenamiento [24], y con la normalización por lotes (BN), el entrenamiento de las redes neuronales se vuelve bastante eficiente [25].
4) La mejora en el rendimiento y la reducción del tiempo de entrenamiento de los modelos de DL, debido al auge de arquitecturas de redes preentrenadas como las de la **Tabla 1**.

El DL consta de cuatro fases: a) la creación de un conjunto de datos (si no existe una base de datos de acceso abierto disponible), que permita caracterizar el problema a resolver y así obtener resultados precisos; b) el entrenamiento, en el que los datos de entrada se emplean para calcular los parámetros del modelo; c) la fase de evaluación, en la que el modelo entrenado da un valor a diferentes muestras de entrada y se evalúa el algoritmo; y d) el despliegue y la ejecución del modelo para una aplicación específica, por ejemplo, en un microcontrolador o en una GPU. En la **Figura 3** se muestra el flujo de trabajo descrito para algoritmos de DL.

**Tabla 1.** Resumen CNN pre-entrenadas.

| CNN | Año | Arquitectura | Información |
|---|---|---|---|
| LeNet5 [26] | 1998 | 3 capas convoluciones + 2 capas totalmente conectadas (FC) | Lenet-5, propuesto por Yann LeCun et al. en el año 1998, fue uno de los primeros modelos preformados. Utilizaron esta arquitectura para reconocer los caracteres escritos a mano e impresos a máquina. |
| AlexNet [27] | 2012 | 5 capas convolucionales + 3 capas totalmente conectadas | Diseñada por Alex Krizhevsky en colaboración con Ilya Sutskever y Geoffrey Hinton, AlexNet compitió y ganó el reto de reconocimiento visual a gran escala de ImageNet de 2012. |
| ZF Net [28] | 2013 | 3 capas convolucionales + 1 capa totalmente conectada | La CNN ganadora del ILSVRC 2013 fue la red de Matthew Zeiler y Rob Fergus. Se conoció como ZFNet (abreviatura de Zeiler & Fergus Net). Se trata de una mejora de AlexNet mediante el ajuste de los hiperparámetros de la arquitectura, en particular ampliando el tamaño de las capas convolucionales intermedias y reduciendo el tamaño de la zancada y del filtro en la primera capa. |
| Overfeat [29] | 2014 | La arquitectura para la clasificación es similar a la de AlexNet con algunas mejoras. 5 capas convolucionales + 3 capas totalmente conectadas (aunque hay modelos más extendidos) | Overfeat es un modelo pionero que integra las tareas de detección, localización y clasificación de objetos en una red neuronal convolucional. La idea principal es realizar la clasificación de la imagen en diferentes ubicaciones en regiones de múltiples escalas de la imagen en forma de ventana deslizante, y predecir las ubicaciones de los cuadros delimitadores con un regresor entrenado sobre las mismas capas de convolución. |
| Inception [30] | V1 2014<br>V2 2015<br>V3 2015<br>V4 2016 | 2 capas convolucionales + 9 capas Inception (según versión) | La red Inception fue un hito importante en el desarrollo de los clasificadores CNN. Era compleja (con una fuerte ingeniería): utilizaba muchos trucos para aumentar el rendimiento, tanto en términos de velocidad como de precisión. Su constante evolución llevó a la creación de varias versiones de la red. |
| ResNet [31] | 2015 | (4 + 1) capas convolucionales en cada módulo. Acompañadas de 1 capa totalmente conectada. En total hacen 18 capas convolucionales para la ResNet 18. | La red residual (ResNet) es una arquitectura de red neuronal convolucional (CNN) que superó el problema del "gradiente de fuga", haciendo posible la construcción de redes con hasta miles de capas convolucionales, que superan a las redes menos profundas. ResNet puede contener un gran número de capas convolucionales, normalmente entre 18-152, pero admite hasta miles de capas. Existen variantes más recientes, denominadas ResNext y DenseNet, que son las más utilizadas en la actualidad. |
| VGG [32] | 2014 | 13 capas convolucionales + 3 capas totalmente conectadas | VGG son las siglas de Visual Geometry Group (Grupo de Geometría Visual). Se trata de una arquitectura de CNN profunda estándar con múltiples capas. Existen también las redes VGG-16 o VGG-19. Por otro lado, la arquitectura VGG es la base de los innovadores modelos de reconocimiento de objetos. Desarrollada como red neuronal profunda, la VGGNet también supera las líneas de base en muchas tareas y conjuntos de datos más allá de ImageNet. Además, sigue siendo una de las arquitecturas de reconocimiento de imágenes más populares. |
| SqueezeNet [33] | 2016 | 2 capas de convolución + 9 módulos de fuego (reducir + expandir) | Al diseñar SqueezeNet, el objetivo de los autores era crear una red neuronal más pequeña con menos parámetros que pudiera caber fácilmente en la memoria del ordenador y que pudiera transmitirse de manera más sencilla a través de una red informática. Tiene una precisión del nivel de AlexNet con 50 veces menos parámetros y un tamaño de modelo de < 0.5 MB. |

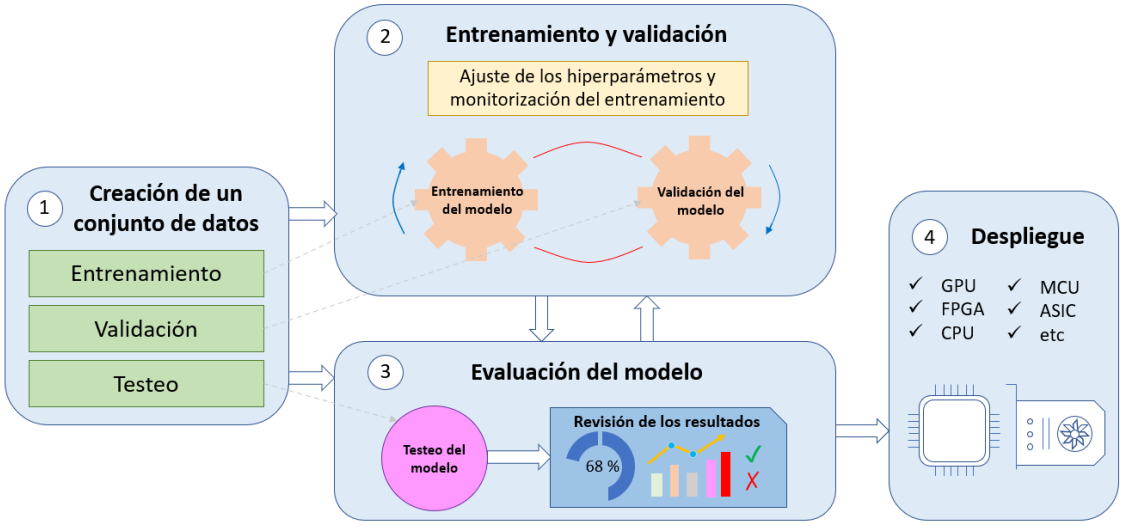



**Figura 3.** Flujo de trabajo del Deep Learning.

### 3.1.1 Importancia de los datos de entrenamiento

En el aprendizaje profundo, como en aprendizaje automático, uno de los aspectos que más hay que tener en cuenta es el tipo de datos que se dan al modelo. Si se tienen más datos, hay más posibilidades de que un algoritmo de aprendizaje automático los entienda y dé predicciones precisas a datos no vistos con anterioridad.

Como hacen referencia Li Liu et al. [34], los conjuntos de datos han desempeñado un papel fundamental a lo largo de la historia de la investigación sobre el aprendizaje profundo, no solo como terreno común para medir y comparar el rendimiento de los algoritmos que compiten entre sí, sino también para impulsar el campo hacia problemas cada vez más complejos y desafiantes. Por ejemplo, el acceso a un gran número de imágenes en Internet permite construir conjuntos de datos extensos para diferentes ámbitos, lo que posibilita un rendimiento sin precedentes para los algoritmos de visión artificial.

Puede haber muchas formas de datos que pueden utilizarse para el aprendizaje automático. Sin embargo, algunos de los principales tipos de datos que se dan a este tipo de algoritmos para realizar predicciones son: datos categóricos, datos numéricos, datos de series temporales y datos de texto.

El primer paso en el proceso de aprendizaje automático, y por consiguiente del aprendizaje profundo es la preparación de los datos. La **Figura 4** presenta el flujo de trabajo de la preparación de datos propuesto por Xin He et al. [35], que se puede resumir en tres fases: recogida de datos, limpieza de datos y aumento de datos. En primer lugar, la recogida de datos es un paso necesario para construir un nuevo conjunto de datos o ampliar el existente. El proceso de limpieza de datos se utiliza para filtrar los datos ruidosos, de modo que no se comprometa el entrenamiento del modelo posterior. Por último, el aumento de los datos desempeña un papel importante en la mejora de la solidez y el rendimiento del modelo.

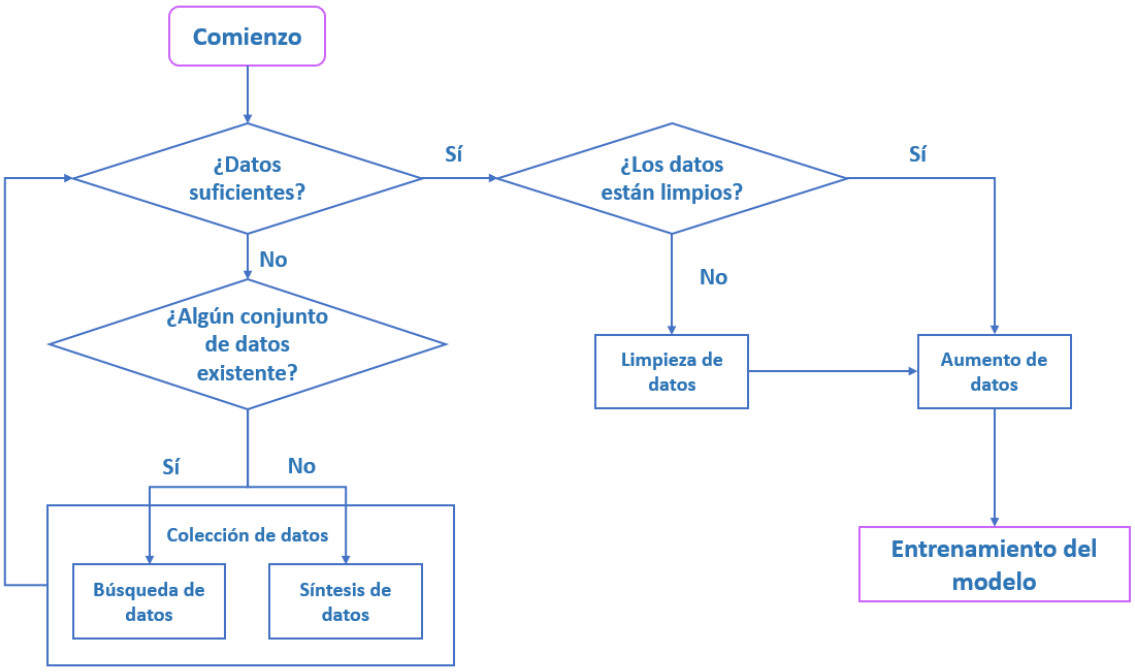



**Figura 4.** Diagrama de flujo para la preparación de datos.

Por otro lado, en la **Tabla 2** se hace un resumen de algunas de las bases de datos más conocidas y empleadas hoy en día para llevar a cabo tareas de aprendizaje profundo.

**Tabla 2.** Bases de datos más conocidas.

| Dataset | Número de imágenes | Categorías | Etiquetado | Página web |
|---|---|---|---|---|
| ImageNet [36] | + 14 millones | 21.841 | Etiquetado categórico y etiquetado de objetos. | https://image-net.org/download |
| MS COCO [37] | + 320.000 | 91 | Etiquetado para detección, segmentación y subtitulación de objetos. | https://cocodataset.org/#home |
| Places [38] | + 10 millones | 434 | Etiquetado para el reconocimiento de escenas. | http://places2.csail.mit.edu/ |
| Open Images [39] | + 9 millones | + 6000 | Etiquetado para la detección de objetos, la segmentación de instancias de objetos y la detección de relaciones visuales. | https://storage.googleapis.com/openimages/web/index.html |
| Ade20k [40] | + 20.000 | 150 | Etiquetado de objetos y partes de objetos a nivel de píxel (segmentación semántica). | http://groups.csail.mit.edu/vision/datasets/ADE20K/ |

### 3.1.2 Introducción a las Redes Neuronales Convolucionales

Una red neuronal convolucional (ConvNet/CNN) es un algoritmo de aprendizaje profundo que puede tomar una imagen de entrada, asignar importancia (pesos y sesgos que se pueden aprender) a varios aspectos/objetos de la imagen y ser capaz de diferenciar una imagen de otra.

Las CNN se distinguen de otras redes neuronales por su rendimiento superior con entradas de señales de imagen, habla o audio. Las CNN son una construcción matemática que suele estar compuesta por tres tipos de capas (o bloques de construcción): convolución, agrupación y capas totalmente conectadas.

La convolución es una operación matemática que permite fusionar dos conjuntos de información. La capa de agrupación recibe el resultado de una capa convolucional y lo comprime, y las capas totalmente conectadas son aquellas en las que todas las entradas de una capa están conectadas a cada unidad de activación de la capa siguiente. Esta capa realiza la tarea de clasificación en base a las características extraídas a través de las capas anteriores y sus diferentes filtros.

El proceso de optimización de los parámetros del algoritmo de DL se denomina entrenamiento, el cual se realiza para minimizar la diferencia entre las salidas predichas y las etiquetas a través de algún algoritmo de optimización como el de retropropagación y descenso de gradiente, entre otros [41]. Un ejemplo de una arquitectura simple de CNN se puede observar en la **Figura 5**.

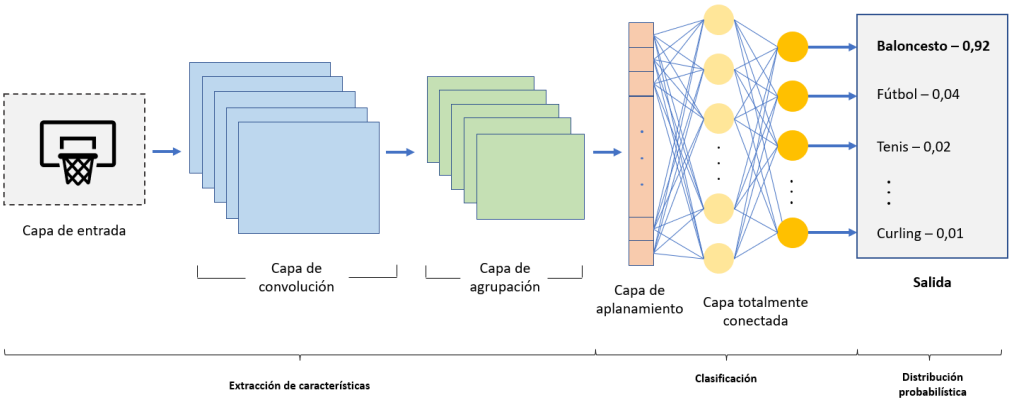



**Figura 5.** Ejemplo de una arquitectura de CNN.

Por otro lado, las CNN son algoritmos de aprendizaje supervisado. El aprendizaje supervisado trabaja con datos etiquetados, es decir, datos para los que se conoce la respuesta esperada. En concreto, el aprendizaje supervisado se utiliza como mecanismo de predicción, en el que se aprende sobre una parte de los datos (también conocida como conjunto de entrenamiento), otra parte se utiliza para evaluar el modelo durante el entrenamiento (validación), y el resto se utiliza para determinar la precisión y la eficacia en la predicción (testeo) [14]. Las dos principales tareas de aplicación en el aprendizaje supervisado, y que se emplearán en este trabajo son la clasificación y la regresión.

1) CLASIFICACIÓN

La clasificación es el proceso de encontrar o descubrir un modelo o función que ayude a separar los datos en múltiples clases categóricas, es decir, valores discretos. Esto puede adoptar la forma

de clasificación binaria de sólo dos clases (0 o 1) o clasificación multiclase que da lugar a una clase entre un conjunto de tres o más clases totales (rojo, verde, azul, etc.).

2) REGRESIÓN

La regresión es el proceso de encontrar un modelo que prediga un valor continuo basado en sus variables de entrada, en lugar de predecir clases o valores discretos. Su objetivo es establecer un método para la relación entre un cierto número de características y una o varias variables objetivo de valor continuo. Los algoritmos de regresión ayudan a predecir variables continuas como los precios de la vivienda, las tendencias del mercado o los patrones climáticos, entre otros.

En la **Figura 6** se puede observar un resumen de las dos principales tareas del aprendizaje supervisado comentadas.

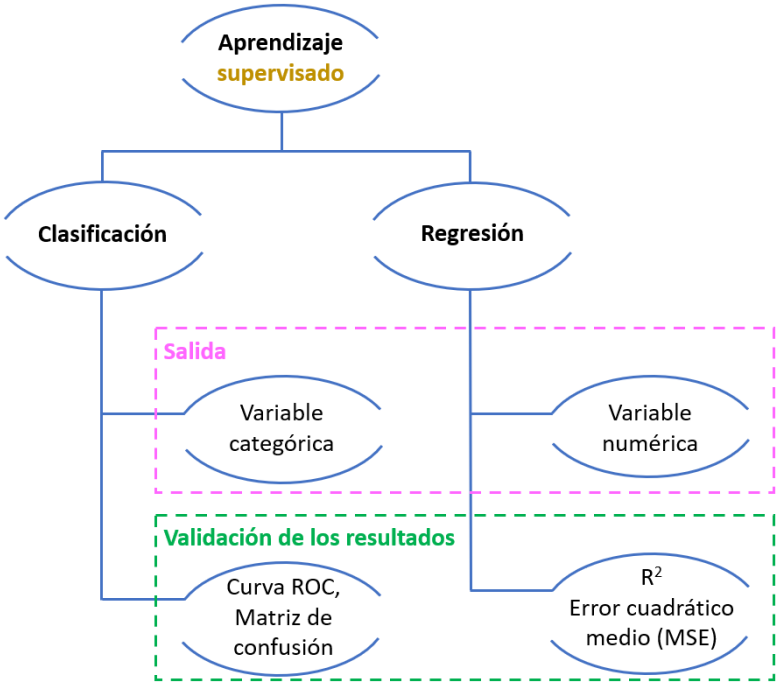



**Figura 6.** Resumen de las principales tareas del aprendizaje supervisado.

### 3.1.3 Introducción al aprendizaje transferido

Durante el entrenamiento de una red neuronal se calculan los pesos óptimos para la clasificación, asociados a las conexiones entre neuronas. Esos pesos determinan el funcionamiento de la red. Después de entrenar la red, se pueden guardar los pesos, así como la arquitectura de la red. De esta manera, al usar la red en el futuro, no se tiene que volver a entrenar [42].

Por otra parte, cabe tener en cuenta que el entrenamiento de una red es largo y tedioso, y requiere una gran cantidad de recursos a nivel computacional, por ejemplo, cuando se trabaja con imágenes. Por este motivo, una forma de reducir el consumo de recursos es partir de una red preentrenada.

La reutilización de un modelo preentrenado en un nuevo problema se conoce como aprendizaje por transferencia en el aprendizaje automático. En el aprendizaje por transferencia, un modelo o algoritmo utiliza los conocimientos aprendidos en una tarea anterior para aumentar la predicción sobre una nueva tarea [43].

El conocimiento de un modelo de aprendizaje automático ya entrenado se transfiere a un problema diferente pero estrechamente relacionado mediante el aprendizaje por transferencia. Por ejemplo, si se ha entrenado un clasificador sencillo para predecir si una imagen contiene un gato, se puede utilizar el conocimiento del modelo entrenado para identificar otros objetos (en este caso otros animales), como un pingüino.

Con el aprendizaje por transferencia, básicamente se intenta utilizar lo que se ha aprendido en una tarea para entender mejor los conceptos de otra. Los pesos se trasladan automáticamente a una red que realiza la nueva "tarea A" desde una red que realizó la "tarea B".

Debido a la enorme cantidad de potencia de CPU que requiere, el aprendizaje por transferencia se aplica normalmente en tareas de visión por ordenador y de procesamiento del lenguaje natural, como el análisis de sentimientos o la clasificación de imágenes.

El aprendizaje por transferencia ofrece una serie de ventajas, entre las que se encuentran la reducción del tiempo de entrenamiento, la mejora del rendimiento de la red neuronal (en la mayoría de las circunstancias) y la ausencia de una gran cantidad de datos.

El aprendizaje por transferencia se utiliza generalmente para [44]:

- Ahorrar tiempo y recursos al tener que entrenar múltiples modelos de aprendizaje automático desde cero para completar tareas similares.
- Mejorar el rendimiento en áreas de aprendizaje automático que requieren grandes cantidades de recursos, como la categorización de imágenes o el procesamiento del lenguaje natural.
- Compensar la falta de datos de entrenamiento etiquetados para un algoritmo en concreto, mediante el uso de modelos preentrenados.

En resumen, el aprendizaje por transferencia se introdujo para superar las limitaciones de los modelos tradicionales de aprendizaje automático y aprendizaje profundo, permitiendo ahorrar tiempo de entrenamiento y mejorar el rendimiento de los nuevos modelos.

Por otro lado, los métodos de aprendizaje profundo han demostrado, en los últimos años, ser muy prometedores, no solo por ofrecer un excelente rendimiento en problemas de control complejos y no lineales, sino también por generalizar las reglas aprendidas previamente a nuevos escenarios. Por estas razones, el uso del aprendizaje profundo para el control de vehículos es cada vez más popular [45].

Al mismo tiempo, los vehículos autónomos (AV) han empezado a pasar de las condiciones de desarrollo y pruebas en los laboratorios a la conducción en la vía pública. Su implantación en nuestro entorno ofrece una disminución de los accidentes de tráfico y de las congestiones, así como una mejora de nuestra movilidad en las ciudades superpobladas [46].

## 3.2 Sistemas de navegación autónoma

La navegación autónoma se ha convertido en un tema candente en los últimos años [47], especialmente para los vehículos autónomos de carretera. Varios grupos de desarrollo, desde fabricantes de automóviles hasta empresas de tecnología e informática, están contribuyendo al transporte autónomo de carga y personas. Paralelamente, la robótica y la automatización están evolucionando a un ritmo creciente para sustituir el trabajo manual y mejorar la ejecución de tareas en casi todos los ámbitos. Por desgracia, la navegación autónoma de los robots móviles de interior no recibe tanta atención como la de los vehículos de carretera, especialmente en el caso de los robots que funcionan por visión [48].

Los robots móviles son dispositivos mecatrónicos que pueden moverse en un entorno físico y que se utilizan para asistir a los seres humanos en diversas actividades, ya sean peligrosas o no, repetitivas y perjudiciales. Esta categoría de robots, a través de sensores y actuadores, interactúan con el entorno para moverse y realizar sus funciones de forma autónoma. Normalmente, se utilizan en entornos interiores como oficinas, hospitales, líneas de producción o transporte de mercancías en la industria. En este último caso, destacan los vehículos autoguiados (AGV) [49].

Los AGV son máquinas capaces de moverse y realizar diversas tareas sin la intervención de los humanos. Los modelos más antiguos de AGV utilizaban marcas y puntos de referencia colocados en el entorno para guiar al robot [50]. Esto significa que había que invertir mucho tiempo y dinero en analizar y preparar el entorno para acomodar estos robots. Uno de los problemas de este sistema es que debe construirse y configurarse por completo para un entorno específico, y cualquier cambio en este entorno requerirá una reconfiguración y modificaciones adicionales en el sistema. Los modelos más recientes son más tolerantes en relación con estas condiciones, pero siguen requiriendo una configuración especializada según su aplicación [48]. Asimismo, los diferentes tipos de robots móviles autónomos (ARM), además de moverse por todo el entorno, deben, por ejemplo, ubicarse y planificar trayectorias entre posiciones.

Como comentan F. Mota et al. [49] un robot móvil típico está compuesto por los módulos de percepción, localización, navegación y movimiento. El ciclo de acciones de un robot móvil comienza con la extracción de datos del entorno real (módulo de percepción) a través de los sensores acoplados al robot como infrarrojos, ultrasonidos, cámaras o lectores RFID. Estos datos se procesan y se utilizan como parámetros de entrada de los algoritmos para que el robot pueda localizarse a sí mismo (módulo de localización). Una vez estimada la localización del robot, se compara con un mapa global y se determina la siguiente acción a realizar (módulo de cognición o movimiento). A continuación, el robot ejecuta movimientos y navega por el entorno hasta alcanzar la posición de destino (módulo de navegación).

Asimismo, Yuri D. V. Yasuda et al. [48] describen la navegación autónoma de un robot móvil como la ejecución de cuatro tareas principales: localización, mapeo, planificación de la trayectoria y locomoción. La localización y el mapeo suelen tratarse simultáneamente, ya que este tipo de solución proporciona mejores resultados en comparación con la resolución de estas de forma independiente [51]. Las soluciones más comúnmente propuestas se basan en el enfoque de Localización y Mapeo Simultáneo (SLAM), que ha tenido muchas implementaciones

diferentes a lo largo de los años. La planificación de la trayectoria depende del mapa del entorno y de la posición actual del robot, a la vez que requiere detalles como su tamaño y capacidad de movimiento. La locomoción incluye el control físico del robot y la detección de obstáculos, lo que le permite moverse con seguridad y eficacia por el entorno hacia su objetivo. Igualmente, comentan que otras aplicaciones más específicas pueden requerir que se resuelvan otras tareas además de estas cuatro.

Por ejemplo, Werner et al. [52], explican que la navegación requiere un sistema de posicionamiento global, la detección de puntos de referencia, la capacidad de seguir paredes y pasillos, evitar obstáculos o girar en cruces, así como decidir el camino a seguir cuando hay más de una opción disponible.

Además, dado un entorno parcialmente conocido, Siegwart y Nourbakhsh [53] presentan la navegación como la capacidad del robot móvil de desplazarse por el entorno hasta una o una serie de posiciones objetivo, basándose en sus conocimientos y en la información proporcionada por sus sensores, de la forma más fiable posible.

Por otro lado, cabe destacar que las técnicas de navegación en interiores requieren de algoritmia diferente a la de exteriores, ya que ciertas tecnologías como el GPS (Global Positioning System) no son utilizables, debido a la falta de recepción de las señales GPS en el interior de los edificios. Igualmente, para poder realizar cualquier movimiento de manera autónoma es necesario el uso de sensores, entre los que se pueden encontrar sensores por impacto, ultrasonidos, LiDAR (Light Detection And Ranging), visión, etc. [54]. El tratamiento de la información de los sensores puede ser múltiple, pero la última tendencia es emplear algoritmia basada en Deep Learning al ser una tecnología más flexible e inteligente [55].

Cuando se fusionan datos de varios sensores, es necesario integrar la posición y la orientación de cada sensor para reconocer el entorno. Por ejemplo, la estimación de la posición con una cámara monocular requiere parámetros internos (distancia focal, centro óptico, etc.), parámetros externos (posición y orientación) y coeficientes de distorsión. En este caso, la calibración de una cámara se refiere al proceso de estimación y corrección de estos parámetros [56]. Además, los sensores también deben calibrarse en función de sus múltiples posiciones de montaje.

A pesar de que existen muchos tipos de sensores que pueden utilizarse en un robot, como los de orientación, localización, medición de distancia o estimación del estado interno del robot, ninguno de ellos destaca como la visión. La visión predomina entre los demás sentidos por su riqueza y practicidad [48]. Una de las razones es que la visión proporciona un conjunto muy rico de información, especialmente si se compara con otros sentidos. Otra ventaja de la visión es que puede utilizarse para muchas otras aplicaciones, que pueden ser complementarias a la navegación o ejecutarse en paralelo para resolver una tarea independiente [57], y también es relativamente barata y fácil de usar.

Como se ha comentado anteriormente, la navegación autónoma consta de cuatro tareas principales. Estas cuatro tareas pueden resolverse con precisión utilizando sólo la visión. Sin embargo, aunque existen soluciones de navegación autónoma bien establecidas, hasta ahora no hay ningún sistema de navegación autónoma que se base únicamente en la visión, que sea adecuado para entornos dinámicos de interior y que haya triunfado plenamente [48].

Uno de los métodos de navegación más empleados hoy en día en la conducción autónoma es la detección de carriles, que, aunque es una aplicación de exteriores, también es útil para los interiores. El objetivo de esta tecnología es detectar las zonas de carril o las líneas de carril mediante una cámara. Para ello hay que tener claros tanto los primeros algoritmos de control, así como los actuales. Es por eso que Tang et al. [58] presentó una revisión sobre la detección de carriles, e incluso se plantean nuevas alternativas basadas en redes neuronales con el objetivo de visualizar el estado actual.

Un robot que sigue una línea es un sistema móvil autónomo que sigue la línea que tiene un color diferente al del fondo. El robot consigue el objetivo de seguir la línea mediante la retroalimentación, por ejemplo, de una cámara (como se verá más adelante en este trabajo). Los robots seguidores de líneas pueden utilizarse en muchas aplicaciones de logística industrial, como el transporte de material pesado y peligroso, el sector agrícola y los sistemas de gestión de inventarios de bibliotecas. Estos robots también son capaces de controlar a los pacientes en los hospitales y avisar a los médicos de síntomas peligrosos [59]. En la **Tabla 3** se hace una revisión bibliográfica de robots móviles seguidores de líneas.

**Tabla 3.** Trabajos relacionados de robots móviles seguidores de líneas.

| Ref. | Sensor | Descripción |
|---|---|---|
| [60] | Infrarrojos | Este artículo presenta una técnica para el seguimiento de líneas y la evasión de colisiones en los sistemas robóticos móviles, la cual se basa en el uso de sensores infrarrojos de bajo coste, e implica un nivel razonable de cálculos, por lo que puede utilizarse fácilmente en aplicaciones de control en tiempo real. |
| [61] | Cámara | En este trabajo un robot es configurado para seguir una trayectoria predefinida mientras mantiene la velocidad especificada por el usuario. Para ello se emplea un controlador de seguimiento de líneas basado en la lógica difusa que permite al robot seguir las trayectorias en el suelo del entorno de fabricación. |
| [62] | Infrarrojos | En este estudio, se diseña y aplica un control en modo deslizante (SMC) robusto y sin parpadeos para un robot que sigue una línea. El robot móvil está diseñado para detectar la trayectoria recta o curva con sus sensores infrarrojos montados en el robot. Por lo tanto, estos sensores infrarrojos proporcionan un flujo continuo de la trayectoria definida para guiar o dirigir los cambios en el robot. |
| [63] | Cámara | Este trabajo propone un enfoque difuso para el seguimiento de la trayectoria de un robot móvil no holonómico, basado en la información de una cámara frontal. La metodología propuesta se divide en tres etapas: 1) se obtiene la imagen de la cámara frontal y procesa la imagen para detectar y aislar la trayectoria deseada a seguir; 2) se estima la orientación de los diferentes tramos de la trayectoria; y 3) se diseña y simula un sistema difuso para controlar la dirección del robot móvil. |
| [64] | Infrarrojos | Este trabajo describe un robot de seguimiento de líneas utilizando Arduino. El sistema propuesto detecta el camino negro en el suelo y procede en su dirección. El objetivo es implementar el movimiento controlado del robot mediante el ajuste de los parámetros de control y así lograr un mejor rendimiento. |

Interpretar el entorno mediante la segmentación de la imagen puede ser otro claro ejemplo de navegación y así lo demostró Daniel Teso-Fz-Betoño [65] al hacer una transferencia de aprendizaje y un ajuste de capas, para que la red neuronal identificase la superficie navegable. De esta manera el vehículo de navegación en interiores no requiere de ruedas muy precisas, ya que su sistema basado en redes neuronales convolucionales le permiten interpretar el entorno.

En resumidas cuentas, la navegación en interiores requiere de algoritmia específica, y puede ser desarrollada partiendo del conocimiento de exteriores. Eso sí hay que saber que parte de los sistemas de localización de exteriores no pueden ser utilizados. En las últimas investigaciones hay una clara tendencia a utilizar redes neuronales. Con ellas es posible reconocer objetos en entornos, como es el caso de las señales de tráfico o incluso poder segmentar la imagen identificando la zona navegable. Sin olvidar que son capaces de detectar las líneas de un carril de forma más eficiente.

## 3.3 Evasión de obstáculos en robótica móvil

Como hacen referencia Nai-Hsiang Chang et al. [66], en el campo de la robótica, la evasión de obstáculos ha sido siempre un tema de investigación muy importante. Especialmente en los últimos años, la tecnología de conducción automática ha ido ganando atención, y el algoritmo de evasión de obstáculos se ha convertido en una tecnología importante e indispensable. Además de poder evitar los obstáculos con precisión, el coste de la máquina y el consumo de energía también son cuestiones muy importantes.

Por otro lado, la evasión de obstáculos en tiempo real es una de las cuestiones clave para el éxito de las aplicaciones de los sistemas robóticos móviles [67]. Todos los robots móviles cuentan con algún tipo de sistema de evasión de colisiones, desde algoritmos que detectan un obstáculo y detienen al robot antes de llegar a él para evitar una colisión, hasta sofisticados algoritmos que permiten al robot sortear los obstáculos. Estos últimos algoritmos son mucho más complejos, ya que implican no sólo la detección de un obstáculo, sino también algún tipo de mediciones cuantitativas sobre las dimensiones de este. Una vez determinadas, el algoritmo de evasión de obstáculos tiene que dirigir el robot alrededor del obstáculo y reanudar el movimiento hacia el objetivo original [68].

Para intentar dar con una solución al problema de evitar obstáculos, varios métodos han sido propuestos en la literatura a lo largo de los años [61], [62], [69], [70]. Hay una gran variedad de sensores disponibles que pueden ser implementados para la detección de obstáculos. Algunos de los sensores más populares son: sensores infrarrojos, cámaras, que pueden ser utilizados como parte de la visión por computador, sonar o LIDAR, que puede medir directamente la distancia de miles a cientos de miles de puntos en su campo de visión, entre otros [71]. Un breve resumen de estos tres métodos puede encontrarse en la **Tabla 4**.

Como exponen R. N. Kandalan y K. Namuduri [72] la detección general de obstáculos es un elemento clave para evitarlos en la robótica móvil y la conducción autónoma. En los últimos años, la aparición de unidades de computación rápidas y de abundantes repositorios de datos ha abierto un nuevo dominio en la detección de obstáculos basada en la visión por ordenador.

**Tabla 4.** Resumen de los métodos de evasión de obstáculos [72].

| Método para la evasión de obstáculos | Ventajas | Desventajas | Ejemplos |
|---|---|---|---|
| Sonar | Simple | Incapaz de detectar la ubicación exacta | [73] |
| LiDAR | Rápido | Sensible al color y a la transparencia | [74] |
| Visión por ordenador | Información detallada | Requiere una gran potencia de cálculo | [66], [75], [76] |

Además, el rápido desarrollo del aprendizaje automático, especialmente el aprendizaje profundo [77], han promovido el autoaprendizaje como un nuevo punto de estudio para que el robot evite los obstáculos [78]. En esta categoría de técnicas, se han utilizado diferentes descendientes de arquitecturas de redes neuronales para detectar el obstáculo. En los métodos basados en redes neuronales, el obstáculo tiene que ser definido por el sistema de antemano. Por ejemplo, se utilizan diferentes técnicas de reconocimiento de objetos, como CNN o redes neuronales profundas, para encontrar el objeto en la imagen y estimar la posición del obstáculo con respecto al usuario. Las redes neuronales profundas (RCNN) y la familia YOLO también son muy utilizadas para este fin [79].

Como se ha comentado anteriormente, la CNN es actualmente el método de red neuronal más potente en el reconocimiento de imágenes. Hay muchas investigaciones y logros que han aplicado la CNN para resolver problemas prácticos, incluyendo problemas de evasión de obstáculos [76], [80], [81]. El método de evasión de obstáculos para el reconocimiento de objetos consiste en clasificar correctamente los diferentes objetos mediante el entrenamiento de las redes neuronales, y luego combinar el algoritmo para que el robot sepa cómo reaccionar cuando se enfrenta a diferentes objetos [81].

Para finalizar, y dadas las necesidades específicas que requieren las diferentes aplicaciones de los robots móviles, especialmente en la navegación, es crucial desarrollar un sistema robótico autónomo que sea capaz de evitar obstáculos mientras sigue la trayectoria en aplicaciones en tiempo real. En consecuencia, y como comentan Marwah M. Almasri et al. [60], una técnica eficiente para evitar colisiones y seguir la trayectoria es esencial para asegurar un sistema de robot móvil autónomo inteligente y eficaz.

## 3.4 Técnicas de localización y mapeo

La localización de los robots móviles autónomos es el proceso de determinar y seguir la posición y la orientación de los robots en un entorno determinado. Además, la localización es una técnica fundamental que permite a los robots navegar, explorar o realizar las tareas que se proponen con éxito sin intervención humana [82]. El tema de la localización de interiores ha recibido una atención considerable en las últimas décadas [83]–[85]. De hecho, la auto-localización es el primer requisito para construir un agente capaz de guiarse a sí mismo a través de un entorno

interior conocido o desconocido, donde los actuales sistemas de posicionamiento por satélite sufren la atenuación de la señal relacionada con la propagación de las ondas electromagnéticas a través de las paredes [86].

Como explican Francisco A. X. Da Mota et al. [49] entre los cuatro módulos claves de la navegación autónoma comentada anteriormente, la localización es el problema que más se aborda en la literatura. Por ejemplo, Lim et al. [83] afirman que un requisito clave para el éxito de la navegación del robot es identificar la localización actual exacta. Asimismo, es importante destacar que, además de determinar su posición absoluta en el espacio, la localización también implica la construcción de un mapa y la determinación de la posición del robot en relación con él.

La localización puede dividirse en localización local y global [49]. En general, el uso de un GPS resuelve el problema de la localización global de forma eficiente en exteriores. En cambio, los principales debates están relacionados con la localización local en entornos interiores. Algunos de los métodos que encontramos en la literatura son la odometría [84], la visión por ordenador [84], los códigos QR [87], los marcadores fiduciarios [85], [88], o la RFID [89], entre otros.

De igual modo, Michail Kalaitzakis et al. [88], anotan que la localización robusta es fundamental para la navegación y el control de los robots móviles. Los sistemas globales de navegación por satélite (GNSS), la odometría visual-inercial (VIO) y la SLAM ofrecen diferentes métodos para lograr este objetivo. Sin embargo, exponen que, en algunos casos, estos métodos pueden no estar disponibles o no proporcionar una precisión suficiente. En tales casos, estos métodos pueden ser aumentados o sustituidos por la estimación de la posición a través de marcadores fiduciarios.

Estos mismos autores realizan también, un resumen de los marcadores fiduciales y sus posibles aplicaciones para la estimación de la posición de robots móviles. Igualmente, comentan que los marcadores fiduciarios pueden aumentar la precisión y robustez de un sistema de localización al proporcionar una característica fácilmente reconocible con detección de fallos incorporada, además de ser una solución sencilla, barata, eficiente y fiable [88].

Igualmente, explican que los marcadores fiduciales en su forma general son objetos utilizados para proporcionar un punto de referencia o una medida en una imagen. Las aplicaciones abarcan varias disciplinas, desde la médica hasta la fabricación de placas de circuito impreso. En los casos en que los marcadores se utilizan únicamente como punto de referencia, se emplean patrones sencillos como círculos o anillos. En otros casos, los marcadores fiduciales también incluyen una identificación o un mensaje específico codificado. Los códigos de barras y los códigos QR son los ejemplos más conocidos de este tipo de marcadores [88].

Una de las principales aplicaciones de los marcadores fiduciales en robótica es la localización y el mapeo. Aunque otros métodos como VIO y SLAM pueden proporcionar datos precisos, requieren condiciones de iluminación ideales y características visuales distintivas. Los marcadores fiduciales se utilizan para mejorar la calidad de la estimación de la posición en entornos donde estas condiciones no se cumplen [88].

En este trabajo se emplearán marcadores AprilTag [90], [91], los cuales se basan en el marco establecido por ARTag [92], [93], pero que ofrecen una serie de mejoras. Estas mejoras,

permiten a AprilTag tener una mayor robustez frente a las oclusiones y las deformaciones, así como una menor cantidad de detecciones erróneas. Además, está en constante desarrollo a medida que los investigadores optimizan el algoritmo. Sin embargo, existen muchos otros tipos de marcadores fiduciales. Por ejemplo, en [88], se realiza un estudio de los marcadores más empleados hoy en día. Un resumen se realiza en la **Tabla 5**.

**Tabla 5.** Resumen de marcadores fiduciarios y sus principales características.

| **Nombre** | **Características** (Forma + Color) | **Marcador** | **Observaciones** |
|---|---|---|---|
| AprilTag [90], [91] | Cuadrada + Monocroma |  | Método de detección mejorado para una detección más rápida y precisa. |
| ARToolKit [94] | Cuadrada + Monocroma |  | Uno de los paquetes más antiguos. Muchos marcadores se basan en este paquete. |
| ArUcO [95] | Cuadrada + Monocroma |  | Permite el uso de bibliotecas configurables. Su detección mejora a otros métodos en términos de distancia entre marcadores o tasa de falsos positivos. |
| TopoTag [96] | Cuadrada o Circular + Monocroma |  | Marcadores con forma customizable que supera a los otros sistemas de marcadores en términos como la precisión de la detección o la fluctuación. |
| ChromaTag [97] | Cuadrada + Multicolor |  | Tasas de detección rápidas, pero no robustas. |

Por otro lado, la SLAM es uno de los problemas fundamentales para los robots móviles que navegan en entornos interiores [98], [99]. Como explican Cesar Cadena et al. [100], la SLAM comprende la estimación simultánea del estado de un robot equipado con sensores a bordo y la construcción de un modelo (el mapa) del entorno que perciben los sensores. En casos sencillos, el estado del robot se describe por su pose (posición y orientación), aunque otras magnitudes pueden incluirse en el estado, como la velocidad del robot, los sesgos de los sensores y los parámetros de calibración. El mapa, por su parte es una representación de los aspectos de interés (por ejemplo, la posición de los puntos de referencia o los obstáculos) que describen el entorno en el que opera el robot. En algunos casos, es posible que ya se tenga un mapa construido utilizando un algoritmo de mapeo, y sólo se necesite estimar las poses relativas en el mapa dado.

La necesidad de utilizar un mapa del entorno es doble. En primer lugar, el mapa suele ser necesario para apoyar otras tareas; por ejemplo, un mapa puede informar sobre la planificación de la trayectoria o proporcionar una visualización intuitiva para un operador humano. En segundo lugar, el mapa permite limitar el error cometido al estimar el estado del robot. En ausencia de un mapa, la estimación del estado del robot se desviaría rápidamente con el tiempo; por otro lado, con un mapa, por ejemplo, con un conjunto de puntos de referencia distinguibles, el robot puede "restablecer" su error de localización volviendo a visitar zonas conocidas (lo que se denomina cierre de bucle). Por lo tanto, la SLAM tiene aplicaciones en todos los escenarios en los que no se dispone de un mapa previo y es necesario construirlo [100].

Finalmente, como expresan Andrea Motroni et al. [86], al abordar las técnicas de localización, se pueden resolver diferentes tareas: el posicionamiento, el seguimiento (tracking) o la SLAM (**Figura 7**). El posicionamiento se da cuando el método estima la posición puntual del vehículo en un momento dado, estando estático o en movimiento. Se habla de seguimiento o tracking cuando el método estima la trayectoria del vehículo, es decir, una secuencia de posiciones consecutivas asumidas por el vehículo en movimiento a lo largo del tiempo. Dichas posiciones pueden ser estimadas de forma independiente (localización sin memoria), o teniendo en cuenta el historial previo de ubicaciones de la unidad móvil, así como de su velocidad y aceleración. Por último, la SLAM consiste en el seguimiento del vehículo junto con la adquisición del mapa del entorno [101]–[103].


**Figura 7.** Principios de posicionamiento, seguimiento y SLAM [86].

## 3.5 Trabajo relacionado

Por último, y para finalizar con el apartado 3, se muestran dos tablas (ver **Tabla 6** y **Tabla 7**), en las que se realiza un resumen de varios trabajos de investigación relacionados con la conducción autónoma de vehículos de interiores. La **Tabla 6** se centra en aplicaciones que no utilizan la cámara como sensor principal, mientras que la **Tabla 7** se centra en aquellas aplicaciones que sí utilizan la visión por ordenador para llevar cabo alguna de las tareas que se van a realizar en este proyecto: la navegación, la evasión de obstáculos o la localización.

| Tipo de algoritmo | Ref. | Sensor primario | Breve descripción de la investigación |
|---|---|---|---|
| Navegación | [104] | Multi-sensor | Este estudio propone la fusión de información multifuente, es decir, la fusión de la tecnología de posicionamiento de banda ultra ancha con un odómetro y un acelerómetro giroscópico de bajo coste, para lograr el posicionamiento de un robot de reparto de comida no ferroviario con navegación. |
| Evasión de obstáculos | [67] | Sensor IR | Este artículo presenta una solución de bajo coste para la evasión de obstáculos para un robot móvil. Además, presenta un algoritmo de dirección dinámica para que el robot no tenga que detenerse ante un obstáculo. |
| Navegación + detección de obstáculos | [74] | LiDAR | En este trabajo, se realizan algoritmos de planificación de trayectorias y evasión de obstáculos basados en un sistema de navegación híbrido en tiempo real para robots móviles. El robot tiene una plataforma de tracción diferencial para el movimiento y un LiDAR para la visión. |
| | [73] | Sonar | En este trabajo se emplea un anillo de sonar multi-DSP en un robot móvil para realizar una navegación siguiendo paredes además de evitar obstáculos |
| Navegación + Localización | [105] | Lectores RFID | En este trabajo se usan redes Petri (PN) para la llevar a cabo las tareas de localización y navegación en un robot móvil autónomo de interiores. El objetivo es, dado un mapa representado por la matriz de incidencia de una red de Petri, evaluar el uso de la tecnología de identificación por radiofrecuencia (RFID) para reconocer la posición de un robot en este mapa, así como el uso de la dinámica de la PN como sistema de cognición de este robot. |
| Localización | [82] | Ultrasonidos | En este trabajo se presenta un sistema de localización híbrido ultrasónico dinámico para la navegación autónoma de robots móviles de interior utilizando múltiples mediciones de distancia ultrasónicas y un filtro de Kalman Extendido (EKF). |

**Tabla 6.** Trabajos de investigación relacionados sobre vehículos autónomos de interior (i).

Tabla 7. Trabajos de investigación relacionados sobre vehículos autónomos de interior (ii).

| Tipo de algoritmo | Ref. | Sensor primario | Breve descripción de la investigación |
|---|---|---|---|
| Navegación | [59] | Cámara | En este artículo, se propone un robot de servicio que actúa como un vehículo autónomo de seguimiento de líneas. Se utilizan dos controladores simultáneos: para el seguimiento del objetivo y la predicción de la trayectoria se utilizan CNNs, y para el control automático de la dirección y la velocidad se diseña un controlador PID. |
| | [65] | Cámara | En este estudio se presenta una red de segmentación semántica para desarrollar un sistema de navegación en interiores para un robot móvil. |
| Evasión de obstáculos | [75] | Cámara | Se presenta un modelo de aprendizaje de extremo a extremo basado en una red neuronal convolucional (CNN), que toma la imagen obtenida de la cámara como única entrada. El método convierte directamente los píxeles en órdenes de dirección, como girar a la izquierda, girar a la derecha o seguir recto. |
| | [76] | Cámara | En este trabajo, se emplea el aprendizaje por refuerzo profundo empleando visión monocular RGB para evitar obstáculos. |
| Localización | [85] | Cámara | En este trabajo se emplea el algoritmo del Filtro Extendido de Kalman (EKF) para fusionar la información odométrica y los datos de medición de una cámara a partir de la detección marcadores Aruco. |
| | [84] | Cámara | En este trabajo, se aborda la localización y el mapeo simultáneos de entornos interiores basados en cámaras RGB-D utilizando características planas. El método propuesto solo utiliza las características del plano. Este método proporciona una potente alternativa a los sistemas SLAM RGB-D basados en características puntuales en entornos sin textura o cuando la cámara RGB-D apunta a un área fuera del rango de profundidad válido. |

# Desarrollo de la solución 4

## 4.1 Algoritmo de navegación

Como se ha comentado en el capítulo anterior, la navegación es el proceso por el que un robot móvil se mueve por el entorno para ejecutar una determinada tarea. La navegación autónoma se realiza cuando el robot se mueve sin ninguna interferencia de un controlador externo (por ejemplo, una persona o un sistema central). Además, la navegación autónoma depende de la resolución de cuatro problemas principales: localización, mapeo, planificación de la trayectoria y locomoción.

La idea de este proyecto es emplear la visión como sentido único para realizar las cuatro tareas principales de un sistema de navegación, ya que dar prioridad a los sistemas de navegación basados en la visión es beneficioso a la hora de integrar los robots en nuestra vida cotidiana. Esto permite minimizar la necesidad de realizar adaptaciones en entornos comunes y configuraciones especializadas para que los robots funcionen correctamente.

Asimismo, el seguimiento de la trayectoria es uno de los aspectos más importantes de los vehículos autónomos. El diseño, tanto de un controlador lateral, como longitudinal, permitirá calcular unos parámetros de control del vehículo adecuados, lo cual es esencial para poder seguir un recorrido de manera precisa.

En este trabajo, se empleará una trayectoria predefinida por el usuario de antemano para guiar al vehículo durante su recorrido, y se utilizará un controlador sencillo basado en visión artificial para controlar las ruedas de un AGV y poder seguir la trayectoria correctamente.

### 4.1.1 Planificación de la trayectoria

La planificación de trayectorias es el proceso de búsqueda y determinación de una trayectoria entre dos ubicaciones en el entorno, normalmente desde la ubicación actual hasta una posición objetivo. El método de planificación de trayectorias depende de la localización precisa del robot, de la cartografía precisa del entorno y del tipo de mapa construido por el proceso de cartografía. La planificación de la trayectoria es una de las cuatro tareas principales del problema de la navegación [48].

El control de seguimiento de la trayectoria en los vehículos autónomos tiene como objetivo principal proporcionar una entrada de dirección suficiente, así como una entrada de aceleración y frenado para controlar la dirección y la velocidad del vehículo para guiarlo de manera controlada a lo largo de una trayectoria predefinida.

La trayectoria que seguirá el AGV se define mediante una línea amarilla sobre el entorno de trabajo. Esta marca todo el recorrido que debe hacer el vehículo, y el objetivo será seguirla hasta finalizarlo, es decir, hasta que no exista más línea. En la Figura 8 se puede observar un ejemplo de cómo podría ser una trayectoria basada en una línea en el laboratorio de la Escuela de Ingeniería de Vitoria-Gasteiz.

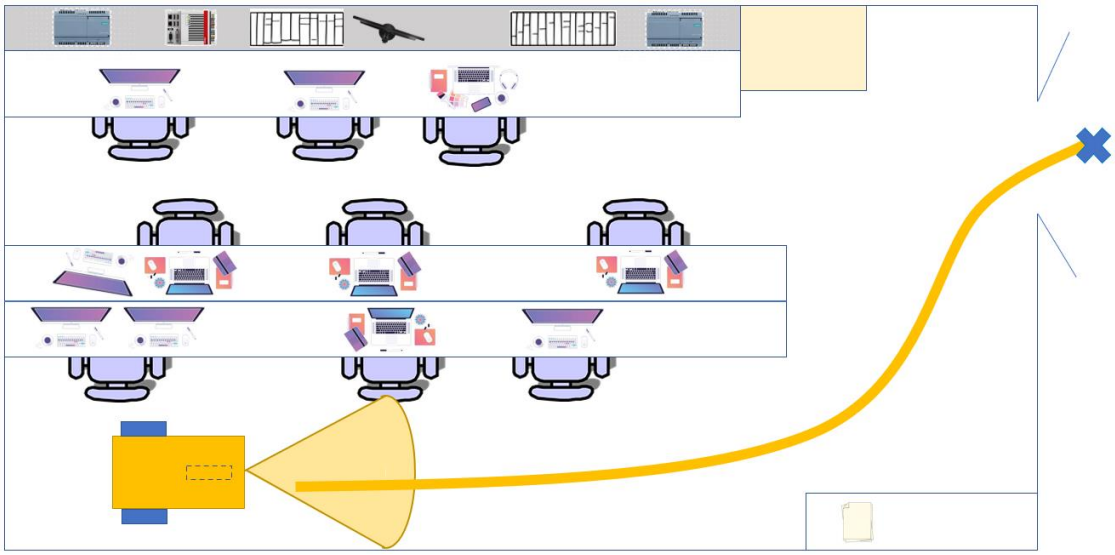


Figura 8. Ejemplo de trayectoria basada en un seguimiento de líneas.

El controlador de seguimiento de trayectoria se desarrolla normalmente para garantizar que el vehículo siga una trayectoria predefinida, determinando y calculando la entrada de actuación deseada para que el vehículo la siga. Puede tratarse de una entrada de dirección correctiva para ajustar la posición del vehículo en dirección lateral o de un ajuste de freno o acelerador correctivo para ajustar el movimiento del vehículo en dirección longitudinal.

### 4.1.2 Flujo de trabajo para el algoritmo de navegación

Para este trabajo se recogerán un conjunto de datos que permitirán a un AGV, como se ha comentado anteriormente, seguir una línea, a través de un algoritmo de aprendizaje profundo. Estos datos permitirán entrenar una red neuronal convolucional con dos salidas: una salida de regresión y otra salida de clasificación. La idea es, por un lado, enseñar al vehículo a detectar una coordenada de imagen objetivo $(x, y)$ que el AGV perseguirá, y, por otro lado, enseñar al vehículo a detectar cuando existe o no carril. Por consiguiente, cuando haya carril, se emplearán las consignas de regresión para determinar la velocidad lineal y el ángulo de giro del vehículo, mientras que, si no hay carril, el vehículo se detendrá. El flujo de trabajo para el algoritmo de navegación se puede observar en la Figura 9.

Para ello se empleará una CNN preentrenada. Esta red se modificará para que tenga dos salidas, y se entrenará dos veces más, con nuevos datos de entrenamiento de tipo regresión y clasificación. De este modo se podrá predecir correctamente cuándo existe o no carril, así como el valor objetivo del recorrido al que debe dirigirse el vehículo.

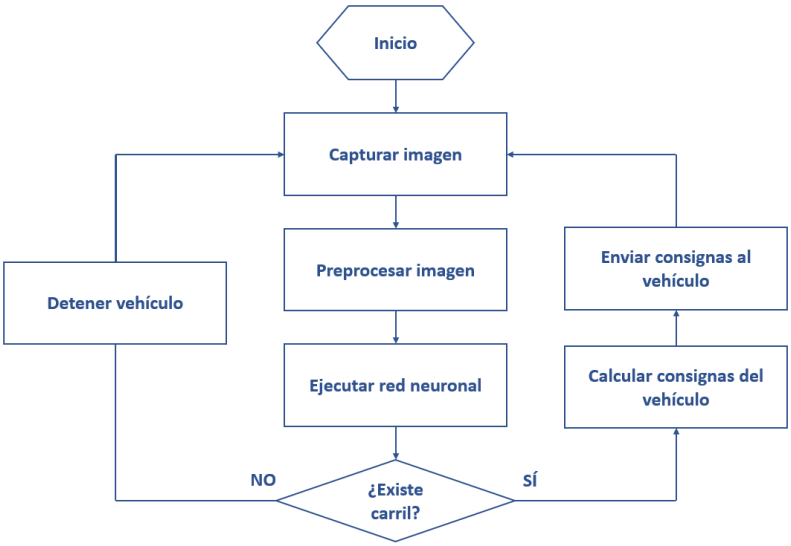



Figura 9. Flujo de trabajo del algoritmo de navegación del vehículo autónomo.

Por otro lado, el proceso de entrenamiento de la CNN multi-salida diseñada se puede observar en el **Algoritmo 1**.

**Algoritmo 1:** Proceso de entrenamiento de una CNN multi-salida

| | |
|---|---|
| **1** | Recopilar dos conjuntos de datos para el entrenamiento y validación de la red neuronal |
| | **a** Conjunto de datos para regresión |
| | **b** Conjunto de datos para clasificación |
| **2** | Diseñar una CNN propia o escoger una preentrenada como algoritmo de navegación para resolver el problema |
| **3** | Modificar la red para que pueda predecir datos de regresión |
| **4** | Entrenar la red para regresión |
| **5** | Modificar la nueva red para realizar la tarea de la clasificación de datos |
| **6** | Entrenar la red para clasificación |
| **7** | Unir ambas redes para que pueda predecir dos salidas simultáneamente |
| **8** | Ejecutar la CNN y mover el vehículo |
| **9** | ***fin*** |

### 4.1.3 Conjuntos de datos para el algoritmo de navegación

Las CNN son herramientas esenciales para el DL y están especialmente indicadas para el análisis de datos de imágenes. Por ejemplo, las CNN pueden utilizarse para clasificar imágenes. Sin embargo, si se quiere predecir datos continuos, como ángulos o distancias, se puede incluir una capa de regresión al final de la red.

En este caso, el objetivo es tanto clasificar imágenes como predecir datos continuos, por lo que será necesario entrenar la red para ambos casos. Para ello, se utilizarán dos conjuntos de datos (ver Figura 10 y Figura 11).

1) Conjunto de datos para clasificación: Imágenes etiquetadas con las clases de “Línea” y “No Línea”.

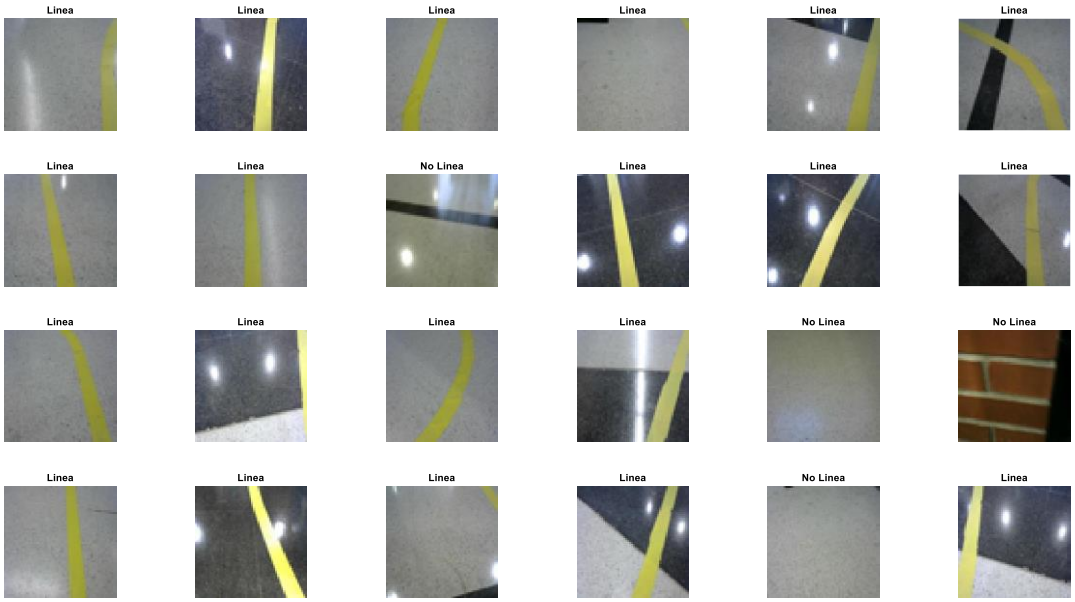



**Figura 10.** Imágenes correspondientes al conjunto de datos de clasificación 'Línea' y 'No Línea'.

2) Conjunto de datos para regresión: Imágenes etiquetadas con las coordenadas $(x, y)$ a la que debe ir el vehículo (puntos rojos en las imágenes de la **Figura 11**).

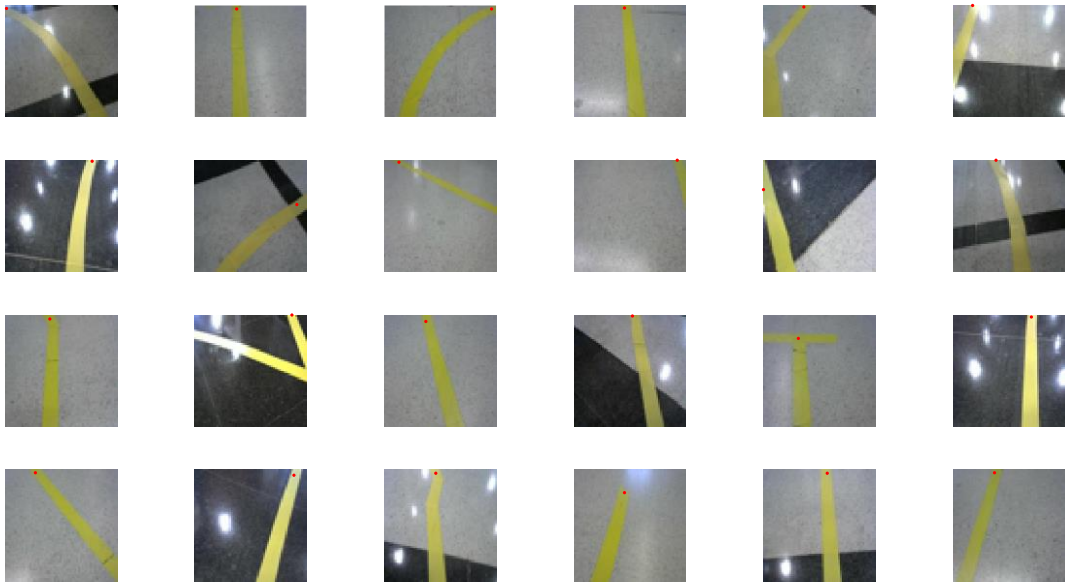


**Figura 11.** Imágenes correspodnientes al conjunto de datos de regresión.

Para reducir el tiempo de entrenamiento y ejecución de la CNN, así como para mejorar la precisión del algoritmo, se ha decidido reducir el tamaño de todas las imágenes de entrenamiento a 56x56 píxeles. Además, se han etiquetado las imágenes para que en caso de que exista una bifurcación el coche siempre se dirija hacia la derecha. Por otro lado, se ha realizado un procesado del conjunto de imágenes de manera manual para aumentar el número de imágenes de entrenamiento. Este aumento ha consistido en voltear las imágenes verticalmente. En la **Tabla 8** se muestra un resumen de los dos conjuntos de datos utilizados para este trabajo.

**Tabla 8.** Resumen de los conjuntos de datos propios.

| Conjunto de datos | Número de imágenes | | | Salida |
|---|---|---|---|---|
| | Entrenamiento (90%) | Validación (10%) | Total | |
| Clasificación | 6271 | 698 | 6968 | Imágenes etiquetadas con 'Línea' o 'No Línea'. |
| Regresión | 4896 | 544 | 5440 | Imágenes etiquetadas con el punto de referencia $(x, y)$ que marca la dirección del vehículo. |

### 4.1.4 Entrenamiento de una CNN multi-salida

Para el entrenamiento de CNN con dos salidas, en primer lugar, será necesario diseñar una arquitectura de CNN. Lo más sencillo es utilizar una red neuronal ya existente y modificar su arquitectura para que se ajuste a las necesidades del problema. En la sección de resultados, se compararán varias arquitecturas existentes para determinar cuál de ellas da los mejores resultados. Esta comparación tendrá en cuenta tanto la precisión de los resultados de clasificación y regresión, como el tiempo de ciclo de ejecución del algoritmo de aprendizaje profundo.

Las CNN se utilizan normalmente para clasificar imágenes. Sin embargo, es necesario predecir los valores de $(x, y)$ para seguir la línea correctamente, por lo que la última capa de clasificación debe cambiarse por una capa de regresión de dos variables.

En segundo lugar, será necesario entrenar la nueva red con datos que representen el problema. Para ello, hay que recoger datos del entorno y etiquetarlos. La base de datos utilizada se muestra en la **Figura 11**, la cual se ha dividido de forma que el 90% de los datos son para el entrenamiento y el 10% para la validación.

Una vez entrenada la red neuronal convolucional para la regresión, se guardan los pesos y se modifican las últimas capas para obtener una red neuronal convolucional para clasificación. La base de datos empleada se observa en la **Figura 10** y se ha dividido igual que el conjunto de datos para regresión. Sin embargo, será necesario modificar algunas características para que las capas no aprendan y la CNN pueda realizar dos tareas de manera simultánea y precisa. Sólo la capa final totalmente conectada (fully connected) debe aprender para que la red funcione correctamente. Las propiedades que hay que modificar se muestran en la **Tabla 9**. De esta forma se anula el aprendizaje de ciertas capas.

Después de terminar el entrenamiento de la red neuronal de clasificación y guardarla, las dos redes se fusionan. De esta manera, se tiene la misma red para realizar la regresión y la clasificación al mismo tiempo.

Tabla 9. Propiedades para modificar.

| Capa | Propiedad | Valor |
|---|---|---|
| Convolución | Weight Learn Rate Factor<br>Weight L2 Factor<br>Bias Learn Rate Factor<br>Bias L2 Factor | 0 |
| Normalización por lotes (Batch Normalization) | Offset Learn Rate Factor<br>Offset L2 Factor<br>Scale Learn Rate Factor<br>Scale L2 Factor | |

La arquitectura de la nueva arquitectura propuesta se muestra en la Figura 12.

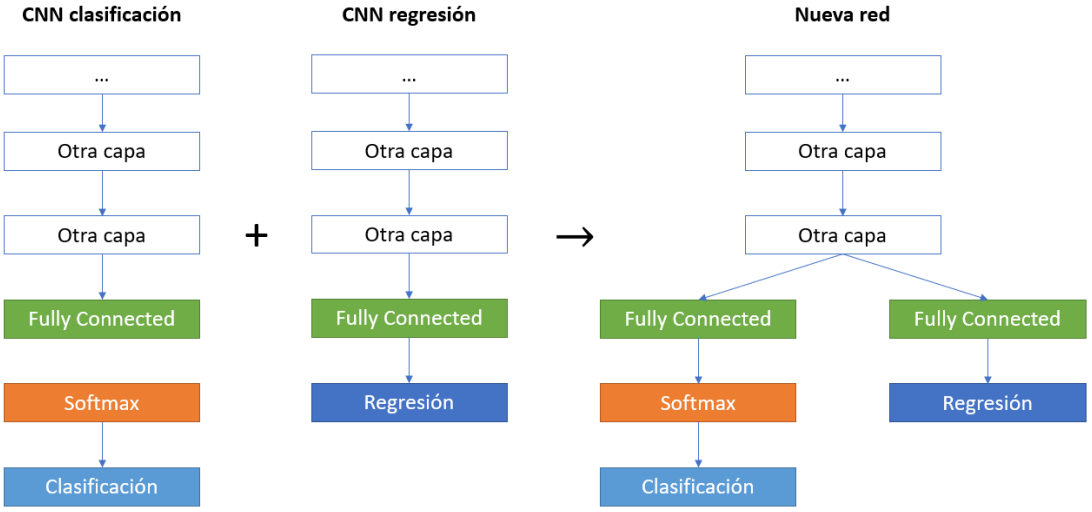



Figura 12. Arquitectura de la red propuesta.

Sin embargo, Matlab no permite generar una CNN con dos salidas con su estructura para redes neuronales convolucionales estándar *DAGNetwork*. Por consiguiente, se deben quitar las capas de clasificación y regresión y convertirla al tipo *dlnetwork*. La estructura final de la nueva red se puede observar en la Figura 13.

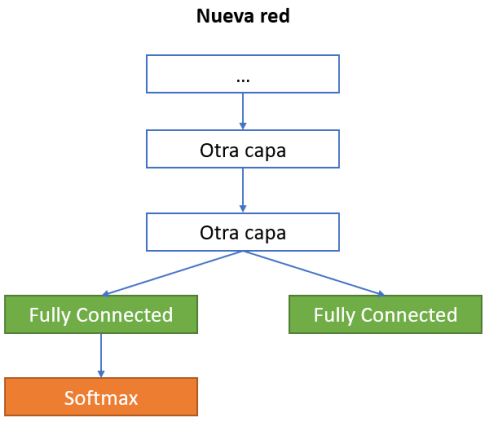



Figura 13. Nueva arquitectura de red *dlnetwork*.

Estos cambios no afectan en nada a los resultados de la red. Simplemente se deben recoger y procesar. Por ejemplo, en la red de clasificación, en vez de obtener la salida categórica "Línea" o "No Línea" se obtendrá el porcentaje de probabilidades que la red devuelve a que la imagen corresponda a las clases "Línea" o "No Línea".

### 4.1.5 Cálculo de las consignas del vehículo

Se han propuesto una variedad de métodos de control de movimiento para robots autónomos: control proporcional integral-derivativo (PID), control difuso, control de redes neuronales y alguna combinación de estos algoritmos de control [106]. Sin embargo, el algoritmo de control PID se utiliza en la mayoría de las aplicaciones de control de movimiento, y los métodos de control PID se han ampliado con técnicas de aprendizaje profundo para lograr un mejor rendimiento y ofrecer una mayor adaptabilidad.

En este último aparatado de la sección 4.1, *Algoritmo de Navegación*, se diseñará un controlador PD para controlar la velocidad de las ruedas del AGV. Para ello, se parte del punto de referencia que debe seguir el vehículo (véase **Figura 14**), el cual se ha obtenido con la red de regresión comentada anteriormente.

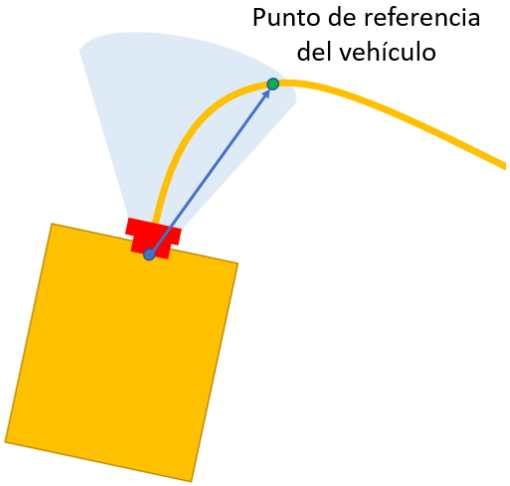



**Figura 14.** Ejemplo del punto de referencia que debe seguir el vehículo en una trayectoria definida.

De esta forma, partiendo del punto $(x, y)$ se calcula el error o la desviación de la magnitud física respecto al punto de consigna de acuerdo con la ecuación (1). La idea es dejar este punto de referencia $x$ en el centro de la imagen.

$$error = \arctan\left(\frac{x - \frac{size_x}{2}}{size_y - y}\right) \quad (1)$$

Donde $[size_x, size_y]$ es el tamaño de la imagen. En este caso es $[56, 56]$. En la **Figura 15** se puede observar mejor este término de error.

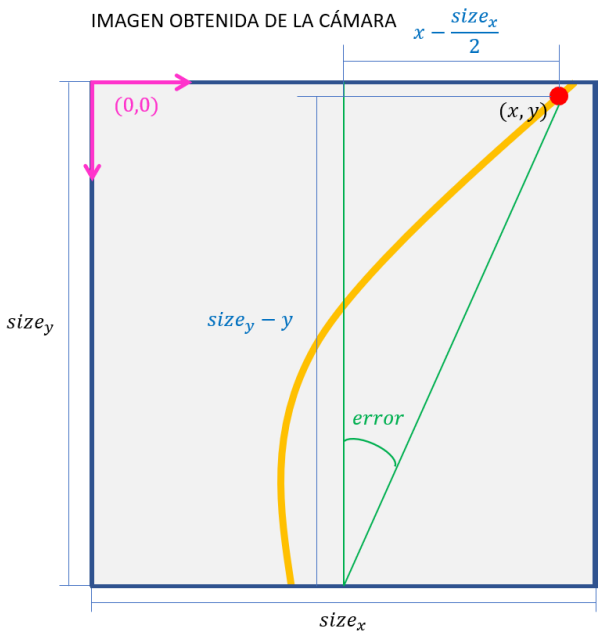



**Figura 15.** Imagen de ayuda para obtener el error de seguimiento.

Partiendo entonces del término de error, se emplea la ecuación (2) para calcular la fórmula del controlador.

$$PD = K_p * error + K_d * (error - error\ anterior) \quad (2)$$

Donde,

- $Kp$ es la constante utilizada para determinar el tiempo de subida del lazo de control o la rapidez con la que alcanzará la consigna.
- $Kd$ es la constante utilizada para variar la estabilidad del sistema.

El PID controlará las velocidades del motor izquierdo y derecho en función de la medición del error previsto del sensor. El controlador PID generará una señal de control (valor PID), que se utilizará para determinar la velocidad izquierda y derecha de las ruedas del robot.

En nuestro vehículo, se consigue un giro a la izquierda si se reduce la velocidad del motor izquierdo, y se consigue un giro a la derecha si se reduce la velocidad del motor derecho. Las ecuaciones (3) y (4) muestran el cálculo de estas velocidades a partir del valor obtenido por el controlador PD.

$$velocidad\ rueda\ derecha = velocidad\ base + PD \quad (3)$$

$$velocidad\ rueda\ izquierda = velocidad\ base - PD \quad (4)$$

Por último, la $velocidad\ base$ dependerá del valor de $y$ obtenido por la CNN de regresión. Cuanto más pequeño sea este valor, más rápido irá el vehículo, ya que la referencia estará marcando un punto más lejano. Esto significa que el vehículo irá más rápido en las rectas, mientras que en las curvas reducirá su velocidad. Entonces, la $velocidad\ base$ se calcula de acuerdo con la ecuación (5).

$$velocidad\ base = \begin{cases} v_{min} & \text{si } velocidad\ base < v_{min} \\ k_v \dfrac{size_y - y}{y} & \text{si } v_{min} < velocidad\ base < v_{max} \\ v_{max} & \text{si } velocidad\ base > v_{max} \end{cases} \quad (5)$$

## 4.2 Algoritmo de detección de obstáculos

La evasión o evitación automática de obstáculos es crucial para muchos sistemas autónomos, como los vehículos aéreos no tripulados (UAV), o los Robots Móviles Autónomos (ARM). En los últimos años, el desarrollo de robots móviles inteligentes y autónomos está en el centro de la investigación sobre vehículos autónomos, debido a su utilidad para ayudar a la seguridad y el control intuitivo cuando se trabaja en diferentes escenarios, por ejemplo, la vigilancia, la cartografía, la supervisión de la construcción, la entrega y la supervisión del tráfico, entre otros. Para adaptarse a los entornos de trabajo y llevar a cabo las tareas mencionadas, en un sistema de control de ARM deben abordarse simultáneamente la percepción, el control y la localización, entre los que se encuentra la detección de obstáculos. De acuerdo Yuri D. V. Yasuda et al. [48] se definen los siguientes conceptos:

- **Obstáculo.** Un obstáculo es una parte del entorno, un agente o cualquier objeto con el que el robot debe evitar colisionar.
- **Detección de obstáculos.** La detección de obstáculos es el proceso de encontrar un obstáculo y determinar su posición. Puede realizarse mediante mediciones de distancia, imágenes y sonidos. Es importante para evitar colisiones con el robot, que podrían provocar lesiones o daños. Como se ha comentado anteriormente, la detección de obstáculos es una subtarea del problema de locomoción.

Aunque los sensores más conocidos que se utilizan para seguir una trayectoria específica mientras se detectan los obstáculos y se mide la distancia entre los robots y los objetos, son los sensores infrarrojos, ultrasónicos y láser, en este caso, se desea emplear la visión como única fuente de información para la navegación del robot móvil. Por consiguiente, dado que una única cámara no es suficiente para realizar el seguimiento de líneas y la detección de obstáculos al mismo tiempo, se ha decidido incorporar otra cámara más, para llevar a cabo esta tarea.

Por otro lado, los obstáculos que se van a considerar para que el vehículo detecte y actúe en consecuencia serán exclusivamente personas. La consigna tras la detección de algún obstáculo será la detención del vehículo, debido a que normalmente los obstáculos serán dinámicos y puede suponer un mayor peligro tratar de evitarlo y perder la trayectoria de vista. Además, los AGV suelen ser empleados para transportar mercancías en la industria y no supondría un inconveniente detener el vehículo y perder algo de tiempo.

Para la evasión de obstáculos de los vehículos autónomos, algunos requisitos básicos para el procesamiento de imágenes incluyen las siguientes características:

1. **Precisión.** Específicamente, la predicción debe alcanzar casi el 100% de aciertos para obtener un sistema de detección de obstáculos fiable.
2. **Rapidez.** La predicción debe garantizar el procesamiento en tiempo real y una velocidad de inferencia rápida para reducir la latencia del bucle de control del vehículo autónomo.

Cumpliendo los requisitos anteriores, la evitación de obstáculos se consigue mediante redes eficientes y ligeras en este trabajo. Por otro lado, la detección de obstáculos en imágenes individuales es un problema difícil en la navegación autónoma en condiciones de bajo coste. En este trabajo se emplea otra CNN para identificar posibles obstáculos en la trayectoria del vehículo. En función de la predicción de la red el vehículo deberá detenerse o deberá continuar con su trayecto. Añadiendo la nueva mejora al flujo de trabajo visto en la Figura 9 se obtiene el diagrama de flujo de la Figura 16 en el que se observa que, si se detecta obstáculo, no se ejecuta el algoritmo de seguimiento de líneas, y el vehículo se detiene.

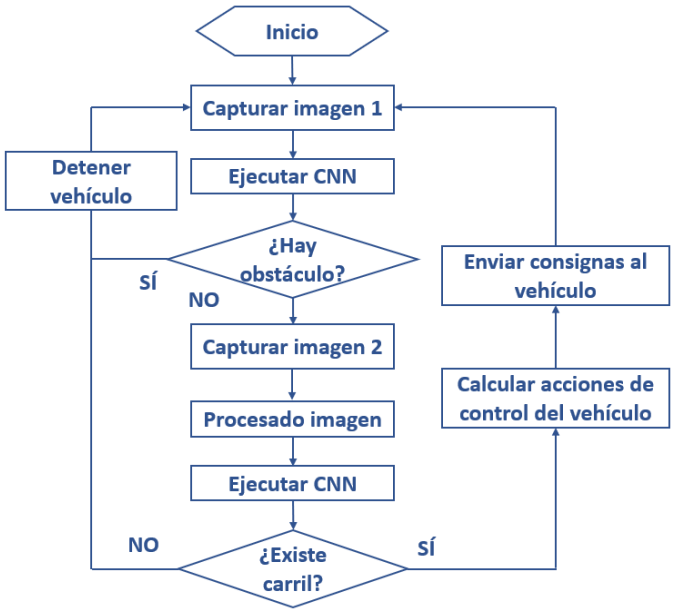



Figura 16. Diagrama de flujo del algoritmo de navegación tras detectar obstáculo.

Por otro lado, en la Figura 17 se pueden observar algunas imágenes del conjunto de datos utilizado para llevar a cabo la detección de obstáculos. En este caso, también se ha realizado un aumento del número de imágenes de entrenamiento. Sin embargo, se han empleado funciones predefinidas de Matlab como se verá más adelante. Por último, en la Tabla 10 se puede observar un resumen de la base de datos empleada para la detección de personas. Para esta CNN las imágenes tienen un tamaño de $[224, 224]$.

Tabla 10. Resumen de la base de datos para la detección de obstáculos.

| **Conjunto de datos** | **Número de imágenes** | | | **Salida** |
|---|---|---|---|---|
| | Entrenamiento (90%) | Validación (10%) | Total | |
| **Clasificación** | 4597 | 511 | 5108 | Imágenes etiquetadas con 'Con Obstáculo' o 'Sin Obstáculo'. |


**Figura 17.** Imágenes de la base de datos para la detección de obstáculos.

## 4.3 Algoritmo de localización y mapeo

En esta sección, se propone un algoritmo de localización para vehículos de conducción autónoma basado en un algoritmo de visión artificial. Para el desarrollo de este algoritmo se emplearán marcadores fiduciarios. Para ello, se discuten tres partes: el algoritmo de detección del marcador fiduciario, la estimación de la posición y el mapeo de la trayectoria del vehículo.

### 4.3.1 Algoritmo de localización

En este trabajo, como se ha comentado anteriormente, se emplean marcadores fiduciales como puntos de referencia. Cada marcador tendrá un ID y una posición $(x, y)$ propia en el espacio asociada. Los marcadores fiduciarios tienen buenas propiedades para extraer indicadores de una escena complicada y mixta, por lo que son una buena solución para un algoritmo de localización. En primer lugar, su tono y forma son fáciles de detectar. El tono de los marcadores fiduciales es blanco y negro, por lo que es fácil de distinguir en la imagen binaria. En segundo lugar, los vértices de los marcadores fiduciarios pueden extraerse fácilmente de los cruces de líneas gracias a su forma cuadrada.

El principio básico de la localización del marcador fiduciario consiste en extraer el marcador de la imagen obtenida por la cámara, determinar las coordenadas de los cuatro vértices del marcador y determinar el borde de este mediante la tecnología de reconocimiento de imágenes. Dado que la dimensión física del marcador es definida, la posición 3D y la dirección del marcador fiduciario en relación con la cámara pueden calcularse mediante la información del borde. Los marcadores fiduciarios empleados para el algoritmo de localización son, como ya se ha comentado, los AprilTag, los cuales se pueden observar en la **Figura 18**. Dentro de las muchas familias AprilTag disponibles, la escogida para esta aplicación ha sido la 'Tag36h11', ya que es una de las familias soportada por Matlab.

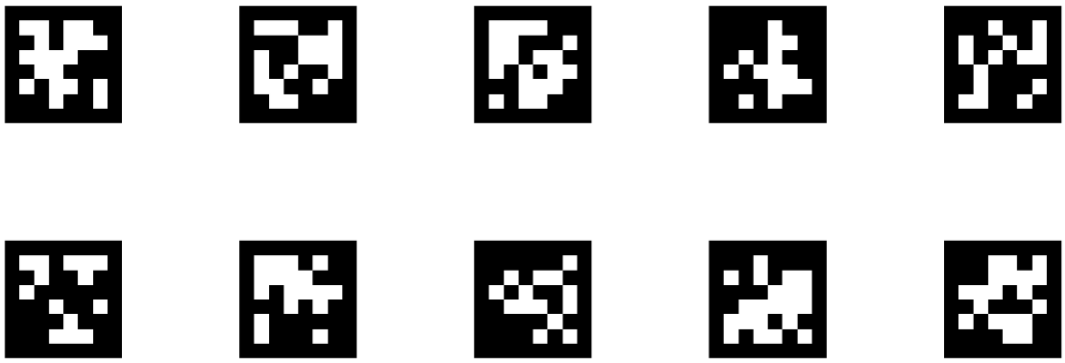


Figura 18. Marcadores AprilTag. Familia Tag36h11.

AprilTag está especialmente indicado para la estimación de la pose en aplicaciones de realidad aumentada (AR) y localización de robots, y se ha demostrado que tiene menores tasas de falsos negativos en comparación con otros sistemas.

El algoritmo de localización consistirá en calibrar la cámara, detectar los marcadores, estimar la posición de la cámara respecto a ellos y guardar las posiciones absolutas del vehículo para poder realizar el mapeo de la trayectoria del vehículo (ver **Algoritmo 2**).

**Algoritmo 2:** Secuencia del algoritmo de localización

```
1   Calibrar cámara
2   Cargar parámetros intrínsecos y extrínsecos
3   repetir
4       Detectar el marcador fiduciario
5       Estimar la posición del marcador respecto a la cámara mediante funciones de
        Matlab como readAprilTag. Se obtienen los siguientes parámetros:
6           Matriz de rotación R
7           Vector de posición T
8       Pasar de la matriz de rotación R a ángulos de Euler XYZ
9       Obtener la posición del marcador respecto de la cámara en el sistema de
        referencia móvil
10      Calcular la posición de la cámara en el sistema de referencia fijo
11      Guardar la posición absoluta del vehículo
12      Graficar la posición del vehículo sobre un mapa del entorno
12  hasta (finalización del recorrido)
14  fin
```

Para desarrollar el algoritmo de localización, el objetivo será localizar donde se encuentra el vehículo en cada instante de muestreo respecto al origen del sistema de referencia fijo $(0,0)$ y mapear su posición en un mapa predefinido del entorno como el que se muestra en la **Figura 19**.

Sin embargo, siempre que se utilizan cámaras en cualquier sistema, por ejemplo, en los sistemas de conducción autónoma de vehículos de exterior o interior, hay que asegurarse de que la cámara instalada en ese coche sea precisa, es decir, que la cámara tenga alta precisión, resolución y poca o ninguna distorsión.

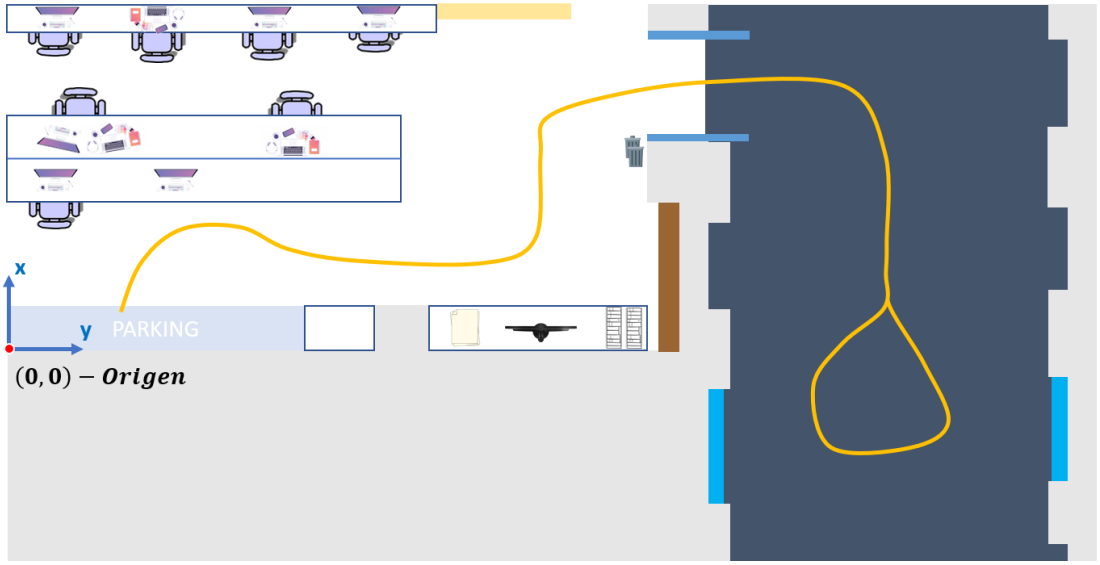



Figura 19. Entorno donde se mueve el vehículo.

Alta precisión significa que la cámara debe proporcionar la misma salida para la misma entrada en todo tipo de condiciones ambientales. Alta resolución significa que la cámara debe ser capaz de descifrar el menor cambio posible. Finalmente, una baja distorsión, ayuda a lograr una representación más precisa del mundo real en las imágenes capturadas.

Por lo tanto, la cámara debe ser calibrada correctamente para lograr una mayor precisión y que los valores empleados para el algoritmo de localización tengan el menor error posible. La calibración de la cámara se detalla en el siguiente apartado.

### 4.3.2 Calibración de la cámara

Una cámara es una parte integral de varios dominios como la robótica o la exploración espacial. La cámara está jugando un papel importante últimamente, debido a que ayuda a capturar todos y cada uno de los momentos y es útil para muchos análisis. Para utilizar la cámara como sensor visual, se deben conocer los parámetros de la cámara. Estos parámetros son la matriz de calibración y la matriz de distorsión de la cámara, y son necesarios para determinar una relación precisa entre un punto 3D en el mundo real y su correspondiente proyección 2D (píxel) en la imagen capturada por esa cámara calibrada. Estos se estiman mediante el proceso de calibración de la cámara.

Normalmente, calibrar la cámara significa encontrar dos tipos de parámetros:

- Parámetros internos (o intrínsecos) del sistema de cámara/lente. Por ejemplo, la distancia focal, el centro óptico y los coeficientes de distorsión radial de la lente.
- Parámetros externos (o extrínsecos). Se refiere a la orientación (rotación y traslación) de la cámara con respecto a algún sistema de coordenadas mundial.

Los parámetros de la cámara son necesarios para poder obtener la localización del robot y calcular su posición exacta en el espacio. El proceso por el que la cámara mapea el punto de coordenadas en el mundo 3D al plano 2D de la imagen puede describirse utilizando el modelo

estenopeico[1]. La relación entre el marco de coordenadas del píxel $(x_i, y_i)$ y el marco de coordenadas del mundo real $(X, Y, Z)$ puede describirse mediante:

$$Z \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (6)$$

Donde $f_x$ y $f_y$ son la distancia focal de la cámara, $c_x$ y $c_y$ son la posición del centro óptico de la cámara, y $K$ es la matriz de calibración de la cámara.

Al mismo tiempo, la matriz de distorsión es necesaria para obtener la calibración correcta de la matriz de transferencia. En primer lugar, la distorsión radial se puede corregir mediante:

$$x_{radial} = x_d \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \quad (7)$$

$$y_{radial} = y_d \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \quad (8)$$

En segundo lugar, la distorsión tangencial se puede corregir mediante:

$$x_{tangencial} = 2p_1 x_d y_d + p_2 (r^2 + 2x_d^2) \quad (9)$$

$$y_{tangencial} = p_1 \left(r^2 + 2y_d^2\right) + 2p_2 x_d y_d \quad (10)$$

Donde $(x_d, y_d)$ son los píxeles distorsionados, $r = \sqrt{x_d^2 + y_d^2}$ y $[k_1, k_2, k_3, p_1, p_2]$ es la matriz de coeficientes de distorsión.

Es conveniente obtener esta matriz mediante, por ejemplo, la herramienta de calibración de Matlab (Matlab Calibration Toolbox) o algún programa de OpenCV. En este caso se ha empleado un programa de Matlab [107], el cual ofrece un marco para la calibración (véase la **Figura 20**).

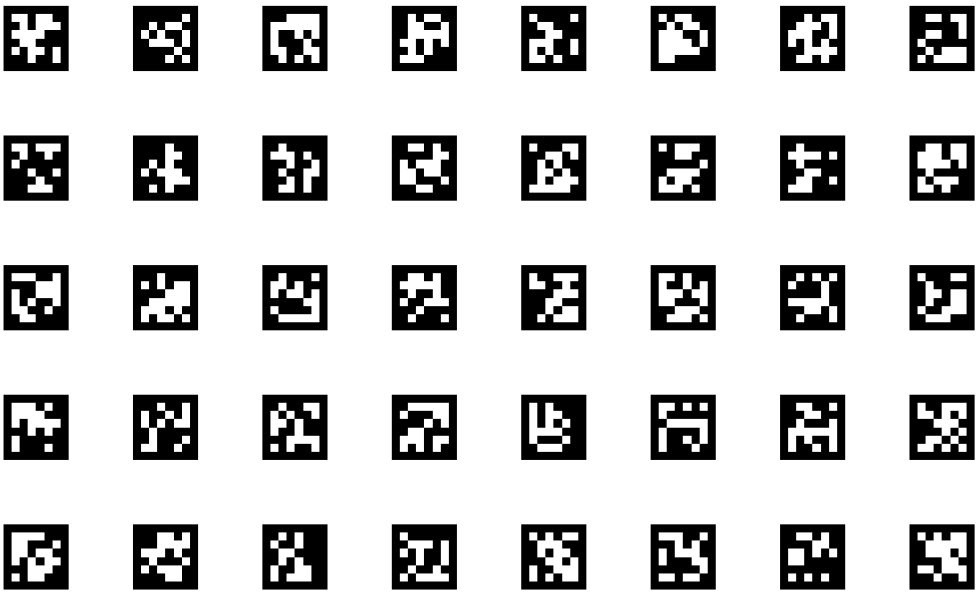


**Figura 20.** Tablero para la calibración de la cámara.

[1] La fotografía estenopeica utiliza una cámara estenopeica, que es aquella que no tiene sistemas ópticos (lentes u objetivos) basados en la refracción de la luz, siendo sustituidos por un orificio (que se llama estenopo), que es el encargado de formar la imagen.

El usuario lo único que debe hacer es capturar imágenes del patrón de calibración con diferentes puntos de vista, tal y como se observa en la **Figura 21**.

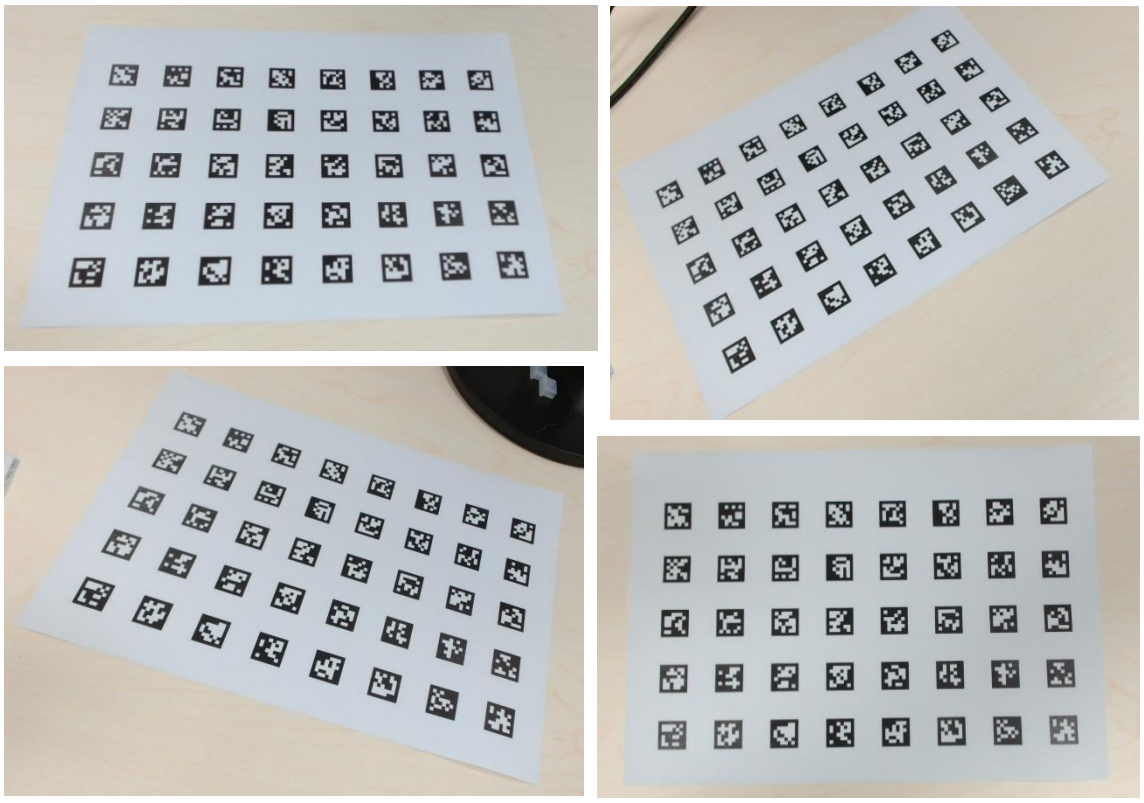


**Figura 21.** Imágenes obtenidas para la calibración de la cámara.

Estas imágenes, se pueden obtener manteniendo el patrón estático y tomando múltiples imágenes del mismo moviendo la cámara, o también se puede mantener la cámara constante y fotografiar el tablero en diferentes orientaciones. Las dos situaciones son similares desde el punto de vista matemático. El objetivo es tener al menos diez imágenes con diferentes distancias y orientaciones para conseguir unos resultados decentes.

Los patrones del tablero son distintos y fáciles de detectar en una imagen. Además, las esquinas de los cuadrados del tablero permiten una localizción sencilla porque tienen gradientes pronunciados en dos direcciones. Además, estas esquinas también están relacionadas por el hecho de que se encuentran en la intersección de las líneas del tablero. Todos estos aspectos se utilizan para localizar de forma robusta las esquinas de los cuadrados en un patrón de calibración.

Al finalizar se obtiene una variable en Matlab con los parámetros necesarios de la cámara para llevar a cabo el algoritmo de localización.

De esta forma, un algoritmo de calibración de cámara tiene las siguientes entradas y salidas:

- Entradas: Una colección de imágenes con puntos cuyas coordenadas 2D de la imagen y 3D del mundo son conocidas.
- Salidas: La matriz intrínseca de la cámara de 3×3, la rotación y la traslación de cada imagen, así como la matriz de distorisón.

Tras la calibración de la cámara es hora de estimar la posición del vehículo para desarrollar el algoritmo de localización.

### 4.3.3 Estimación de la posición del vehículo

El primer paso para la estimación de la posición del vehículo empleando marcadores fiduciarios, será detectar los propios marcadores.

En términos generales, el algoritmo de detección de marcadores funciona como sigue. En primer lugar, se capturan imágenes a gran velocidad y se analizan para detectar la presencia de un marcador. Este proceso se denomina segmentación. En segundo lugar, el ordenador descodifica la información de los marcadores en forma de 1s y 0s y determina la identificación única del marcador mediante referencias cruzadas con la biblioteca de marcadores. Se pueden encontrar más detalles sobre el algoritmo de detección en [108]–[110].

Después de la detección exitosa del marcador, las funciones de la biblioteca AprilTag de Matlab proporcionan información del marcador respecto a la posición relativa al centro óptico de la cámara. Para la rotación del sistema Matlab proporciona una matriz de transformación estándar $R$. A efectos de localización estos datos deben ser transformados a ángulos de Euler. Para la posición del vehículo se proporciona un vector $P$, con las distancias $(x, y, z)$ de la cámara respecto al marcador.

En este caso, se deben considerar tres sistemas de coordenadas: sistema de coordenadas del robot $(x_{rob}, z_{rob}, \theta_{rob})$, sistema de coordenadas de la cámara $(x_{cam}, z_{cam}, \theta_{cam})$ y sistema de coordenadas de la etiqueta $(x_{mar}, z_{mar}, \theta_{mar})$. La **Figura 22** muestra estos sistemas de referencia. Cabe destacar que el robot y la cámara tienen la misma orientación del sistema de referencia, debido a que la cámara se ha colocado horizontalmente para facilitar los cálculos.

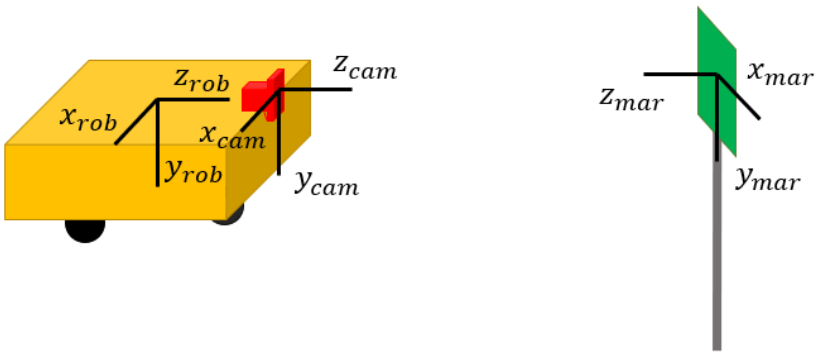



**Figura 22.** Sistemas de referencia del robot, cámara y marcador.

La posición relativa entre la cámara y el marcador puede determinarse mediante un total de seis componentes, tres de los cuales corresponden a la traslación $(x, y, z)$ y tres a la rotación (por ejemplo, los ángulos de $balanceo$, $cabeceo$ y $guiñada$), más conocidos por sus términos en inglés $(roll, pitch, yaw)$. Esto se representa en forma de una matriz de transformación homogénea $T$ en la que $R$ $(3x3)$ denota una matriz de rotación y $P$ $(3x1)$ denota una matriz de traslación (ecuación (11)). Sin embargo, dependiendo del tipo de aplicación, puede que sólo se necesiten algunos de estos seis componentes. Por ejemplo, en este caso, solo se emplearán las

distancias $x$ y $z$, debido a que exclusivamente se trabajará en el plano, y el ángulo de giro del marcador $pitch$.

$$T = \begin{bmatrix} R_{3x3} & P_{3x1} \\ 0 & 1 \end{bmatrix} \quad (11)$$

Asimismo, la orientación de la cámara se define como roll $(\psi)$, pitch $(\varphi)$ y yaw $(\theta)$ y se calcula de la siguiente manera a partir de la matriz de rotación $R$:

$$\psi = \text{atan}\left(\frac{r_{21}}{r_{11}}\right) \quad (12)$$

$$\varphi = \text{atan}\left(-\frac{r_{31}}{\sqrt{r_{32}^2 + r_{33}^2}}\right) \quad (13)$$

$$\theta = \text{atan}\left(\frac{r_{32}}{r_{33}}\right) \quad (14)$$

Donde $r_{ij}$ son términos de la matriz de rotación tal y como se muestra en la ecuación (15):

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (15)$$

Siguiendo el sistema de referencia de la **Figura 22** el ángulo de giro que interesa para este trabajo es el pitch $(\varphi)$, el cual se puede observar en la **Figura 23**.

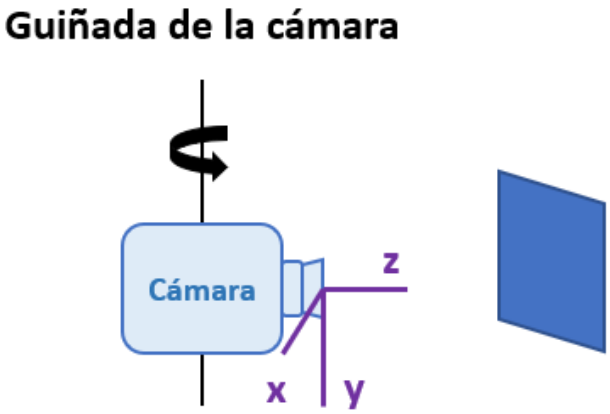



**Figura 23**. Ilustración del ángulo de giro del marcador.

En resumen, los parámetros que nos interesan son: las posiciones $(x, z)$ de la cámara respecto al marcador y la diferencia de giro entre ambos $(\theta)$. Sin embargo, estos parámetros están en el sistema de referencia de la cámara, y a nosotros nos interesarán respecto a un sistema de referencia absoluto $(x_v, y_v)$, los cuales se obtienen con las ecuaciones (16) y (17).

$$x_v = z * \sin(\theta) + x * \cos(\theta) \quad (16)$$

$$y_v = z * \cos(\theta) - x * \sin(\theta) \quad (17)$$

En la **Figura 24** se muestran estos parámetros de manera clara.

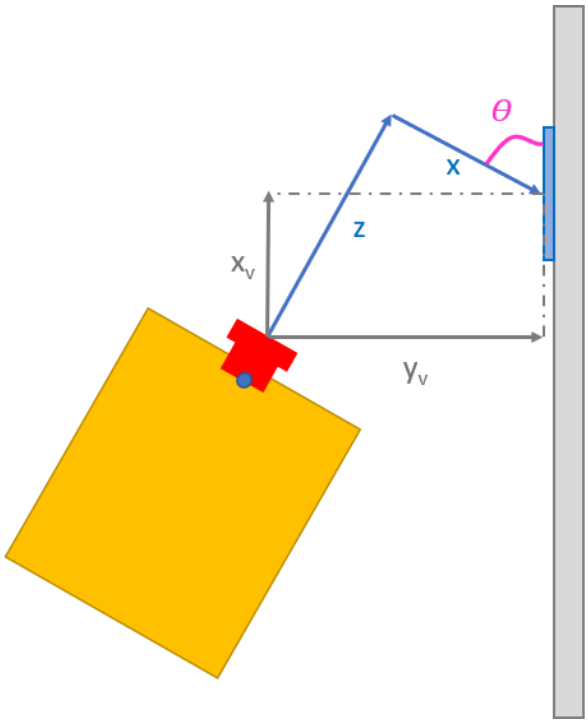



**Figura 24.** Determinación de la posición del robot respecto al marcador.

Por otro lado, la posición global del robot $(x_t, y_t)$ se calculará de acuerdo con la **Figura 25** a través del conocimiento de la posición del marcador $(x_m, y_m)$:

$$x_t = x_v \pm x_m \tag{18}$$

$$y_t = y_v \pm y_m \tag{19}$$

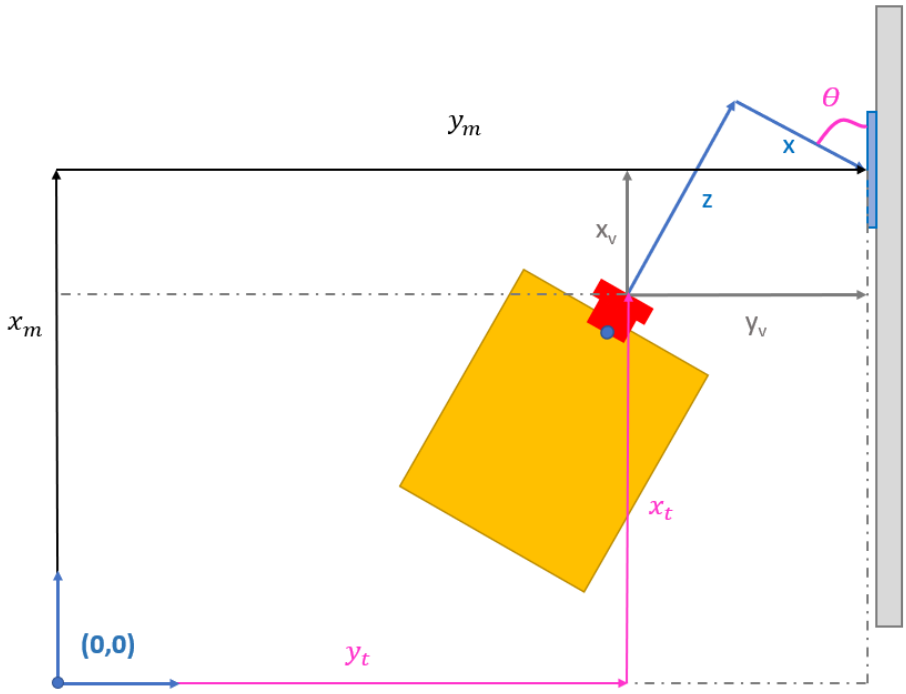



**Figura 25.** Determinación de la posición global del robot a partir de la posición del marcador.

Sin embargo, para que las ecuaciones (18) y (19) funcionen correctamente será necesario tener en cuenta la orientación del marcador, ya que los valores $(x_v, y_v)$ variarán en función de cómo estén estos colocados (véase **Figura 26**). Para este trabajo solo se tendrán en cuenta las cuatro orientaciones de la imagen. No obstante, es interesante considerar el ángulo de giro como se muestra en la ecuación (20). La cara del marcador se dibuja de color rosa.

$$\theta_{mar} = \theta - \theta_{offset} \tag{20}$$

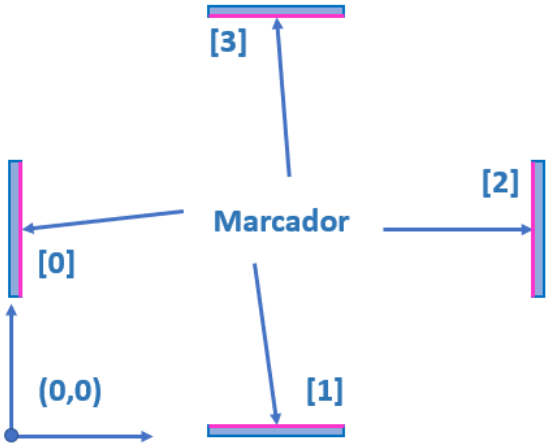



**Figura 26.** Diferentes orientaciones de marcadores.

Donde $\theta_{offset}$ depende de la posición del marcador (ver **Figura 27**).

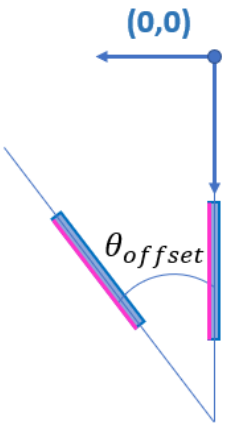



**Figura 27.** Ilustración ángulo offset del marcador.

En la **Tabla 11** se muestran los desfases con los que se ha trabajado en este proyecto, así como su referencia a la hora de definir cada marcador.

**Tabla 11.** Desfase en función de la colocación del marcador.

| Número de orientación del marcador | $\theta_{offset}$ |
|---|---|
| 0 | 0 |
| 1 | 90 |
| 2 | 180 |
| 3 | 270 |

Por lo que reescribiendo las ecuaciones iniciales quedan:

$$x_v = z * \sin(\theta_{mar}) + x * \cos(\theta_{mar}) \quad (21)$$

$$y_v = z * \cos(\theta_{mar}) - x * \sin(\theta_{mar}) \quad (22)$$

Por último, se debe hacer referencia a que el sistema de detección de marcadores AprilTag de Matlab es capaz de encontrar más de un marcador en una imagen determinada. Entonces, es posible calcular la posición del robot basándose en la posición de cada marcador. Para afinar la posición resultante, se puede utilizar la media de los datos obtenidos. Por otro lado, si el sistema encuentra un marcador que no está incluido en el conjunto de marcadores conocidos (con posiciones conocidas en un vector predefinido) no lo tendrá en cuenta para el cálculo de la posición del robot, es decir, se considerará que el algoritmo se ha equivocado.

### 4.3.4 Mapeo de la posición del vehículo

Un mapa es la representación del entorno en el que opera el robot. Un mapa puede ser métrico, topológico o híbrido. Se genera utilizando las características de la estructura del entorno, los puntos de referencia o cualquier otro dato procedente de las observaciones y la detección [48].

El mapeo es el proceso de creación del mapa del entorno. Se realiza de forma manual (por una persona), automática (por el robot durante una fase de aprendizaje, normalmente controlado por un operador) o autónoma (por el robot solo durante su funcionamiento). Aunque el mapeo puede ser una tarea independiente, a menudo se combina con el problema de la localización, ya que los mapas resultantes suelen ser más precisos en comparación con la resolución de ambos de forma independiente [111]. El mapeo es otra de las cuatro tareas principales del problema de la navegación [48].

Como se ha comentado en el aparatado de *Estado del Arte*, el SLAM es una solución para localizar el robot y crear un mapa del entorno simultáneamente. Para que el robot se localice en el entorno, necesita un mapa de referencia para poder definir su posición, mientras que para crear un mapa del entorno el robot necesita saber dónde está y dónde están las observaciones en relación con él mismo. Como la precedencia de estas dos tareas es paradójica, la solución más adoptada es tratar ambos problemas simultáneamente. Los métodos de SLAM visual (VSLAM) son soluciones que utilizan una secuencia de imágenes proporcionadas por una o varias cámaras. Algunos autores utilizan también el término TAM, por Tracking And Mapping (Seguimiento y Mapeo), que es la técnica de localización que se empleará en este proyecto.

Para ello, se llevará un histórico de las posiciones del vehículo, es decir, se hará un seguimiento de la trayectoria. Para ello, sobre un mapa predefinido se seguirá la trayectoria del vehículo en tiempo real. Con este fin, se colocarán marcadores por todo el entorno con una posición definida y se calculará, en primer lugar, la posición del vehículo respecto al marcadores detectados (sistema local), para posteriormente calcular su posición absoluta en el entorno (sistema global).

Para construir mapas de navegación consistentes, se requiere una buena localización. Para la mayoría de los robots comerciales, la información odométrica no es lo suficientemente precisa para una localización razonable. Con el tiempo, el odómetro tiende a acumular errores sin límite. Además, como se ha comentado, exclusivamente se hará uso de información visual capturada por las cámaras dispuestas sobre el robot para desarrollar el sistema de navegación. Para lograr

la información para el algoritmo de localización se hará uso de la cámara utilizada para la detección de obstáculos.

Por otro lado, como se ha mencionado a lo largo de este apartado, se ha utilizado un método de localización basado en puntos de referencia característicos del entorno que pueden ser detectados por la cámara del robot. Sin embargo, la localización puede no ser completamente precisa, debido a la no óptima calibración de la cámara, errores en las posiciones de los marcadores en el mapa o la imprecisa estimación de la pose debido a, por ejemplo, que el vehículo está en circulación y la cámara puede sufrir ligeros movimientos, o que la iluminación de todos los marcadores no es idéntica.

Por consiguiente, para realizar el mapeo de la posición del vehículo, se lleva a cabo un filtrado de las posiciones estimadas. Este filtrado consiste en realizar una media móvil de las posiciones que se han calculado con el algoritmo de localización tras descartar aquellas posiciones estimadas que difieran significativamente de la posición calculada anterior. Es decir, en primer lugar, se eliminan aquellos valores que no hayan sido correctamente estimados (ecuación (23)), y después, se realiza la media móvil de $N$ elementos con la ecuación (24).

- **Descarte posiciones incorrectas:**

$$Pose_i = \begin{cases} Descarte & \text{si } Pose_{i-1} - umbral > Pose_i \\ Pose_i & \text{si } Pose_{i-1} - umbral < Pose_i < Pose_{i-1} + umbral \\ Descarte & \text{si } Pose_i > Pose_{i-1} + umbral \end{cases} \quad (23)$$

- **Media móvil:**

$$Pose_{media} = \sum_{i}^{i+N} Pose_i \quad (24)$$

De esta forma, es posible obtener un mapa más preciso de la posición del vehículo que indique cual ha sido realmente su trayectoria real.

# Análisis de los resultados 5

En este apartado, se evaluarán los algoritmos de navegación, detección de obstáculos y localización desarrollados en el apartado 4. En primer lugar, se comentarán los resultados de cada algoritmo por separado, para finalmente, mostrar los resultados del conjunto completo. Por un lado, en los dos primeros algoritmos, al emplear redes neuronales convolucionales, se evaluará el rendimiento de las CNN diseñadas en términos de precisión y tiempo de ejecución. Además, se compararán con otras CNN predefinidas para determinar cuál es la que mejores prestaciones ofrece para la solución propuesta y se explicarán cuáles han sido las opciones de entrenamiento escogidas para entrenar estos algoritmos de aprendizaje profundo. Por otro lado, para evaluar el rendimiento del algoritmo de navegación, se mostrarán diferentes pruebas del algoritmo variando parámetros de filtrado, de manera que se establezca la mejor configuración para obtener los resultados más concisos. Igualmente, se comentarán varios cambios que se han hecho al vehículo, y al entorno, para mejorar los resultados, y se mostrarán imágenes del entorno real donde se han llevado a cabo los experimentos.

Sin embargo, antes de comenzar con los resultados de los algoritmos es necesario hacer referencia a todos aquellos aspectos que han sido necesarios para llevar a cabo la solución. En primer lugar, para programar todo el código (el cual se puede observar en el Anexo II) se ha empleado Matlab. En segundo lugar, se ha empleado un AGV el cual está controlado por un PLC Beckhoff. A tal fin, se dispone de su respectivo programa en TwinCAT, con el que se gobiernan las ruedas del vehículo, y con el que es necesario comunicarse a través del protocolo de comunicación ADS (Automation Device Specification). Esta conexión permite leer y escribir el código escrito en TwinCAT, logrando el objetivo de controlar el AGV desde un programa externo. Asimismo, y como se ha comentado a lo largo del documento, se han empleado dos cámaras para llevar a cabo las tareas de navegación, detección de personas y localización. Por un lado, se ha utilizado la webcam Logitech C310, para el seguimiento de la trayectoria, mientras que para la detección de obstáculos y localización se ha empleado la webcam Logitech C920s. El AGV (con el montaje de las cámaras) empleado para el desarrollo de este trabajo se puede observar en la **Figura 28**.

## 5.1 Algoritmo de navegación

En esta sección, primero se evalúa el rendimiento de la red neuronal propuesta en términos de precisión y rapidez de ejecución. El tiempo de ciclo varía en función de las características del ordenador, pero la relación entre estos tiempos más o menos se mantendría en otro tipo de hardware. Además, se presentan los resultados cuantitativos obtenidos por diferentes modelos de redes neuronales convolucionales que se pueden encontrar en la literatura. Por último, se evalúa el rendimiento del sistema de seguimiento de líneas en un entorno interior real.

**Figura 28.** AGV empleado para el desarrollo de este trabajo.

Para implementar las redes neuronales convolucionales se ha empleado la aplicación Deep Network Designer en Matlab. Además, las opciones de entrenamiento de las CNNs pueden verse en la **Tabla 12**.

**Tabla 12.** Opciones de entrenamiento del algoritmo de navegación.

| **Opciones de entrenamiento** | **Valor** |
|---|---|
| Tamaño del mini lote | 128 |
| Épocas Máximas | 200 |
| Tasa de aprendizaje inicial | 0,001 |
| Factor de caída de la tasa de aprendizaje | 0,1 |
| Periodo de caída de la tasa de aprendizaje | 20 |
| Mezcla o Baraja | Cada Época |
| Frecuencia de validación | Número de datos de entrenamiento / Tamaño del mini lote |
| Entorno de ejecución | GPU |

Los resultados obtenidos para las diferentes CNN se muestran en la **Tabla 13**. Los resultados se miden, como se ha comentado, en términos de precisión y tiempo de ejecución del algoritmo. La CNN con mejores resultados en términos de precisión es ResNet 18, mientras que la CNN con mejores resultados en términos de tiempo es MobileNet v2. En consecuencia, y dado que la precisión no varía considerablemente, la CNN utilizada para realizar las pruebas de rendimiento fue MobileNet v2. Las MobileNets se basan en una arquitectura racionalizada que utiliza

convoluciones separables en profundidad para construir redes neuronales profundas de poco peso. Son especialmente útiles para aplicaciones de visión móviles e integradas.

Por otro lado, cabe destacar que el tiempo de ejecución de la CNN de múltiples salidas es dos veces menor que el tiempo de ciclo obtenido con dos CNNs por separado, ya que, en vez de ejecutar dos CNNs para dos tareas separadas, se ejecuta una única CNN para dos tareas. Esto permite ahorrar tiempo de ejecución en sistemas donde el tiempo es de vital importancia.

**Tabla 13.** Comparación de las diferentes redes neuronales preentrenadas.

| **Red Base** | **Precisión (%)** | | | **Tiempo de ciclo (ms)** |
|---|---|---|---|---|
| | Regresión | | Clasificación | |
| | x | y | | |
| **MobileNet v2** | 95,23 | 96,70 | 99,00 | 8,43 |
| **ResNet 18** | 96,88 | 98,72 | 99,14 | 9,14 |
| **ResNet 50** | 95,23 | 96,33 | 99,00 | 12,02 |
| **EfficentNet** | 95,23 | 96,70 | 96,85 | 15,64 |
| **Inception** | 95,78 | 98,17 | 96,28 | 15,04 |

Por último, se han llevado a cabo varios experimentos en interiores en nuestro entorno de laboratorio, donde se ha colocado una cinta amarilla en el suelo como camino que debe seguir el vehículo. El AGV debe recorrer todo el trayecto y detenerse al final de este. La viabilidad del algoritmo se ha evaluado demostrando la navegación autónoma del vehículo en escenarios reales de interior. Un experimento se considera exitoso si el AGV consigue realizar todo el trayecto de forma completamente autónoma. Basándonos en este método, el vehículo se ha puesto en marcha unas 50 veces y ha resuelto correctamente el recorrido en casi todas las pruebas, lo que ha supuesto una tasa de éxito del 90%, si la velocidad era moderada (hasta aproximadamente 0,75 m/s). Además, el vehículo fue llevado a otro entorno en el que el aspecto del suelo variaba considerablemente y completó las rutas correctamente. Cabe destacar que este algoritmo de seguimiento de líneas es mucho más rápido que otros algoritmos basados en el aprendizaje profundo, como la segmentación semántica. La alta tasa de éxito del método demuestra una buena generalización y su viabilidad para entornos prácticos no vistos.

Los parámetros empleados para el control del vehículo son:

**Tabla 14.** Parámetros empleados para el control del vehículo.

| **Parámetro** | **Descripción** | **Valor** | **Unidades** |
|---|---|---|---|
| Kp | Ganancia proporcional | 6 | |
| Kd | Ganancia derivativa | 12 | |
| Kv | Ganancia de la velocidad | 10 | |
| Vmax | Velocidad máxima | 14 | % de la velocidad máxima del vehículo[2] |
| Vmin | Velocidad mínima | 4 | % de la velocidad mínima del vehículo |

[2] La velocidad máxima permitida por las ruedas del AGV es 1,8 m/s.

Asimismo, se ha establecido un umbral de detección de “No Línea” en 5 imágenes. Es decir, es necesario predecir que 5 imágenes son “No Línea” para detener el vehículo. De esta manera se evitan falsas detecciones, y como el algoritmo se ejecuta rápidamente no supone un peligro ni para el vehículo, ni para el entorno.

En la **Figura 29** se pueden observar varios ejemplos de la trayectoria del vehículo.

**Figura 29.** Parte de la trayectoria del vehículo.

Por último, y a modo de resumen, en la **Figura 30** se muestra el diagrama de bloques del algoritmo de navegación de seguimiento de líneas.

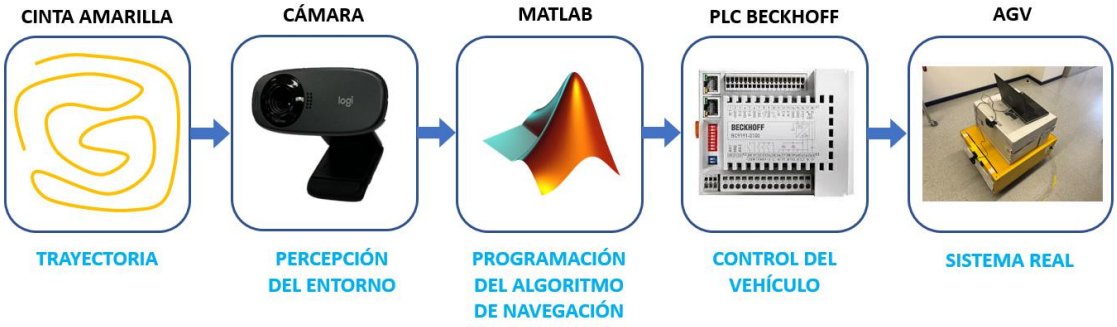



**Figura 30.** Diagrama de bloques del algoritmo de seguimiento de líneas.

## 5.2 Algoritmo de detección de obstáculos

La implementación del algoritmo de detección de obstáculos se ha implementado al igual que el algoritmo de navegación con la aplicación Deep Network Designer de Matlab. La cámara se ha colocado horizontalmente a aproximadamente 1 metro de altura. De esta manera es capaz de detectar si alguna persona está obstruyendo su camino y detenerse en caso de que sea necesario.

Como se ha comentado anteriormente para mejorar el rendimiento del algoritmo se han empleado técnicas de aumento de datos predefinidas de Matlab. Un resumen de las técnicas utilizadas se puede observar en la **Tabla 15**. El aumento de datos es un conjunto de técnicas para aumentar artificialmente la cantidad de datos generando nuevos puntos de datos a partir de los existentes. Esto incluye realizar pequeños cambios en los datos o utilizar modelos de aprendizaje profundo para generar nuevos puntos de datos. Esto permite mejorar los resultados de los algoritmos de aprendizaje profundo, sobre todo cuando se dispone de un conjunto de datos pequeño.

Tabla 15. Aumento de datos para la detección de obstáculos.

| Variable | Valores | Descripción |
|---|---|---|
| Rotación | [-20, 20] | Rango de rotación, en grados, aplicado a la imagen de entrada. |
| Traslación en X | [-10, 10] | Rango de traslación horizontal aplicado a la imagen de entrada. La distancia de traslación se mide en píxeles. |
| Traslación en Y | [-10, 10] | Rango de traslación vertical aplicado a la imagen de entrada. La distancia de traslación se mide en píxeles. |

Por otro lado, al igual que en el apartado anterior, es necesario definir unas opciones de entrenamiento para la CNN. Estas se pueden observar en la Tabla 16.

Tabla 16. Opciones de entrenamiento para el algoritmo de detección de obstáculos.

| Opciones de entrenamiento | | Valor |
|---|---|---|
| Mini Batch Size | Tamaño del mini lote | 16 |
| Max Epochs | Épocas Máximas | 20 |
| Initial Learn Rate | Tasa de aprendizaje inicial | 0,001 |
| Learn Rate Drop Factor | Factor de caída de la tasa de aprendizaje | 0,1 |
| Shuffle | Mezcla o Baraja | Cada Época |
| Validation Frequency | Frecuencia de validación | Número de datos de entrenamiento / Tamaño del mini lote |
| Execution Environment | Entorno de ejecución | GPU |

En esta sección también se compararán diferentes redes neuronales para determinar cuál es la que mejores resultados nos puede ofrecer para nuestro algoritmo. En este caso, también se compararán en términos de precisión y tiempo de ejecución, aunque se añade información relacionada con el número de parámetros de la red y el tiempo necesario para su entrenamiento con las opciones comentadas en la Tabla 16. Los nuevos resultados se pueden observar en la Tabla 17.

Observando los resultados de la Tabla 17, y al igual que en el caso anterior, se ha decidido que como la precisión entre redes no varía considerablemente, la red que se emplee como algoritmo de detección de obstáculos será aquella que más rápido se ejecute. En este caso es la CNN Places365. Place365-GoogLeNet es un modelo preentrenado de CNN el cual ha sido entrenado con el conjunto de datos Places365 y tiene la misma arquitectura de red subyacente que GoogLeNet la cual fue entrenada con el conjunto de datos ImageNet.

Otra técnica empleada para valorar el rendimiento de los algoritmos de aprendizaje profundo es la matriz de confusión. Una matriz de confusión es una técnica para resumir el rendimiento de un algoritmo de clasificación. El cálculo de una matriz de confusión puede dar al usuario una mejor idea de lo que su modelo de clasificación está acertando y qué tipos de errores está cometiendo.

**Tabla 17.** Resultados del algoritmo de detección de obstáculos.

| Red | Parámetros | Precisión | Tiempo de entrenamiento | Tiempo de ejecución (ms) |
|---|---|---|---|---|
| ResNet 18 | 11,7 M | 99,41 % | 11 min 26 s | 60,3 |
| ResNet 50 | 25,6 M | 99,41 % | 49 min 29 s | 152,4 |
| ShuffleNet | 1,40 M | 99,61 % | 23 min 45 s | 50,4 |
| GoogleNet | 7,0 M | 99,41 % | 15 min 32 s | 23,3 |
| Places365 | 7,0 M | 99,61 % | 15 min 51 s | 15,4 |
| EfficentNet | 5,31 M | 100,0 % | 86 min 05 s | 72,5 |
| MobileNet v2 | 3,5 M | 99,80 % | 33 min 26 s | 21,7 |
| Inception v3 | 23,9 M | 99,41 % | 53 min 49 s | 23,4 |
| Xception | 22,9 M | 99,61 % | 50 min 20 s | 29,8 |
| VGG-19 | 144 M | 99,80 % | 162 min 49 s | 21,6 |
| DenseNet | 20,0 M | 99,80 % | 198 min 23 s | 87,0 |

En el gráfico de la matriz de confusión, las filas corresponden a la clase predicha (clase de salida) y las columnas a la clase verdadera (clase objetivo). Las celdas diagonales corresponden a las observaciones correctamente clasificadas, mientras que las celdas fuera de la diagonal corresponden a las observaciones clasificadas incorrectamente. En cada celda se muestra tanto el número de observaciones como el porcentaje del número total de observaciones.

La matriz de confusión del algoritmo desarrollado se puede observar en la **Figura 31**.

Analizando la matriz de confusión, las dos primeras celdas diagonales muestran el número y el porcentaje de clasificaciones correctas de la red entrenada. Por ejemplo, 303 imágenes se clasifican correctamente como “Con obstáculo”. Esto corresponde al 59,3 % de las 511 imágenes. Del mismo modo, 236 casos se clasifican correctamente como “Sin obstáculo”. Esto corresponde al 40,3 % de todas las imágenes.

En general, el 99,6 % de las predicciones son correctas y el 0,4 % son erróneas. Esto demuestra que cuando el algoritmo falla, no detecta un obstáculo cuando sí debería. Esto reafirma el criterio de usar un umbral para evitar falsos positivos y que el vehículo se detenga cuando no deba.

Finalmente, este algoritmo ha sido correctamente implementado en el AGV, y es capaz de detectar correctamente personas y detenerse ante ellas. Sin embargo, tiene algunas limitaciones: no es capaz de detectar otros objetos que no sean peatones y solo es capaz de detectar personas delante del robot. Además, el robot simplemente se detiene ante el peligro y no es capaz de evitarlo, por lo que este algoritmo puede resultar insuficiente para otras aplicaciones. Sin embargo, sus aspectos positivos son que es rápido y eficaz para el trabajo que ha sido entrenado.

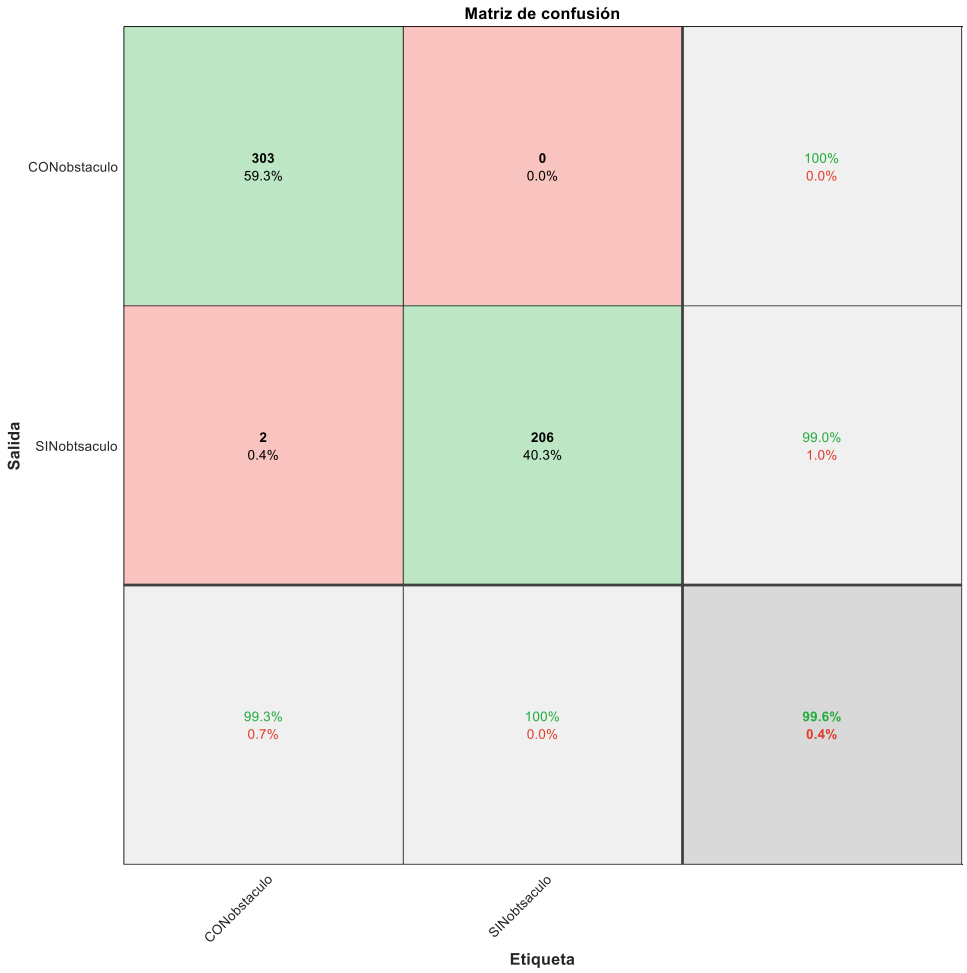



**Figura 31.** Matriz de confusión del algoritmo de detección de obstáculos.

## 5.3 Algoritmo de localización y mapeo

En este trabajo, además de los algoritmos de seguimientos de trayectoria y detección de obstáculos, se propone un método de seguimiento y mapeo del recorrido para un vehículo autónomo en entornos de interiores utilizando marcadores fiduciarios. El método propuesto se basa exclusivamente en la estimación de la postura de los marcadores fiduciales usando un algoritmo de visión artificial. Las principales ventajas de este enfoque pueden resumirse como sigue: en primer lugar, podemos construir un sistema TAM de una manera rentable porque el sistema propuesto requiere una sola cámara y marcadores fiduciarios fácilmente generados utilizando impresoras de escritorio. En segundo lugar, los puntos de referencia pueden instalarse de forma flexible y aleatoria sin ningún conocimiento previo, simplemente midiendo su posición en el espacio y asignándoles su respectiva orientación. Esto significa que no hay un proceso complicado de calibración y medición. En tercer lugar, el método propuesto tiene características de fácil asociación de datos y buena solidez.

Para probar la efectividad del método de seguimiento propuesto se ha llevado a cabo una implementación práctica de un sistema de seguimiento de un AGV en una parte de la Escuela de Ingeniería de Vitoria-Gasteiz. Como se ha comentado anteriormente, se han empleado marcadores fiduciarios para estimar la posición absoluta del robot. Estos marcadores están dispuestos por todo el entorno donde se mueve el robot y sus posiciones y orientaciones son

conocidas (ver Tabla 18). De este modo es posible calcular la posición absoluta del vehículo durante todo su recorrido tal y como se ha explicado en el apartado anterior.

Tabla 18. Posición marcadores en el Laboratorio.

| ID marcador | Posición (m) | | Orientación |
|---|---|---|---|
| | $x$ | $y$ | |
| **1** | 0,00 | 0,90 | 1 |
| **2** | 1,50 | 5,15 | 2 |
| **4** | 0,75 | 5,97 | 1 |
| **5** | -3,23 | 8,58 | 2 |
| **6** | 0,8 | 4,25 | 1 |
| **7** | 5,65 | 4,19 | 3 |
| **8** | 5,36 | 6,09 | 3 |
| **9** | 5,61 | 8,59 | 2 |
| **10** | -2,64 | 5,93 | 1 |
| **11** | -0.57 | 8,68 | 3 |
| **12** | 2,53 | 8,56 | 3 |
| **13** | 2,5 | 1,46 | 0 |
| **14** | 3,66 | 3,22 | 0 |
| **15** | 3,05 | 5,00 | 3 |
| **16** | 3,71 | 8,56 | 2 |
| **17** | 0,75 | 8,56 | 1 |
| **18** | 2,53 | 6,01 | 3 |

Por otro lado, si se dibujan los marcadores, junto con el mapa donde se moverá el vehículo y el recorrido que se debe seguir, se obtiene la Figura 32. De esta forma, se parte de un mapa preestablecido del entorno y solo es necesario dibujar la posición del vehículo.

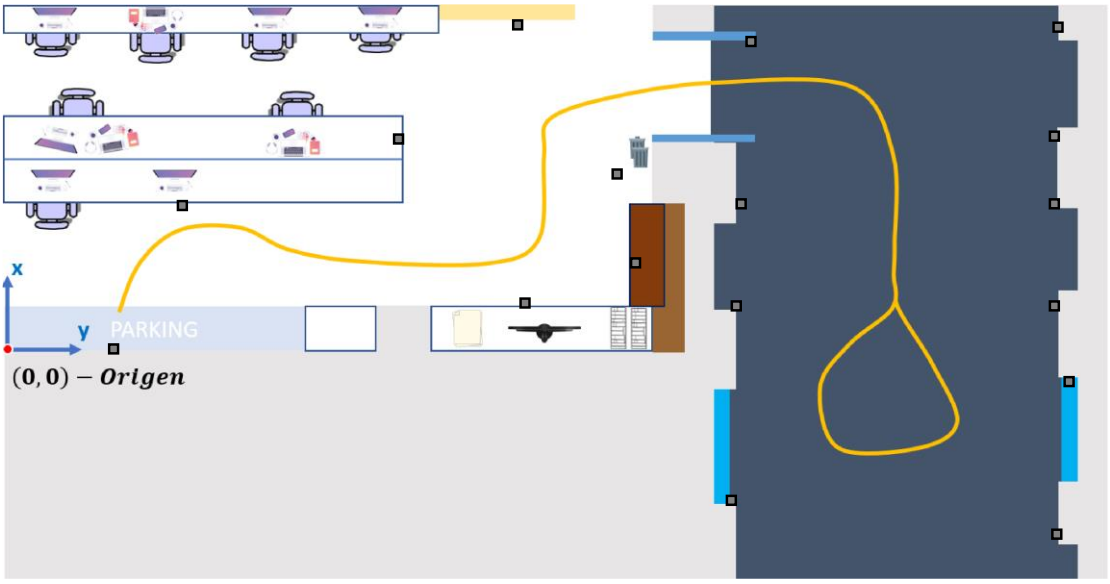



Figura 32. Posición de los marcadores y recorrido del vehículo.

Por otro lado, en la **Figura 33** se pueden observar imágenes reales del entorno donde se han llevado a cabo las pruebas correspondientes al algoritmo de localización.

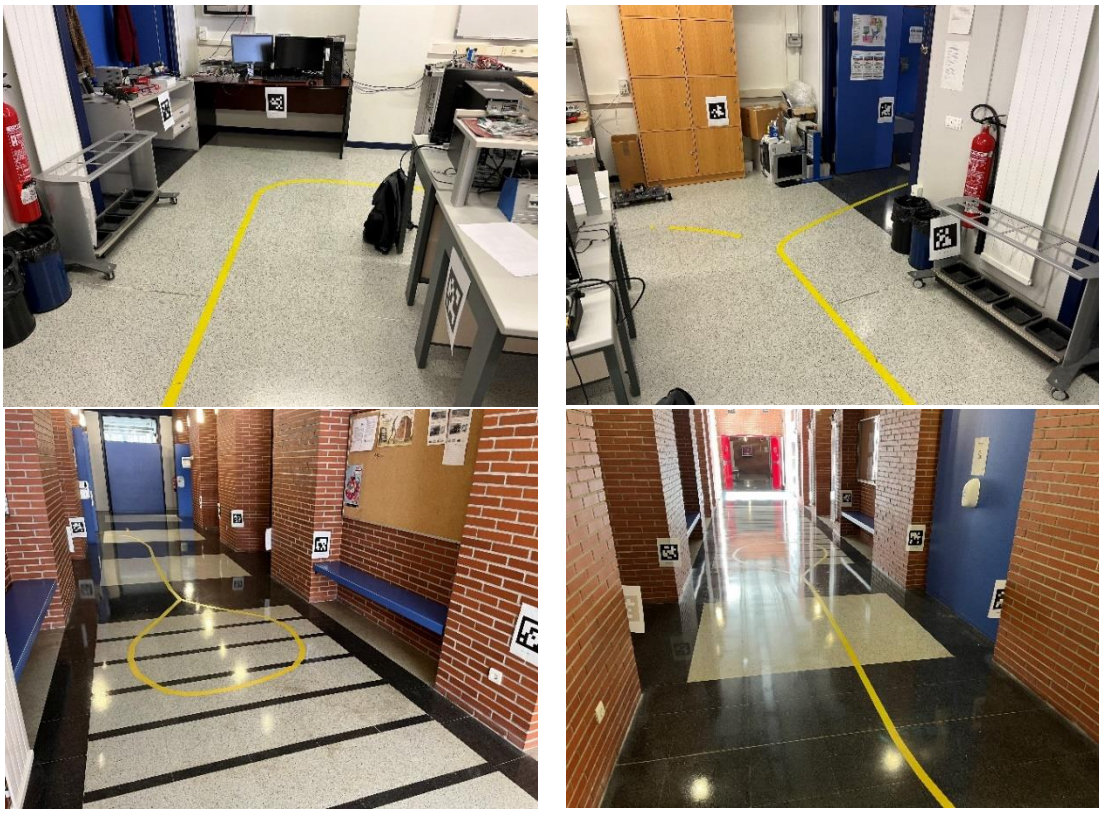


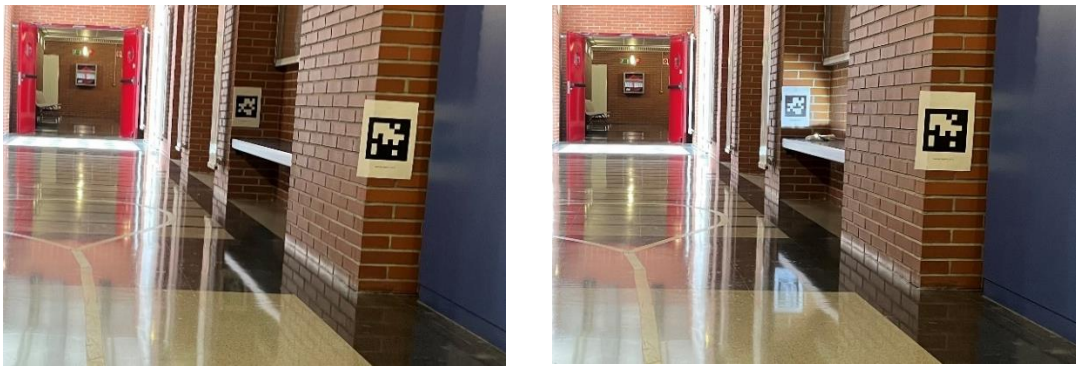
**Figura 33.** Imágenes del entorno donde se ha llevado a cabo la experimentación.

Uno de los aspectos más importantes que se han tenido que considerar para lograr que el algoritmo desarrollado funcione correctamente ha sido la intensidad de la luz en el entorno de trabajo. Para capturar una imagen y no disminuir la probabilidad de detección del marcador, la intensidad de la luz debe estar por encima de cierto nivel. Como se puede observar en la **Figura 33**, la iluminación varía considerablemente durante todo el recorrido, lo que puede llevar a una incorrecta detección del marcador y a una estimación de la posición que difiere considerablemente de la correcta. Como solución, se propone utilizar una fuente de luz adicional procedente de una linterna montada en el vehículo (véase **Figura 34** (a)), además de proveer de iluminación extra a aquellos marcadores que tengan baja luz (véase **Figura 34** (b)).

a) b)

**Figura 34.** Iluminación extra para el algoritmo de localización. A) Luz adicional montada sobre el vehículo. B) Luz adicional para el marcador con poca luz.

En la **Figura 35** se puede observar el marcador con la adición de luz. Esto permite que el resultado del algoritmo de seguimiento del vehículo ofrezca resultados más precisos sobre la posición del vehículo.



a) b)

**Figura 35.** Comparación del marcador sin (a) y con (b) luz adicional.

Por otro lado, y como se ha comentado en el apartado 4.3.4 es necesario realizar un filtrado de los datos de localización del vehículo para obtener un buen resultado para el seguimiento del vehículo en tiempo real. Por esta razón, en la **Figura 37** se muestran varios ejemplos del resultado que se obtiene en función de los valores del umbral de filtrado y de la media móvil.

Igualmente, es necesario encontrar el equilibrio entre estos parámetros para obtener un resultado satisfactorio: por ejemplo, si la media es muy baja, la señal tendrá mucho ruido (**Figura 37** (g)), mientras que si es muy alta se perderá información acerca del recorrido en aquellos puntos donde el vehículo no sea capaz de encontrar muchos marcadores (**Figura 37** (h)). Por otro lado, si el umbral es muy pequeño no será posible seguir toda la trayectoria del vehículo (**Figura 37** (e)). Por consiguiente, se ha decidido que un resultado correcto podría ser el de la **Figura 36**, donde el umbral se ha establecido a 1,5 y la media se realiza cada 20 posiciones.

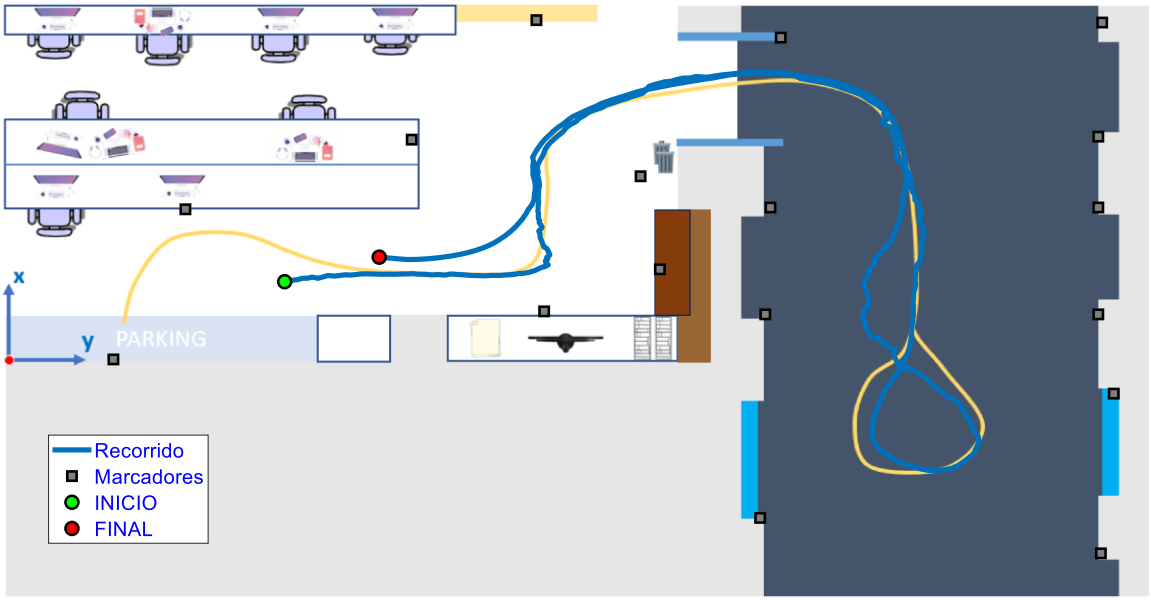
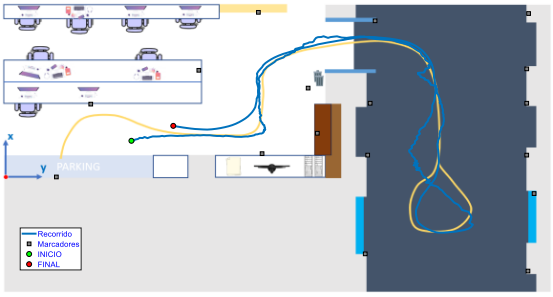
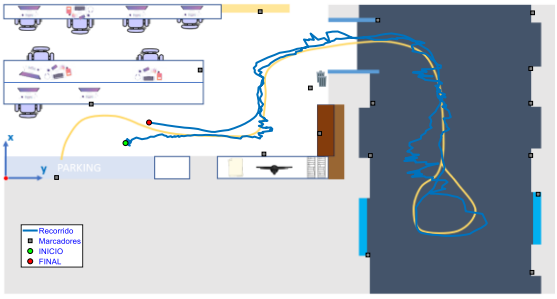



**Figura 36.** Ejemplo de algoritmo de localización exitoso. Umbral = 1,5 y Media = 20.

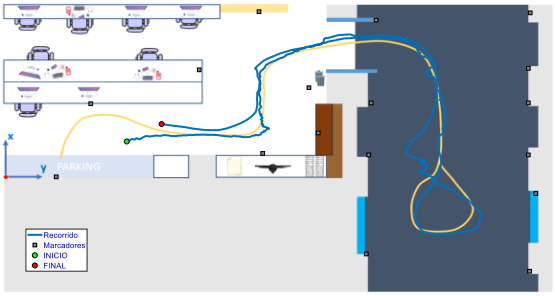
a) Umbral = 1.5 y Media móvil = 15



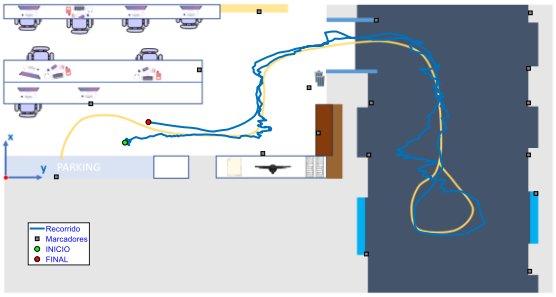

b) Umbral = 1.5 y Media móvil = 5



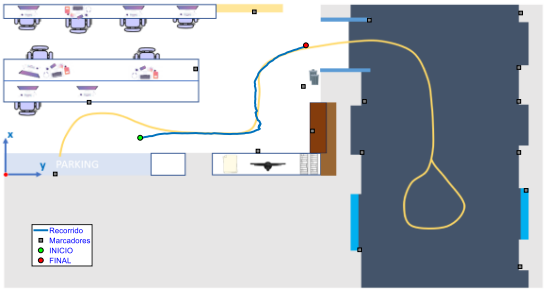

c) Umbral = 1 y Media móvil = 10



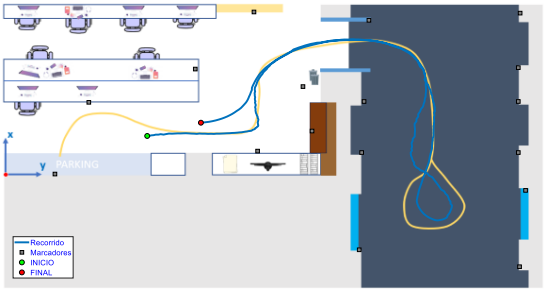

d) Umbral = 1 y Media móvil = 5



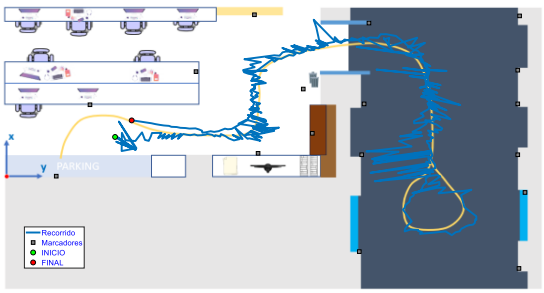

e) Umbral = 0.5 y Media móvil = 20




f) Umbral = 1 y Media móvil = 30




g) Umbral = 1 y Media móvil = 1




h) Umbral = 1.5 y Media móvil = 50

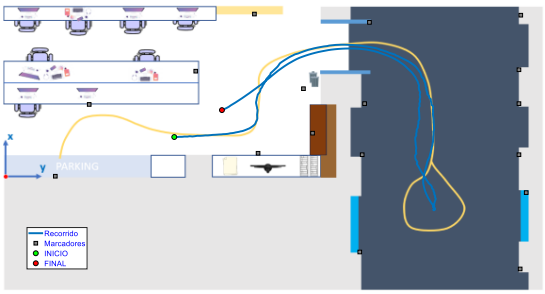



**Figura 37.** Ejemplos de filtrado para el algoritmo de seguimiento.

Para finalizar con este apartado en la **Figura 38** se muestra el diagrama de bloques del algoritmo de seguimiento del vehículo.

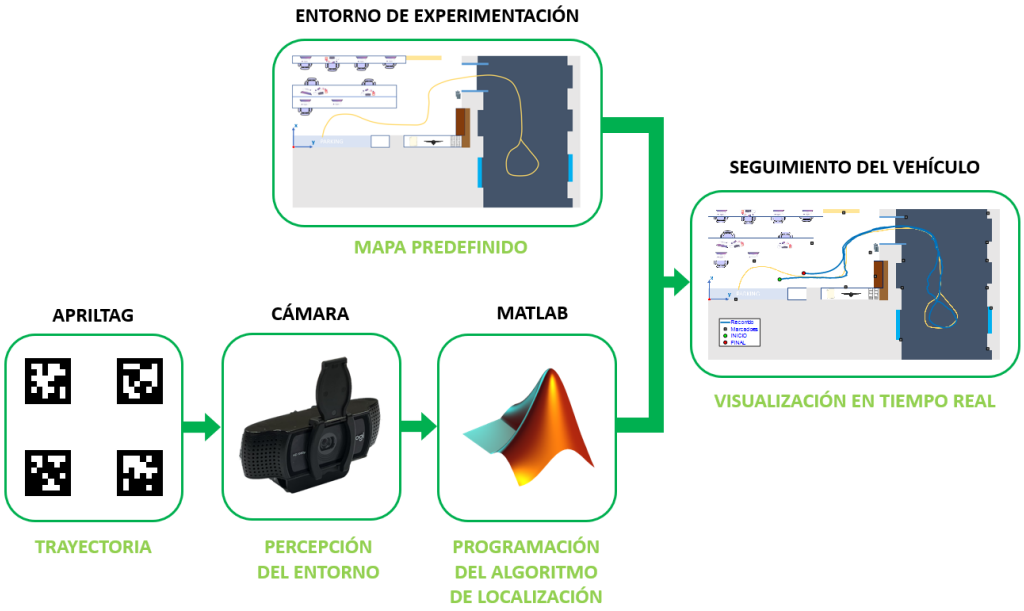



**Figura 38.** Diagrama de bloques del algoritmo de localización.

## 5.4 Sistema en conjunto

En este apartado se muestra una secuencia de imágenes correspondientes al sistema de navegación desarrollado en este proyecto. El conjunto de imágenes se puede observar en la **Figura 39**. En primer lugar, en las imágenes de la izquierda se muestran aquellas correspondientes al algoritmo de seguimiento de líneas. En el título de la imagen se puede observar si el algoritmo ha detectado o no línea. Además, en color rojo, se percibe el punto de referencia para el control del vehículo. En segundo lugar, en las imágenes del centro se muestran aquellas correspondientes al algoritmo de detección de obstáculos. En caso de que se detecten obstáculos, el vehículo se detendrá y se añadirá un marco rojo a la imagen. Finalmente, en la derecha, se muestran las imágenes del algoritmo de seguimiento del vehículo.

Por otro lado, en la primera imagen de localización, como no se ha llegado al número de elementos para poder realizar la media, no se muestra la posición del vehículo. Asimismo, cuando se llega al número de posiciones para realizar la media, la primera posición la marca con un punto verde que indica el inicio del recorrido. Cuando ha llegado al final, cierra el algoritmo de localización con un punto rojo.

## 5.5 Importancia del hardware en algoritmos de aprendizaje profundo

Con los recientes avances en el aprendizaje profundo, las redes neuronales son cada vez más grandes y pesadas, y se han convertido en una práctica habitual en muchas aplicaciones de aprendizaje automático. Su capacidad para alcanzar una precisión similar a la humana, e incluso mejor, las ha convertido en un hito en la historia de la Inteligencia Artificial. Sin embargo, alcanzar tales niveles de precisión es complicado en términos de potencia de cálculo. Por esta razón, es importante emplear un hardware adecuado para trabajar con este tipo de algoritmos. En esta última sección, se comparan los diferentes ordenadores empleados durante el desarrollo de este trabajo. En la **Tabla 19** se pueden observar las características de los ordenadores empleados para la realización de este proyecto.

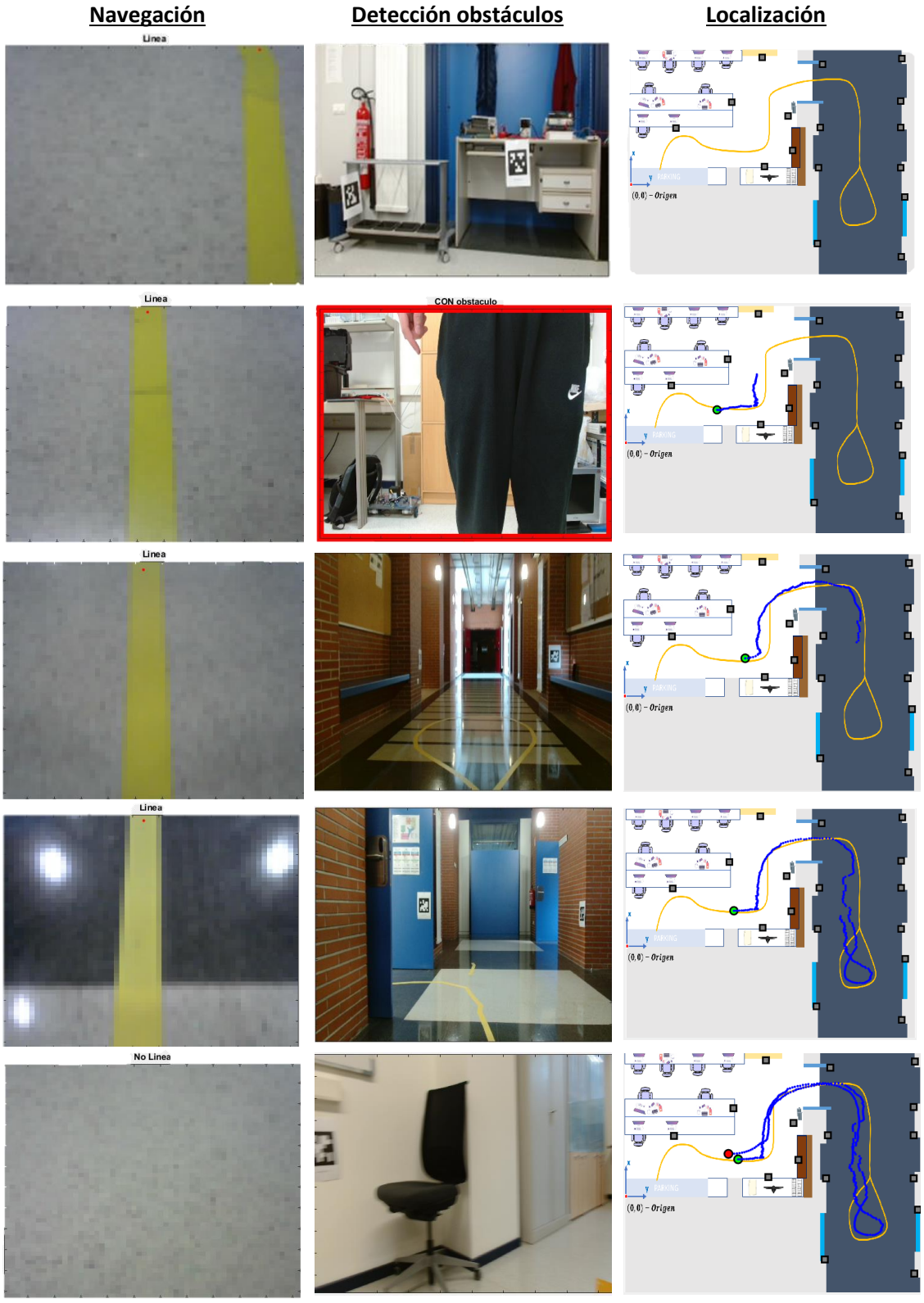



**Figura 39.** Secuencia de imágenes del algoritmo de navegación.

Tabla 19. HW empleado para este proyecto.

| Ordenador | Procesador | RAM | Gráfica |
|---|---|---|---|
| Ordenador 1 | Intel® Core™ i7-10510U CPU @ 1,80 GHz | 16 GB | NVIDIA GeForce MX330 |
| Ordenador 2 | AMD Ryzen 5 3600 6-Core Processor | 16 GB | NVIDIA GeForce GTX 1660 |
| Ordenador 3 | Intel® Core™ i9-9880H CPU @ 2,30 GHz | 16 GB | NVIDIA Quadro T1000 |

Por otro lado, los resultados mostrados tanto en la Tabla 13 como en la Tabla 17, se han logrado empleando el segundo ordenador. Sin embargo, para llevar a cabo los experimentos en laboratorio de la Escuela de Ingeniería se ha empleado el ordenador 3, debido a que es el que mejores prestaciones ofrece siendo un portátil. Una comparativa de las tres opciones de hardware (HW) se realiza en la Tabla 20.

Tabla 20. Comparación HW.

| Tarea | Ordenador 1 | Ordenador 2 | Ordenador 3 |
|---|---|---|---|
| Cargar ambas CNNs | 5,06 s | 4,32 s | 4,69 s |
| Ejecutar ambas CNNs | 0,21 s | 0,038 s | 0,064 s |

Como se puede observar en la Tabla 20, el ordenador que mejores resultados ofrece es el segundo. Sin embargo, este es un ordenador de sobremesa y no puede ejecutar el algoritmo de navegación y comunicarse con el AGV al mismo tiempo. Por otro lado, el resultado de ejecución de ambas CNNs que se obtiene con el ordenador tres, es aproximadamente cuatro veces inferior al tiempo de ciclo obtenido con el ordenador uno. Esta mejora de tiempos supone que el código total se ejecute más rápido y, por consiguiente, se analicen más imágenes por segundo, lo que aumenta el rendimiento de los algoritmos. Por un lado, se calculan más puntos de consigna, lo que supone un control más estable y preciso. Por otro lado, se analizan más imágenes de obstáculos, lo que significa que en caso de que exista obstáculo, el vehículo se detendrá más rápido, además de que se analizarán más puntos para el algoritmo de localización.

# Conclusiones y trabajos futuros

# 6

Este trabajo presenta un sistema de navegación para vehículos autónomos de interior. El sistema de navegación consiste, en primer lugar, en un método de seguimiento de líneas mediante una red neuronal convolucional multi-salida. En segundo lugar, se ha incorporado un algoritmo de detección de obstáculos basado en otra CNN. Finalmente, se realiza un seguimiento del vehículo en tiempo real mediante un algoritmo de localización a través de marcadores fiduciarios.

En lo que respecta al algoritmo de seguimiento de líneas, la técnica clave de nuestro sistema es integrar dos CNN con diferentes propósitos aunando los pesos de ambas redes, de modo que se puedan ejecutar dos tareas diferentes simultáneamente. El uso de imágenes de tamaño pequeño, además, permite unos tiempos de entrenamiento y ejecución reducidos.

Los resultados experimentales muestran que nuestra CNN con dos salidas puede realizar las tareas de clasificación y regresión de manera efectiva. Igualmente, nuestra red puede reducir eficazmente el tiempo de ejecución de otros algoritmos de visión artificial basados en aprendizaje profundo como la segmentación semántica.

En lo que respecta al algoritmo de detección de obstáculos, se ha demostrado que funciona perfectamente para realizar la tarea para el que ha sido entrenado. La clave es utilizar un conjunto de datos que defina perfectamente el problema, y emplear una red neuronal convolucional que se ejecute rápidamente.

El algoritmo diseñado, puede resultar insuficiente para realizar tareas más complejas, debido a que simplemente se detiene hasta que el obstáculo ha desaparecido. No obstante, para un vehículo que podría, por ejemplo, servir para mover mercancía en un almacén, es suficiente.

El algoritmo de localización diseñado no emplea técnicas de aprendizaje profundo, simplemente utiliza librerías de acceso abierto basadas en visión artificial a través de marcadores fiduciarios. Esto permite realizar un algoritmo de seguimiento del vehículo muy sencillo, pero efectivo. Simplemente es necesario colocar estos marcadores por el entorno de trabajo y medir donde están colocados.

A pesar de que este trabajo puede parecer tedioso o inviable para entornos de trabajo grandes, se podrían emplear a modo de referencia para localizar localmente el vehículo, y realizar una determinada tarea. Por ejemplo, en Boston Dynamics los emplean para localizar una puerta y abrirla con un robot bípedo [112] (segundo 44 del vídeo).

Las ventajas de este algoritmo son que es muy sencillo de ejecutar y simplemente es necesaria una cámara calibrada. A partir de ahí, se puede realizar un seguimiento tanto, de un vehículo de tierra, como de un vehículo aéreo.

Como hemos visto a lo largo de este proyecto, es posible utilizar la visión como sentido principal (o incluso como único) en un sistema de navegación completamente autónomo para robots móviles. Aunque otros sensores proporcionan información adicional que puede utilizarse para mejorar el rendimiento, existen soluciones basadas en la imagen para las cuatro tareas principales de un sistema de navegación.

Como comentan Yuri D. V. Yasuda et al. [48], el uso de la visión por sí sola tiene múltiples ventajas, como un hardware más sencillo y barato, la aptitud para ser utilizada en la mayoría de los entornos, un menor número de fuentes de flujo de datos, así como la disponibilidad de enfoques y soluciones procedentes de las áreas de procesamiento de imágenes y visión por ordenador. Como se ha demostrado en este proyecto, una sola cámara se puede emplear para localizar el vehículo y detectar posibles obstáculos al mismo tiempo, lo que permite enviar consignas de seguridad al robot y mapear su posición en tiempo real. Asimismo, si se emplean dos cámaras, se obtiene un sistema de navegación bastante completo.

Por otro lado, emplear técnicas de aprendizaje profundo es beneficioso para los vehículos autónomos. Actualmente, el aprendizaje profundo es la tecnología principal detrás de un gran número de sistemas de percepción. La disponibilidad de grandes cantidades de datos etiquetados de acceso abierto (la eficacia de los sistemas de aprendizaje profundo está directamente ligada a la disponibilidad de datos de entrenamiento), así como las diversas y numerosas referencias bibliográficas y recursos de internet, permiten desarrollar algoritmos complejos y obtener resultados con precisiones y rendimientos muy superiores a los obtenidos con otras técnicas hasta el momento.

A modo de conclusión, los siguientes pasos consistirán en mejorar el control completo del vehículo empleando exclusivamente algoritmos de visión por ordenador, ya que actualmente existen cámaras suficientes para este tipo de aplicaciones que tienen un precio más asequible que otros sensores como el LiDAR. Algunos de estos pasos consistirán en estudiar y desarrollar, por ejemplo, nuevas técnicas de navegación autónoma para el vehículo. Además, se quiere estudiar la posibilidad de desarrollar un algoritmo de evasión de obstáculos (ya sean estáticos o dinámicos), que además de identificarlo, sea capaz de bordearlo, y de encontrar de nuevo su trayectoria. Asimismo, se han comenzado a estudiar técnicas de localización que no solo sigan al vehículo, sino que realicen también un SLAM del entorno (como pueden ser los algoritmos de SLAM visual). Por último, todas las técnicas desarrolladas durante este trabajo se quieren poner en marcha en vehículos aéreos no tripulados, de modo que se pueda comprobar su viabilidad para diferentes técnicas de navegación.

# Referencias bibliográficas


[1] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,” 2018.

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[2] N. Taglic, “What are automated guided vehicles?,” 2021.

https://6river.com/what-are-automated-guided-vehicles/#:~:text=6 River Systems uses LiDAR,of a warehouse floor change.

[3] Wikipedia, “Vehículo de guiado automático.”

https://es.wikipedia.org/wiki/Vehículo_de_guiado_automático.

[4] Mecalux, “Así funcionan los robots AGV.”

https://www.mecalux.es/blog/robot-agv#:~:text=Los robots AGV se caracterizan,posean circuitos de transporte permanentes.

[5] R. Brown, “Role of Computer Vision in AI for Developing Robotics, Drones & Self-driving Cars.”

https://becominghuman.ai/role-of-computer-vision-in-ai-for-developing-robotics-drones-self-driving-cars-9c92b89d57c.

[6] K. Kersting, “Machine Learning and Artificial Intelligence: Two Fellow Travelers on the Quest for Intelligent Behavior in Machines,” *Front. Big Data*, vol. 1, no. November, pp. 1–4, 2018, doi: 10.3389/fdata.2018.00006.

[7] T. M. Mitchell, *Machine learning*. McGraw Hill Series in Computer Science, 1997.

[8] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[9] I. Goodfellow, Y. Bengio, and A. Courville, “Deep Learning,” *MIT Press*, 2016, [Online]. Available: http://www.deeplearningbook.org.

[10] Y. Zhang, J. M. Gorriz, and Z. Dong, “Deep learning in medical image analysis,” *J. Imaging*, vol. 7, no. 4, p. NA, 2021, doi: 10.3390/jimaging7040074.

[11] A. S. Polydoros, L. Nalpantidis, and V. Kruger, “Real-time deep learning of robotic manipulator inverse dynamics,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2015-Decem, pp. 3442–3448, 2015, doi: 10.1109/IROS.2015.7353857.

[12] O. Araque, I. Corcuera-Platas, J. F. Sánchez-Rada, and C. A. Iglesias, “Enhancing deep learning sentiment analysis with ensemble techniques in social applications,” *Expert Syst. Appl.*, vol. 77, pp. 236–246, 2017, doi: 10.1016/j.eswa.2017.02.002.

[13] D. Rav, C. Wong, B. Lo, and G. Yang, “A Deep Learning Approach to on-Node Sensor Data Analytics for Mobile or Wearable Devices,” vol. 12, no. 1, pp. 106–137, 2017.

[14] W. G. Hatcher and W. Yu, “A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends,” *IEEE Access*, vol. 6, pp. 24411–24432, 2018, doi: 10.1109/ACCESS.2018.2830661.

[15] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006, doi: 10.1162/neco.2006.18.7.1527.

[16] Y. LeCun and others, “Generalization and network design strategies,” *Connect. Perspect.*, pp. 143–155, 1989.

[17] D. E. Rumelhart and G. E. Hintont, “Learning Representations by Back-Propagating Errors,” *Cogn. Model.*, no. 2, pp. 3–6, 1986, doi: 10.7551/mitpress/1888.003.0013.

[18] Alex Sherstinsky, “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network.” PhysicaD.

[19] W. Pitts and W. S. McCulloch, “How we know universals the perception of auditory and visual forms,” *Bull. Math. Biophys.*, vol. 9, no. 3, pp. 127–147, 1947, doi: 10.1007/BF02478291.

[20] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, “Object Detection with Deep Learning: A Review,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, 2019, doi: 10.1109/TNNLS.2018.2876865.

[21] G. E. Hilton and R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” vol. 313, no. July, pp. 504–507, 2006.

[22] L. Deng, M. Seltzer, D. Yu, A. Acero, A. Mohamed, and G. Hinton, “Binary coding of speech spectrograms using a deep auto-encoder,” *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH 2010*, no. September, pp. 1692–1695, 2010, doi: 10.21437/interspeech.2010-487.

[23] G. E. Dahl, M. Ranzato, A. R. Mohamed, and G. Hinton, “Phone recognition with the mean-covariance restricted Boltzmann machine,” *Adv. Neural Inf. Process. Syst. 23 24th Annu. Conf. Neural Inf. Process. Syst. 2010, NIPS 2010*, pp. 1–9, 2010.

[24] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” pp. 1–18, 2012, [Online]. Available: http://arxiv.org/abs/1207.0580.

[25] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *Proc. ICML*, pp. 448–456, 2015, doi: 10.1080/17512786.2015.1058180.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Adv. Neural Inf. Process. Syst.*, pp. 1097–1105, 2012, doi: 10.1145/3383972.3383975.

[28] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014, doi: 10.1007/978-3-319-10590-1_53.

[29] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” *2nd Int. Conf. Learn. Represent. ICLR 2014 - Conf. Track Proc.*, 2014.

[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 2818–2826, 2016, doi: 10.1109/CVPR.2016.308.

[31] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, pp. 770–778, 2016, doi: 10.1002/chin.200650130.

[32] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[33] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size,” pp. 1–13, 2016, [Online]. Available: http://arxiv.org/abs/1602.07360.

[34] L. Liu *et al.*, “Deep Learning for Generic Object Detection: A Survey,” *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, 2020, doi: 10.1007/s11263-019-01247-4.

[35] X. He, K. Zhao, and X. Chu, “AutoML: A survey of the state-of-the-art,” *Knowledge-Based Syst.*, vol. 212, p. 106622, 2021, doi: 10.1016/j.knosys.2020.106622.

[36] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” pp. 248–255, 2009, doi: 10.1109/cvprw.2009.5206848.

[37] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context,” in *Computer Vision -- ECCV 2014*, 2014, pp. 740–755.

[38] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 Million Image Database for Scene Recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, 2018, doi: 10.1109/TPAMI.2017.2723009.

[39] A. Kuznetsova *et al.*, “The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale,” *Int. J. Comput. Vis.*, vol. 128, no. 7, pp. 1956–1981, 2020, doi: 10.1007/s11263-020-01316-z.

[40] B. Zhou *et al.*, “Semantic Understanding of Scenes Through the ADE20K Dataset,” *Int. J. Comput. Vis.*, vol. 127, no. 3, pp. 302–321, 2019, doi: 10.1007/s11263-018-1140-0.

[41] A. Patil and M. Rane, “Convolutional Neural Networks: An Overview and Its Applications in Pattern Recognition,” *Smart Innov. Syst. Technol.*, vol. 195, pp. 21–30, 2021, doi: 10.1007/978-981-15-7078-0_3.

[42] D. Lerch, “Deep Learning con redes pre-entrenadas en ImageNet,” *Medium*, 2018.

https://medium.com/neuron4/redes-pre-entrenadas-en-imagenet-30d858c37b1f.

[43] P. Sharma, “Understanding Transfer Learning for Deep Learning,” *Analytics Vidhya*, 2021.

https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/.

[44] Seldon, “Transfer Learning for Machine Learning,” 2021, [Online]. Available:

https://www.seldon.io/transfer-learning#:~:text=Transfer learning is generally used,categorisation or natural language processing.

[45] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, “A Survey of Deep Learning Applications to Autonomous Vehicle Control,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 712–733, 2021, doi: 10.1109/TITS.2019.2962338.

[46] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, “A survey of deep learning techniques for autonomous driving,” *J. F. Robot.*, vol. 37, no. 3, pp. 362–386, 2020, doi: 10.1002/rob.21918.

[47] T. Taketomi, H. Uchiyama, and S. Ikeda, “Visual SLAM algorithms: A survey from 2010 to 2016,” *IPSJ Trans. Comput. Vis. Appl.*, vol. 9, 2017, doi: 10.1186/s41074-017-0027-2.

[48] Y. D. V. Yasuda, L. E. G. Martins, and F. A. M. Cappabianco, “Autonomous visual navigation for mobile robots: A systematic literature review,” *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–34, 2020, doi: 10.1145/3368961.

[49] F. Mota, M. Rocha, J. Rodrigues, V. H. C. Albuquerque, and A. Alexandria, “Localization and Navigation for Autonomous Mobile Robots Using Petri Nets in Indoor Environments,” *IEEE Access*, vol. 6, p. 1, 2018, doi: 10.1109/ACCESS.2018.2846554.

[50] N. G. Hockstein, C. G. Gourin, R. A. Faust, and D. J. Terris, “A history of robots: From science fiction to surgical robotics,” *J. Robot. Surg.*, vol. 1, no. 2, pp. 113–118, 2007, doi: 10.1007/s11701-007-0021-2.

[51] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, “Visual simultaneous localization and mapping: a survey,” *Artif. Intell. Rev.*, vol. 43, no. 1, pp. 55–81, 2015, doi: 10.1007/s10462-012-9365-8.

[52] S. Werner, B. Krieg-Brückner, H. A. Mallot, K. Schweizer, and C. Freksa, “Spatial Cognition: The Role of Landmark, Route, and Survey Knowledge in Human and Robot Navigation1,” pp. 41–50, 1997, doi: 10.1007/978-3-642-60831-5_8.

[53] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. The MIT Press, 2011.

[54] G. Younes, D. Asmar, E. Shammas, and J. Zelek, “Keyframe-based monocular SLAM: design, survey, and future directions,” *Rob. Auton. Syst.*, vol. 98, pp. 67–88, 2017, doi: 10.1016/j.robot.2017.09.010.

[55] P. Kolar, P. Benavidez, and M. Jamshidi, “Survey of datafusion techniques for laser and vision based sensor integration for autonomous navigation,” *Sensors (Switzerland)*, vol. 20, no. 8, pp. 1–49, 2020, doi: 10.3390/s20082180.

[56] MathWorks, “Desarrollo de robots móviles autónomos con MATLAB y Simulink.”

https://es.mathworks.com/campaigns/offers/autonomous-mobile-robots.html.

[57] D. M. Rojas Castro, A. Revel, and M. Menard, “Document image analysis by a mobile robot for autonomous indoor navigation,” *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*,

vol. 2015-Novem, pp. 156–160, 2015, doi: 10.1109/ICDAR.2015.7333743.

[58] X. B. Jin, T. L. Su, J. L. Kong, Y. T. Bai, B. B. Miao, and C. Dou, *State-of-the-art mobile intelligence: Enabling robots to move like humans by estimating mobility with artificial intelligence*, vol. 8, no. 3. 2018.

[59] R. Farkh, M. T. Quasim, K. Al Jaloud, S. Alhuwaimel, and S. T. Siddiqui, "Computer Vision-Control-Based CNN-PID for Mobile Robot," *Comput. Mater. Contin.*, vol. 68, no. 1, pp. 1065–1079, 2021, doi: 10.32604/cmc.2021.016600.

[60] M. M. Almasri, A. M. Alajlan, and K. M. Elleithy, "Trajectory Planning and Collision Avoidance Algorithm for Mobile Robotics System," *IEEE Sens. J.*, vol. 16, no. 12, pp. 5021–5028, 2016, doi: 10.1109/JSEN.2016.2553126.

[61] B. Horan, Z. Najdovski, T. Black, S. Nahavandi, and P. Crothers, "OzTug mobile robot for manufacturing transportation," *Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern.*, pp. 3554–3560, 2011, doi: 10.1109/ICSMC.2011.6084220.

[62] H. Yildiz, N. Korkmaz Can, O. C. Ozguney, and N. Yagiz, "Sliding mode control of a line following robot," *J. Brazilian Soc. Mech. Sci. Eng.*, vol. 42, no. 11, pp. 1–13, 2020, doi: 10.1007/s40430-020-02645-3.

[63] Y. Rubio *et al.*, "Path following fuzzy system for a nonholonomic mobile robot based on frontal camera information," *Stud. Comput. Intell.*, vol. 749, pp. 223–240, 2018, doi: 10.1007/978-3-319-71008-2_18.

[64] J. Chaudhari, A. Desai, and S. Gavarskar, "Line following robot using arduino for hospitals," *2019 2nd Int. Conf. Intell. Commun. Comput. Tech. ICCT 2019*, pp. 330–332, 2019, doi: 10.1109/ICCT46177.2019.8969022.

[65] D. Teso-Fz-Betoño, E. Zulueta, A. Sánchez-Chica, U. Fernandez-Gamiz, and A. Saenz-Aguirre, "Semantic segmentation to develop an indoor navigation system for an autonomous mobile robot," *Mathematics*, vol. 8, no. 5, 2020, doi: 10.3390/MATH8050855.

[66] N. H. Chang, Y. H. Chien, H. H. Chiang, W. Y. Wang, and C. C. Hsu, "A Robot Obstacle Avoidance Method Using Merged CNN Framework," *Proc. - Int. Conf. Mach. Learn. Cybern.*, vol. 2019-July, pp. 1–5, 2019, doi: 10.1109/ICMLC48188.2019.8949168.

[67] V. Hanumante, S. Roy, and S. Maity, "Low Cost Obstacle Avoidance Robot," *Int. J. Soft Comput. Eng.*, vol. 3, no. 4, pp. 52–55, 2013.

[68] J. Borenstein and Y. Koren, "Real-Time Obstacle Avoidance for Fast Mobile Robots," *IEEE Trans. Syst. Man Cybern.*, vol. 19, no. 5, pp. 1179–1187, 1989, doi: 10.1109/21.44033.

[69] A. Shitsukane, W. Cheriuyot, C. Otieno, and M. Mvurya, "A Survey on Obstacles Avoidance Mobile Robot in Static Unknown Environment," *Int. J. Comput.*, no. March, 2018, [Online]. Available: http://ijcjournal.org/.

[70] K. A. Joshi and D. G. Thakore, "A Survey on Moving Object Detection and Tracking in Video Surveillance System," *Int. J. Soft Comput. Eng. ISSN 2231-2307, Vol. Issue-3, July 2012 A*, no. 3, pp. 2231–2307, 2012.

[71] P. Kinsky and Q. ZHou, "Obstacle Avoidance Robot," *A Major Qualif. Proj. Rep. Submitt. to Fac. WORCESTER Polytech. Inst. Partial fulfillment Requir. Degree Bachelor Sci.*, 2011.

[72] R. N. Kandalan and K. Namuduri, “Techniques for Constructing Indoor Navigation Systems for the Visually Impaired: A Review,” *IEEE Trans. Human-Machine Syst.*, vol. 50, no. 6, pp. 492–506, 2020, doi: 10.1109/THMS.2020.3016051.

[73] S. Fazli and L. Kleeman, “Wall following and obstacle avoidance results from a multi-DSP sonar ring on a mobile robot,” *IEEE Int. Conf. Mechatronics Autom. ICMA 2005*, no. July, pp. 432–437, 2005, doi: 10.1109/icma.2005.1626586.

[74] P. G. Luan and N. T. Thinh, “Real-time hybrid navigation system-based path planning and obstacle avoidance for mobile robots,” *Appl. Sci.*, vol. 10, no. 10, 2020, doi: 10.3390/APP10103355.

[75] C. Liu, B. Zheng, C. Wang, Y. Zhao, S. Fu, and H. Li, “CNN-based vision model for obstacle avoidance of mobile robot,” *MATEC Web Conf.*, vol. 139, pp. 4–7, 2017, doi: 10.1051/matecconf/201713900007.

[76] L. Xie, S. Wang, A. Markham, and N. Trigoni, “Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning,” 2017, [Online]. Available: http://arxiv.org/abs/1706.09829.

[77] D. L and Y. D, “Deep learning: methods and applications,” *Found. Trends® Signal Process.*, vol. 7, no. 3–4, pp. 197–387, 2014, doi: 10.1113/expphysiol.1998.sp004170.

[78] B. Jia, W. Feng, and M. Zhu, “Obstacle detection in single images with deep neural networks,” *Signal, Image Video Process.*, vol. 10, no. 6, pp. 1033–1040, 2016, doi: 10.1007/s11760-015-0855-4.

[79] B. S. Lin, C. C. Lee, and P. Y. Chiang, “Simple smartphone-based guiding system for visually impaired people,” *Sensors (Switzerland)*, vol. 17, no. 6, 2017, doi: 10.3390/s17061371.

[80] P. Chakravarty, K. Kelchtermans, T. Roussel, S. Wellens, T. Tuytelaars, and L. Van Eycken, “CNN-based single image obstacle avoidance on a quadrotor,” *Proc. - IEEE Int. Conf. Robot. Autom.*, no. February, pp. 6369–6374, 2017, doi: 10.1109/ICRA.2017.7989752.

[81] L. Tai, S. Li, and M. Liu, “A deep-network solution towards model-less obstacle avoidance,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2016-Novem, pp. 2759–2764, 2016, doi: 10.1109/IROS.2016.7759428.

[82] S. J. Kim and B. K. Kim, “Dynamic ultrasonic hybrid localization system for indoor mobile robots,” *IEEE Trans. Ind. Electron.*, vol. 60, no. 10, pp. 4562–4573, 2013, doi: 10.1109/TIE.2012.2216235.

[83] J. Lim, S. Lee, G. Tewolde, and J. Kwon, “Indoor localization and navigation for a mobile robot equipped with rotating ultrasonic sensors using a smartphone as the robot’s brain,” in *2015 IEEE International Conference on Electro/Information Technology (EIT)*, 2015, pp. 621–625, doi: 10.1109/EIT.2015.7293407.

[84] Q. Sun, J. Yuan, X. Zhang, and F. Sun, “RGB-D SLAM in Indoor Environments with STING-Based Plane Feature Extraction,” *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 3, pp. 1071–1082, 2018, doi: 10.1109/TMECH.2017.2773576.

[85] S. Jingxiang Zheng and D. Y. Bo Cao, “Visual Localization of Inspection Robot using Extended Kalman Filter and Aruco Markers,” *Int. Conf. Robot. Biomimimetrics*, 2018.

[86] A. Motroni, A. Buffi, and P. Nepa, “A Survey on Indoor Vehicle Localization through RFID

Technology," *IEEE Access*, vol. 9, pp. 17921–17942, 2021, doi: 10.1109/ACCESS.2021.3052316.

[87] H. Zhang, C. Zhang, W. Yang, and C. Chen, "Localization and navigation using QR code for mobile robot in indoor environment," 2015, pp. 2501–2506, doi: 10.1109/ROBIO.2015.7419715.

[88] M. Kalaitzakis, B. Cain, S. Carroll, A. Ambrosi, C. Whitehead, and N. Vitzilaios, "Fiducial Markers for Pose Estimation: Overview, Applications and Experimental Comparison of the ARTag, AprilTag, ArUco and STag Markers," *J. Intell. Robot. Syst. Theory Appl.*, vol. 101, no. 4, 2021, doi: 10.1007/s10846-020-01307-9.

[89] A. Abdelgawad, "Localization system for indoor robot using RFID," in *2014 IEEE Symposium on Industrial Electronics Applications (ISIEA)*, 2014, pp. 157–160, doi: 10.1109/ISIEA.2014.8049890.

[90] E. Olson, "AprilTag: A robust and flexible visual fiducial system," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 3400–3407, 2011, doi: 10.1109/ICRA.2011.5979561.

[91] J. Wang and E. Olson, "AprilTag 2: Efficient and Robust Fiducial Detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4193–4198, doi: 10.1109/IROS.2016.7759617.

[92] M. Fiala, "ARTag, a fiducial marker system using digital techniques," *Proc. - 2005 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition, CVPR 2005*, vol. II, pp. 590–596, 2005, doi: 10.1109/CVPR.2005.74.

[93] M. Fiala, "Designing highly reliable fiducial markers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 7, pp. 1317–1324, 2010, doi: 10.1109/TPAMI.2009.146.

[94] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," *Proc. - 2nd IEEE ACM Int. Work. Augment. Reality, IWAR 1999*, pp. 85–94, 1999, doi: 10.1109/IWAR.1999.803809.

[95] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognit.*, vol. 47, no. 6, pp. 2280–2292, 2014, doi: 10.1016/j.patcog.2014.01.005.

[96] G. Yu, Y. Hu, and J. Dai, "TopoTag: A Robust and Scalable Topological Fiducial Marker System," *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 9, pp. 3769–3780, 2021, doi: 10.1109/TVCG.2020.2988466.

[97] J. Degol, T. Bretl, and D. Hoiem, "ChromaTag: A Colored Marker and Fast Detection Algorithm," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, pp. 1481–1490, 2017, doi: 10.1109/ICCV.2017.164.

[98] H. Gao, X. Zhang, J. Wen, J. Yuan, and Y. Fang, "Autonomous Indoor Exploration Via Polygon Map Construction and Graph-Based SLAM Using Directional Endpoint Features," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 4, pp. 1531–1542, 2019, doi: 10.1109/TASE.2018.2883587.

[99] J. Cheng, C. Wang, and M. Q. H. Meng, "Robust Visual Localization in Dynamic Environments Based on Sparse Motion Removal," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 2, pp. 658–669, 2020, doi: 10.1109/TASE.2019.2940543.

[100] C. Cadena *et al.*, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, 2016, doi: 10.1109/TRO.2016.2624754.

[101] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part I,” *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, 2006, doi: 10.1109/MRA.2006.1638022.

[102] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (SLAM): Part II,” *IEEE Robot. Autom. Mag.*, vol. 13, no. 3, pp. 108–117, 2006, doi: 10.1109/MRA.2006.1678144.

[103] D. Dardari, P. Closas, and P. M. Djuric, “Indoor tracking: Theory, methods, and technologies,” *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1263–1278, 2015, doi: 10.1109/TVT.2015.2403868.

[104] Y. Sun, L. Guan, Z. Chang, C. Li, and Y. Gao, “Design of a low-cost indoor navigation system for food delivery robot based on multi-sensor information fusion,” *Sensors (Switzerland)*, vol. 19, no. 22, 2019, doi: 10.3390/s19224980.

[105] F. A. X. Da Mota, M. X. Rocha, J. J. P. C. Rodrigues, V. H. C. De Albuquerque, and A. R. De Alexandria, “Localization and navigation for autonomous mobile robots using petri nets in indoor environments,” *IEEE Access*, vol. 6, pp. 31665–31676, 2018, doi: 10.1109/ACCESS.2018.2846554.

[106] Y. Pan, X. Li, and H. Yu, “Efficient PID Tracking Control of Robotic Manipulators Driven by Compliant Actuators,” *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 2, pp. 915–922, 2019, doi: 10.1109/TCST.2017.2783339.

[107] MathWorks, “Camera Calibration Using AprilTag Markers.”

https://es.mathworks.com/help/vision/ug/camera-calibration-using-apriltag-markers.html.

[108] C. Feng and V. R. Kamat, “Augmented Reality Markers as Spatial Indices for Indoor Mobile AECFM Applications,” *12th Int. Conf. Constr. Appl. Virtual Real. Corresp.*, no. August 2014, pp. 235–242, 2012, doi: 10.13140/2.1.4484.4166.

[109] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, “Speeded up detection of squared fiducial markers,” *Image Vis. Comput.*, vol. 76, pp. 38–47, 2018, doi: https://doi.org/10.1016/j.imavis.2018.05.004.

[110] B. R. K. Mantha and B. Garcia de Soto, “Designing a reliable fiducial marker network for autonomous indoor robot navigation,” *Proc. 36th Int. Symp. Autom. Robot. Constr. ISARC 2019*, no. May, pp. 74–81, 2019, doi: 10.22260/isarc2019/0011.

[111] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, “Visual simultaneous localization and mapping: a survey,” *Artif. Intell. Rev.*, vol. 43, no. 1, pp. 55–81, 2015, doi: 10.1007/s10462-012-9365-8.

[112] AwsomeTech, “Boston Dynamics’ amazing robots Atlas and Handle.”

https://www.youtube.com/watch?v=uhND7Mvp3f4&t=64s&ab_channel=AwesomeTech.

Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

**MÁSTER UNIVERSITARIO EN INGENIERÍA DE CONTROL, AUTOMATIZACIÓN Y ROBÓTICA**

# TRABAJO FIN DE MÁSTER

## *DESARROLLO DE ALGORITMOS DE APRENDIZAJE PROFUNDO BASADOS EN VISIÓN ARTIFICIAL PARA VEHÍCULOS AUTÓNOMOS DE INTERIORES*

## ANEXOS

**Estudiante** *Azurmendi Marquinez, Iker*
**Director/Directora** *Zulueta Guerrero, Ekaitz*
**Departamento** *Ingeniería de Sistemas y Automática*
**Curso académico** *2021-2022*

*Bilbao, 20 de junio de 2022*

# Pliego de condiciones

# Anexo A

Se autoriza al profesor Ekaitz Zulueta Guerrero, a la Universidad del País Vasco/Euskal Herriko Unibertsitatea y al departamento de Sistemas y Automática de ella, para hacer uso y modificaciones, tanto del presente documento, como del código diseñado para llevar a cabo el proyecto.

# Código

# Anexo B

## B.1 Seguimiento de líneas

a) Crear el conjunto de datos para regresión

Este código es el primer paso para generar el algoritmo de seguimiento de líneas: permite crear el conjunto de datos para regresión. Para ello, partiendo de los directorios donde se encuentran las imágenes, así como de las variables que guardan las etiquetas con las coordenadas $(x, y)$ de cada imagen, se realiza el aumento de los datos y se generar los conjuntos de datos para entrenamiento y validación. En este caso se emplean vectores 4d para almacenar las imágenes y un vector 2xn para guardar las etiquetas.

**Juntar imágenes si se encuentran en diferentes carpetas**

```
folder1 = '.\LineaLabo1\';
files1 = dir(fullfile(folder1, '*.jpg'));
folder2 = '.\LineaLabo2\';
files2 = dir(fullfile(folder2, '*.jpg'));
folder3 = '.\LineaLabo3\';
files3 = dir(fullfile(folder3, '*.jpg'));


num_images = length(files1) + length(files2) + length(files3);

j=1;
for i = 1:length(files1)
    img = imread(fullfile(folder1,files1(i).name));
    filename = fullfile(files1(i).name);
    num = extractBetween(filename, 1,'_');
    x = extractBetween(filename, 'xy_','_');
    patron_y =append('_',string(x),'_');
    y = extractBetween(filename, patron y, '.jpg');
    new_filename=append('.\LineaLaboCortaTotal\',string(j),'_xy_',x,'_',y,'.jpg');
    imwrite(img, new_filename);
    j=j+1;
end

for i = 1:length(files2)
    img = imread(fullfile(folder2,files2(i).name));
    filename = fullfile(files2(i).name);
    num = extractBetween(filename, 1,'_');
    x = extractBetween(filename, 'xy_','_');
    patron_y =append('_',string(x),'_');
    y = extractBetween(filename, patron_y, '.jpg');
    new_filename=append('.\LineaLaboCortaTotal\',string(j),'_xy_',x,'_',y,'.jpg');
    imwrite(img, new_filename);
    j=j+1;
end

for i = 1:length(files3)
    img = imread(fullfile(folder3,files3(i).name));
    filename = fullfile(files3(i).name);
    num = extractBetween(filename, 1,'_');
    x = extractBetween(filename, 'xy_','_');
    patron_y =append('_',string(x),'_');
    y = extractBetween(filename, patron_y, '.jpg');
```

```
    new_filename=append('.\LineaLaboCortaTotal\',string(j),'_xy_',x,'_',y,'.jpg');
    imwrite(img, new_filename);
    j=j+1;
end
```

**Voltear imágenes horizontalmente**

```
folder = '.\LineaLaboCortaTotal\';
files = dir(fullfile(folder, '*.jpg'));

for i=1:length(files)
    img = imread(fullfile(folder,files(i).name));
    img2 = flip(img,2);
    filename = fullfile(files(i).name);
    num = extractBetween(filename, 1,'_');
    x = extractBetween(filename, 'xy_','_');
    patron_y =append('_',string(x),'_');
    y = extractBetween(filename, patron_y, '.jpg');
    x=224-str2double(x);

new_filename=append('.\LineaLaboCortaTotal_Flip\',num,'_xy_',string(x),'_',y,'.jpg');
    imwrite(img2, new_filename);
end
```

**Juntar todas las imágenes en una carpeta**

```
folder1 = '.\LineaLaboCortaTotal\';
files1 = dir(fullfile(folder1, '*.jpg'));

folder2 = '.\LineaLaboCortaTotal_Flip\';
files2 = dir(fullfile(folder2, '*.jpg'));

j=1;
for i=1:length(files1)
    img = imread(fullfile(folder1,files1(i).name));
    filename = fullfile(files1(i).name);
    num = extractBetween(filename, 1,'_');
    x = extractBetween(filename, 'xy_','_');
    patron_y =append('_',string(x),'_');
    y = extractBetween(filename, patron_y, '.jpg');

new_filename=append('.\LineaLaboCortaRegresion\',string(j),'_xy_',string(x),'_',y,'.jpg'
);
    imwrite(img, new_filename);
    j=j+1;
end

for i=1:length(files2)
    img = imread(fullfile(folder2,files2(i).name));
    filename = fullfile(files2(i).name);
    num = extractBetween(filename, 1,'_');
    x = extractBetween(filename, 'xy_','_');
    patron_y =append('_',string(x),'_');
    y = extractBetween(filename, patron_y, '.jpg');

new_filename=append('.\LineaLaboCortaRegresion\',string(j),'_xy_',string(x),'_',y,'.jpg'
);
    imwrite(img, new_filename);
    j=j+1;
end
```

**Crear vector de salidas reducido**

```
folder = '.\LineaLaboCortaRegresion\';
files = dir(fullfile(folder, '*.jpg'));
Salidas_cnn_linea_labo = zeros(length(files),3);
array4d_total = zeros(56,56,3,length(files));
```

```
for slice = 1 : length(files)
    filename1 = fullfile(folder,files(slice).name);
    filename2 = fullfile(files(slice).name);
    num = extractBetween(filename2, 1,'_');
    x = extractBetween(filename2, 'xy_','_');
    patron_y =append('_',string(x),'_');
    y = extractBetween(filename2, patron_y, '.jpg');
    Salidas_cnn_linea_labo(slice,1) = str2double(x);
    Salidas cnn linea labo(slice,2) = str2double(y);
    Salidas_cnn_linea_labo(slice,3) = str2double(num);
    thisImage = imread(filename1);
    thisImage = imresize(thisImage, [56, 56]);

thisImage = im2double(thisImage);
    array4d total(:,:,:,str2double(num)) = thisImage;
end

for i=1:length(files)-1
    for j = 1:length(files)-1
        if Salidas_cnn_linea_labo(j,3)>Salidas_cnn_linea_labo(j+1,3)
            aux1 = Salidas_cnn_linea_labo(j,1);
            Salidas_cnn_linea_labo(j,1) = Salidas_cnn_linea_labo(j+1,1);
            Salidas_cnn_linea_labo(j+1,1) = aux1;
            aux2 = Salidas_cnn_linea_labo(j,2);
            Salidas_cnn_linea_labo(j,2) = Salidas_cnn_linea_labo(j+1,2);
            Salidas_cnn_linea_labo(j+1,2) = aux2;
            aux3 = Salidas_cnn_linea_labo(j,3);
            Salidas_cnn_linea_labo(j,3) = Salidas_cnn_linea_labo(j+1,3);
            Salidas_cnn_linea_labo(j+1,3) = aux3;
        end
    end
end
```

**Dividir el conjunto de datos en entrenamiento y validación**

```
[heigth, width] = size(Salidas_cnn_linea_labo);
num_train = floor(heigth*0.9);
num_test = heigth-num_train;

array4d_test = zeros(61,61,3,num_train);
array4d_test = array4d_total(:,:,:,num_train:heigth);

array4d_train = zeros(61,61,3,num_train);
array4d_train = array4d_total(:,:,:,1:num_train);

Salidas_cnn_linea_labo_train = Salidas_cnn_linea_labo(1:num_train,1:2)/4;
Salidas_cnn_linea_labo_test = Salidas_cnn_linea_labo(num_train:heigth,1:2)/4;
```

**Comprobar datos que no son validos**

```
[heigth_train, width_train] = size(Salidas_cnn_linea_labo_train);
[heigth_test, width_test] = size(Salidas_cnn_linea_labo_test);

for i=1:heigth train
    if Salidas_cnn_linea_labo_train(i,1) <= 0
        Salidas_cnn_linea_labo_train(i,1) = Salidas_cnn_linea_labo_train(i,1)+0.75;
        if Salidas_cnn_linea_labo_train(i,1) <= 0
            Salidas_cnn_linea_labo_train(i,1) = 0.05;
        end
    end
    if Salidas_cnn_linea_labo_train(i,2) <= 0
       Salidas_cnn_linea_labo_train(i,2) = Salidas_cnn_linea_labo_train(i,2)+0.75;
       if Salidas_cnn_linea_labo_train(i,2) <= 0
            Salidas_cnn_linea_labo_train(i,2) = 0.05;
       end
    end
end

for i=1:heigth_test
```

```
    if Salidas_cnn_linea_labo_test(i,1) <= 0
        Salidas_cnn_linea_labo_test(i,1) = Salidas_cnn_linea_labo_test(i,1)+0.75;
        if Salidas cnn linea labo test(i,1) <= 0
            Salidas_cnn_linea_labo_test(i,1) = 0.05;
        end
    end
    if Salidas_cnn_linea_labo_test(i,2) <= 0
       Salidas_cnn_linea_labo_test(heigth_test,2) =
Salidas cnn linea labo test(i,2)+0.75;
       if Salidas_cnn_linea_labo_test(i,2) <= 0
            Salidas_cnn_linea_labo_test(i,2) = 0.05;

        end
    end
end
```

b) MobileNet para regresión

Este código genera el grafo de CNN MobileNet v2 de Matlab, pero con la modificación necesaria para realizar la tarea de regresión. Esta modificación consiste en eliminar las tres últimas capas de la CNN original y añadir una capa totalmente conectada del número de variables a predecir y la capa de regresión.

```
params = load(".\params_mobilenet_regresion.mat");
lgraph = layerGraph();
tempLayers = [
    imageInputLayer([56 56
3],"Name","input 1","Normalization","zscore","Mean",params.input 1.Mean,"StandardDeviati
on",params.input_1.StandardDeviation)
    convolution2dLayer([3 3],32,"Name","Conv1","Padding","same","Stride",[2
2],"Bias",params.Conv1.Bias,"Weights",params.Conv1.Weights)

batchNormalizationLayer("Name","bn_Conv1","Epsilon",0.001,"Offset",params.bn_Conv1.Offse
t,"Scale",params.bn_Conv1.Scale,"TrainedMean",params.bn_Conv1.TrainedMean,"TrainedVarian
ce",params.bn_Conv1.TrainedVariance)
    clippedReluLayer(6,"Name","Conv1_relu")
    groupedConvolution2dLayer([3
3],1,32,"Name","expanded_conv_depthwise","Padding","same","Bias",params.expanded_conv_de
pthwise.Bias,"Weights",params.expanded_conv_depthwise.Weights)

batchNormalizationLayer("Name","expanded_conv_depthwise_BN","Epsilon",0.001,"Offset",par
ams.expanded_conv_depthwise_BN.Offset,"Scale",params.expanded_conv_depthwise_BN.Scale,"T
rainedMean",params.expanded_conv_depthwise_BN.TrainedMean,"TrainedVariance",params.expan
ded_conv_depthwise_BN.TrainedVariance)
    clippedReluLayer(6,"Name","expanded_conv_depthwise_relu")
    convolution2dLayer([1
1],16,"Name","expanded_conv_project","Padding","same","Bias",params.expanded_conv_projec
t.Bias,"Weights",params.expanded_conv_project.Weights)

batchNormalizationLayer("Name","expanded_conv_project_BN","Epsilon",0.001,"Offset",param
s.expanded_conv_project_BN.Offset,"Scale",params.expanded_conv_project_BN.Scale,"Trained
Mean",params.expanded_conv_project_BN.TrainedMean,"TrainedVariance",params.expanded_conv
_project_BN.TrainedVariance)
    convolution2dLayer([1
1],96,"Name","block_1_expand","Padding","same","Bias",params.block_1_expand.Bias,"Weight
s",params.block_1_expand.Weights)

batchNormalizationLayer("Name","block_1_expand_BN","Epsilon",0.001,"Offset",params.block
_1_expand_BN.Offset,"Scale",params.block_1_expand_BN.Scale,"TrainedMean",params.block_1_
expand_BN.TrainedMean,"TrainedVariance",params.block_1_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_1_expand_relu")
    groupedConvolution2dLayer([3
3],1,96,"Name","block_1_depthwise","Padding","same","Stride",[2
2],"Bias",params.block_1_depthwise.Bias,"Weights",params.block_1_depthwise.Weights)

batchNormalizationLayer("Name","block_1_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_1_depthwise_BN.Offset,"Scale",params.block_1_depthwise_BN.Scale,"TrainedMean",params
.block_1_depthwise_BN.TrainedMean,"TrainedVariance",params.block_1_depthwise_BN.TrainedV
```

```
ariance)
    clippedReluLayer(6,"Name","block_1_depthwise_relu")
    convolution2dLayer([1
1],24,"Name","block_1_project","Padding","same","Bias",params.block_1_project.Bias,"Weig
hts",params.block_1_project.Weights)

batchNormalizationLayer("Name","block_1_project_BN","Epsilon",0.001,"Offset",params.bloc
k_1_project_BN.Offset,"Scale",params.block_1_project_BN.Scale,"TrainedMean",params.block
 1 project BN.TrainedMean,"TrainedVariance",params.block 1 project BN.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],144,"Name","block_2_expand","Padding","same","Bias",params.block_2_expand.Bias,"Weigh
ts",params.block 2 expand.Weights)

batchNormalizationLayer("Name","block_2_expand_BN","Epsilon",0.001,"Offset",params.block
_2_expand_BN.Offset,"Scale",params.block_2_expand_BN.Scale,"TrainedMean",params.block_2_
expand_BN.TrainedMean,"TrainedVariance",params.block_2_expand_BN.TrainedVariance)

clippedReluLayer(6,"Name","block_2_expand_relu")
    groupedConvolution2dLayer([3
3],1,144,"Name","block_2_depthwise","Padding","same","Bias",params.block_2_depthwise.Bia
s,"Weights",params.block_2_depthwise.Weights)

batchNormalizationLayer("Name","block_2_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_2_depthwise_BN.Offset,"Scale",params.block_2_depthwise_BN.Scale,"TrainedMean",params
.block_2_depthwise_BN.TrainedMean,"TrainedVariance",params.block_2_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_2_depthwise_relu")
    convolution2dLayer([1
1],24,"Name","block_2_project","Padding","same","Bias",params.block_2_project.Bias,"Weig
hts",params.block_2_project.Weights)

batchNormalizationLayer("Name","block_2_project_BN","Epsilon",0.001,"Offset",params.bloc
k_2_project_BN.Offset,"Scale",params.block_2_project_BN.Scale,"TrainedMean",params.block
_2_project_BN.TrainedMean,"TrainedVariance",params.block_2_project_BN.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","block_2_add")
    convolution2dLayer([1
1],144,"Name","block_3_expand","Padding","same","Bias",params.block_3_expand.Bias,"Weigh
ts",params.block_3_expand.Weights)

batchNormalizationLayer("Name","block_3_expand_BN","Epsilon",0.001,"Offset",params.block
_3_expand_BN.Offset,"Scale",params.block_3_expand_BN.Scale,"TrainedMean",params.block_3_
expand_BN.TrainedMean,"TrainedVariance",params.block_3_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_3_expand_relu")
    groupedConvolution2dLayer([3
3],1,144,"Name","block_3_depthwise","Padding","same","Stride",[2
2],"Bias",params.block_3_depthwise.Bias,"Weights",params.block_3_depthwise.Weights)

batchNormalizationLayer("Name","block_3_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_3_depthwise_BN.Offset,"Scale",params.block_3_depthwise_BN.Scale,"TrainedMean",params
.block_3_depthwise_BN.TrainedMean,"TrainedVariance",params.block_3_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_3_depthwise_relu")
    convolution2dLayer([1
1],32,"Name","block_3_project","Padding","same","Bias",params.block_3_project.Bias,"Weig
hts",params.block_3_project.Weights)

batchNormalizationLayer("Name","block_3_project_BN","Epsilon",0.001,"Offset",params.bloc
k_3_project_BN.Offset,"Scale",params.block_3_project_BN.Scale,"TrainedMean",params.block
 3 project BN.TrainedMean,"TrainedVariance",params.block 3 project BN.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],192,"Name","block_4_expand","Padding","same","Bias",params.block_4_expand.Bias,"Weigh
ts",params.block 4 expand.Weights)

batchNormalizationLayer("Name","block_4_expand_BN","Epsilon",0.001,"Offset",params.block
_4_expand_BN.Offset,"Scale",params.block_4_expand_BN.Scale,"TrainedMean",params.block_4_
expand_BN.TrainedMean,"TrainedVariance",params.block_4_expand_BN.TrainedVariance)
```

```
    clippedReluLayer(6,"Name","block_4_expand_relu")
    groupedConvolution2dLayer([3
3],1,192,"Name","block_4_depthwise","Padding","same","Bias",params.block_4_depthwise.Bia
s,"Weights",params.block_4_depthwise.Weights)

batchNormalizationLayer("Name","block_4_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_4_depthwise_BN.Offset,"Scale",params.block_4_depthwise_BN.Scale,"TrainedMean",params
.block_4_depthwise_BN.TrainedMean,"TrainedVariance",params.block_4_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_4_depthwise_relu")
    convolution2dLayer([1
1],32,"Name","block_4_project","Padding","same","Bias",params.block_4_project.Bias,"Weig
hts",params.block_4_project.Weights)

batchNormalizationLayer("Name","block_4_project_BN","Epsilon",0.001,"Offset",params.bloc
k_4_project_BN.Offset,"Scale",params.block_4_project_BN.Scale,"TrainedMean",params.block
_4_project_BN.TrainedMean,"TrainedVariance",params.block_4_project_BN.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = additionLayer(2,"Name","block_4_add");

lgraph = addLayers(lgraph,tempLayers);
tempLayers = [
    convolution2dLayer([1
1],192,"Name","block_5_expand","Padding","same","Bias",params.block_5_expand.Bias,"Weigh
ts",params.block_5_expand.Weights)

batchNormalizationLayer("Name","block_5_expand_BN","Epsilon",0.001,"Offset",params.block
_5_expand_BN.Offset,"Scale",params.block_5_expand_BN.Scale,"TrainedMean",params.block_5_
expand_BN.TrainedMean,"TrainedVariance",params.block_5_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_5_expand_relu")
    groupedConvolution2dLayer([3
3],1,192,"Name","block_5_depthwise","Padding","same","Bias",params.block_5_depthwise.Bia
s,"Weights",params.block_5_depthwise.Weights)

batchNormalizationLayer("Name","block_5_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_5_depthwise_BN.Offset,"Scale",params.block_5_depthwise_BN.Scale,"TrainedMean",params
.block_5_depthwise_BN.TrainedMean,"TrainedVariance",params.block_5_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_5_depthwise_relu")
    convolution2dLayer([1
1],32,"Name","block_5_project","Padding","same","Bias",params.block_5_project.Bias,"Weig
hts",params.block_5_project.Weights)

batchNormalizationLayer("Name","block_5_project_BN","Epsilon",0.001,"Offset",params.bloc
k_5_project_BN.Offset,"Scale",params.block_5_project_BN.Scale,"TrainedMean",params.block
_5_project_BN.TrainedMean,"TrainedVariance",params.block_5_project_BN.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","block_5_add")
    convolution2dLayer([1
1],192,"Name","block_6_expand","Padding","same","Bias",params.block_6_expand.Bias,"Weigh
ts",params.block_6_expand.Weights)

batchNormalizationLayer("Name","block_6_expand_BN","Epsilon",0.001,"Offset",params.block
_6_expand_BN.Offset,"Scale",params.block_6_expand_BN.Scale,"TrainedMean",params.block_6_
expand_BN.TrainedMean,"TrainedVariance",params.block_6_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_6_expand_relu")
    groupedConvolution2dLayer([3
3],1,192,"Name","block_6_depthwise","Padding","same","Stride",[2
2],"Bias",params.block_6_depthwise.Bias,"Weights",params.block_6_depthwise.Weights)

batchNormalizationLayer("Name","block_6_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_6_depthwise_BN.Offset,"Scale",params.block_6_depthwise_BN.Scale,"TrainedMean",params
.block_6_depthwise_BN.TrainedMean,"TrainedVariance",params.block_6_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_6_depthwise_relu")
    convolution2dLayer([1
1],64,"Name","block_6_project","Padding","same","Bias",params.block_6_project.Bias,"Weig
hts",params.block_6_project.Weights)

batchNormalizationLayer("Name","block_6_project_BN","Epsilon",0.001,"Offset",params.bloc
k_6_project_BN.Offset,"Scale",params.block_6_project_BN.Scale,"TrainedMean",params.block
_6_project_BN.TrainedMean,"TrainedVariance",params.block_6_project_BN.TrainedVariance)];
```

```
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],384,"Name","block_7_expand","Padding","same","Bias",params.block_7_expand.Bias,"Weigh
ts",params.block_7_expand.Weights)

batchNormalizationLayer("Name","block_7_expand_BN","Epsilon",0.001,"Offset",params.block
 7 expand BN.Offset,"Scale",params.block 7 expand BN.Scale,"TrainedMean",params.block 7
expand_BN.TrainedMean,"TrainedVariance",params.block_7_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_7_expand_relu")
    groupedConvolution2dLayer([3
3],1,384,"Name","block_7_depthwise","Padding","same","Bias",params.block_7_depthwise.Bia
s,"Weights",params.block_7_depthwise.Weights)

batchNormalizationLayer("Name","block_7_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_7_depthwise_BN.Offset,"Scale",params.block_7_depthwise_BN.Scale,"TrainedMean",params
.block_7_depthwise_BN.TrainedMean,"TrainedVariance",params.block_7_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_7_depthwise_relu")
    convolution2dLayer([1
1],64,"Name","block_7_project","Padding","same","Bias",params.block_7_project.Bias,"Weig
hts",params.block_7_project.Weights)

batchNormalizationLayer("Name","block_7_project_BN","Epsilon",0.001,"Offset",params.bloc
k_7_project_BN.Offset,"Scale",params.block_7_project_BN.Scale,"TrainedMean",params.block
_7_project_BN.TrainedMean,"TrainedVariance",params.block_7_project_BN.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = additionLayer(2,"Name","block_7_add");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],384,"Name","block_8_expand","Padding","same","Bias",params.block_8_expand.Bias,"Weigh
ts",params.block_8_expand.Weights)

batchNormalizationLayer("Name","block_8_expand_BN","Epsilon",0.001,"Offset",params.block
_8_expand_BN.Offset,"Scale",params.block_8_expand_BN.Scale,"TrainedMean",params.block_8_
expand_BN.TrainedMean,"TrainedVariance",params.block_8_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_8_expand_relu")
    groupedConvolution2dLayer([3
3],1,384,"Name","block_8_depthwise","Padding","same","Bias",params.block_8_depthwise.Bia
s,"Weights",params.block_8_depthwise.Weights)

batchNormalizationLayer("Name","block_8_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_8_depthwise_BN.Offset,"Scale",params.block_8_depthwise_BN.Scale,"TrainedMean",params
.block_8_depthwise_BN.TrainedMean,"TrainedVariance",params.block_8_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_8_depthwise_relu")
    convolution2dLayer([1
1],64,"Name","block_8_project","Padding","same","Bias",params.block_8_project.Bias,"Weig
hts",params.block_8_project.Weights)

batchNormalizationLayer("Name","block_8_project_BN","Epsilon",0.001,"Offset",params.bloc
k_8_project_BN.Offset,"Scale",params.block_8_project_BN.Scale,"TrainedMean",params.block
_8_project_BN.TrainedMean,"TrainedVariance",params.block_8_project_BN.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = additionLayer(2,"Name","block_8_add");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],384,"Name","block 9 expand","Padding","same","Bias",params.block 9 expand.Bias,"Weigh
ts",params.block_9_expand.Weights)

batchNormalizationLayer("Name","block_9_expand_BN","Epsilon",0.001,"Offset",params.block
_9_expand_BN.Offset,"Scale",params.block_9_expand_BN.Scale,"TrainedMean",params.block_9_
expand_BN.TrainedMean,"TrainedVariance",params.block_9_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block 9 expand relu")
    groupedConvolution2dLayer([3
3],1,384,"Name","block_9_depthwise","Padding","same","Bias",params.block_9_depthwise.Bia
s,"Weights",params.block_9_depthwise.Weights)
```

```
batchNormalizationLayer("Name","block_9_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_9_depthwise_BN.Offset,"Scale",params.block_9_depthwise_BN.Scale,"TrainedMean",params
.block_9_depthwise_BN.TrainedMean,"TrainedVariance",params.block_9_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_9_depthwise_relu")
    convolution2dLayer([1
1],64,"Name","block_9_project","Padding","same","Bias",params.block_9_project.Bias,"Weig
hts",params.block_9_project.Weights)

batchNormalizationLayer("Name","block_9_project_BN","Epsilon",0.001,"Offset",params.bloc
k_9_project_BN.Offset,"Scale",params.block_9_project_BN.Scale,"TrainedMean",params.block
_9_project_BN.TrainedMean,"TrainedVariance",params.block_9_project_BN.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","block_9_add")
    convolution2dLayer([1
1],384,"Name","block_10_expand","Padding","same","Bias",params.block_10_expand.Bias,"Wei
ghts",params.block_10_expand.Weights)
batchNormalizationLayer("Name","block_10_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_10_expand_BN.Offset,"Scale",params.block_10_expand_BN.Scale,"TrainedMean",params.block
_10_expand_BN.TrainedMean,"TrainedVariance",params.block_10_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_10_expand_relu")
    groupedConvolution2dLayer([3
3],1,384,"Name","block_10_depthwise","Padding","same","Bias",params.block_10_depthwise.B
ias,"Weights",params.block_10_depthwise.Weights)

batchNormalizationLayer("Name","block_10_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_10_depthwise_BN.Offset,"Scale",params.block_10_depthwise_BN.Scale,"TrainedMean",par
ams.block_10_depthwise_BN.TrainedMean,"TrainedVariance",params.block_10_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_10_depthwise_relu")
    convolution2dLayer([1
1],96,"Name","block_10_project","Padding","same","Bias",params.block_10_project.Bias,"We
ights",params.block_10_project.Weights)

batchNormalizationLayer("Name","block_10_project_BN","Epsilon",0.001,"Offset",params.blo
ck_10_project_BN.Offset,"Scale",params.block_10_project_BN.Scale,"TrainedMean",params.bl
ock_10_project_BN.TrainedMean,"TrainedVariance",params.block_10_project_BN.TrainedVarian
ce)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],576,"Name","block_11_expand","Padding","same","Bias",params.block_11_expand.Bias,"Wei
ghts",params.block_11_expand.Weights)

batchNormalizationLayer("Name","block_11_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_11_expand_BN.Offset,"Scale",params.block_11_expand_BN.Scale,"TrainedMean",params.block
_11_expand_BN.TrainedMean,"TrainedVariance",params.block_11_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_11_expand_relu")
    groupedConvolution2dLayer([3
3],1,576,"Name","block_11_depthwise","Padding","same","Bias",params.block_11_depthwise.B
ias,"Weights",params.block_11_depthwise.Weights)

batchNormalizationLayer("Name","block_11_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_11_depthwise_BN.Offset,"Scale",params.block_11_depthwise_BN.Scale,"TrainedMean",par
ams.block_11_depthwise_BN.TrainedMean,"TrainedVariance",params.block_11_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_11_depthwise_relu")
    convolution2dLayer([1
1],96,"Name","block_11_project","Padding","same","Bias",params.block_11_project.Bias,"We
ights",params.block_11_project.Weights)

batchNormalizationLayer("Name","block_11_project_BN","Epsilon",0.001,"Offset",params.blo
ck_11_project_BN.Offset,"Scale",params.block_11_project_BN.Scale,"TrainedMean",params.bl
ock_11_project_BN.TrainedMean,"TrainedVariance",params.block_11_project_BN.TrainedVarian
ce)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = additionLayer(2,"Name","block_11_add");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [convolution2dLayer([1
1],576,"Name","block_12_expand","Padding","same","Bias",params.block_12_expand.Bias,"Wei
```

```
ghts",params.block_12_expand.Weights)

batchNormalizationLayer("Name","block 12 expand BN","Epsilon",0.001,"Offset",params.bloc
k_12_expand_BN.Offset,"Scale",params.block_12_expand_BN.Scale,"TrainedMean",params.block
_12_expand_BN.TrainedMean,"TrainedVariance",params.block_12_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_12_expand_relu")
    groupedConvolution2dLayer([3
3],1,576,"Name","block_12_depthwise","Padding","same","Bias",params.block_12_depthwise.B
ias,"Weights",params.block 12 depthwise.Weights)

batchNormalizationLayer("Name","block_12_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_12_depthwise_BN.Offset,"Scale",params.block_12_depthwise_BN.Scale,"TrainedMean",par
ams.block_12_depthwise_BN.TrainedMean,"TrainedVariance",params.block_12_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block 12 depthwise relu")
    convolution2dLayer([1
1],96,"Name","block_12_project","Padding","same","Bias",params.block_12_project.Bias,"We
ights",params.block_12_project.Weights)

batchNormalizationLayer("Name","block_12_project_BN","Epsilon",0.001,"Offset",params.blo
ck_12_project_BN.Offset,"Scale",params.block_12_project_BN.Scale,"TrainedMean",params.bl
ock_12_project_BN.TrainedMean,"TrainedVariance",params.block_12_project_BN.TrainedVarian
ce)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","block_12_add")
    convolution2dLayer([1
1],576,"Name","block_13_expand","Padding","same","Bias",params.block_13_expand.Bias,"Wei
ghts",params.block_13_expand.Weights)

batchNormalizationLayer("Name","block_13_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_13_expand_BN.Offset,"Scale",params.block_13_expand_BN.Scale,"TrainedMean",params.block
_13_expand_BN.TrainedMean,"TrainedVariance",params.block_13_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_13_expand_relu")
    groupedConvolution2dLayer([3
3],1,576,"Name","block_13_depthwise","Padding","same","Stride",[2
2],"Bias",params.block_13_depthwise.Bias,"Weights",params.block_13_depthwise.Weights)

batchNormalizationLayer("Name","block_13_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_13_depthwise_BN.Offset,"Scale",params.block_13_depthwise_BN.Scale,"TrainedMean",par
ams.block_13_depthwise_BN.TrainedMean,"TrainedVariance",params.block_13_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_13_depthwise_relu")
    convolution2dLayer([1
1],160,"Name","block_13_project","Padding","same","Bias",params.block_13_project.Bias,"W
eights",params.block_13_project.Weights)

batchNormalizationLayer("Name","block_13_project_BN","Epsilon",0.001,"Offset",params.blo
ck_13_project_BN.Offset,"Scale",params.block_13_project_BN.Scale,"TrainedMean",params.bl
ock_13_project_BN.TrainedMean,"TrainedVariance",params.block_13_project_BN.TrainedVarian
ce)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],960,"Name","block_14_expand","Padding","same","Bias",params.block_14_expand.Bias,"Wei
ghts",params.block_14_expand.Weights)

batchNormalizationLayer("Name","block_14_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_14_expand_BN.Offset,"Scale",params.block_14_expand_BN.Scale,"TrainedMean",params.block
_14_expand_BN.TrainedMean,"TrainedVariance",params.block_14_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_14_expand_relu")
    groupedConvolution2dLayer([3
3],1,960,"Name","block 14 depthwise","Padding","same","Bias",params.block 14 depthwise.B
ias,"Weights",params.block_14_depthwise.Weights)

batchNormalizationLayer("Name","block_14_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_14_depthwise_BN.Offset,"Scale",params.block_14_depthwise_BN.Scale,"TrainedMean",par
ams.block_14_depthwise_BN.TrainedMean,"TrainedVariance",params.block_14_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_14_depthwise_relu")
    convolution2dLayer([1
1],160,"Name","block_14_project","Padding","same","Bias",params.block_14_project.Bias,"W
```

```
eights",params.block_14_project.Weights)

batchNormalizationLayer("Name","block 14 project BN","Epsilon",0.001,"Offset",params.blo
ck_14_project_BN.Offset,"Scale",params.block_14_project_BN.Scale,"TrainedMean",params.bl
ock_14_project_BN.TrainedMean,"TrainedVariance",params.block_14_project_BN.TrainedVarian
ce)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = additionLayer(2,"Name","block 14 add");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],960,"Name","block_15_expand","Padding","same","Bias",params.block_15_expand.Bias,"Wei
ghts",params.block 15 expand.Weights)

batchNormalizationLayer("Name","block_15_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_15_expand_BN.Offset,"Scale",params.block_15_expand_BN.Scale,"TrainedMean",params.block
_15_expand_BN.TrainedMean,"TrainedVariance",params.block_15_expand_BN.TrainedVariance)
clippedReluLayer(6,"Name","block_15_expand_relu")

   groupedConvolution2dLayer([3
3],1,960,"Name","block_15_depthwise","Padding","same","Bias",params.block_15_depthwise.B
ias,"Weights",params.block_15_depthwise.Weights)

batchNormalizationLayer("Name","block_15_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_15_depthwise_BN.Offset,"Scale",params.block_15_depthwise_BN.Scale,"TrainedMean",par
ams.block_15_depthwise_BN.TrainedMean,"TrainedVariance",params.block_15_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_15_depthwise_relu")
    convolution2dLayer([1
1],160,"Name","block_15_project","Padding","same","Bias",params.block_15_project.Bias,"W
eights",params.block_15_project.Weights)

batchNormalizationLayer("Name","block_15_project_BN","Epsilon",0.001,"Offset",params.blo
ck_15_project_BN.Offset,"Scale",params.block_15_project_BN.Scale,"TrainedMean",params.bl
ock_15_project_BN.TrainedMean,"TrainedVariance",params.block_15_project_BN.TrainedVarian
ce)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","block_15_add")
    convolution2dLayer([1
1],960,"Name","block_16_expand","Padding","same","Bias",params.block_16_expand.Bias,"Wei
ghts",params.block_16_expand.Weights)

batchNormalizationLayer("Name","block_16_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_16_expand_BN.Offset,"Scale",params.block_16_expand_BN.Scale,"TrainedMean",params.block
_16_expand_BN.TrainedMean,"TrainedVariance",params.block_16_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_16_expand_relu")
    groupedConvolution2dLayer([3
3],1,960,"Name","block_16_depthwise","Padding","same","Bias",params.block_16_depthwise.B
ias,"Weights",params.block_16_depthwise.Weights)

batchNormalizationLayer("Name","block_16_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_16_depthwise_BN.Offset,"Scale",params.block_16_depthwise_BN.Scale,"TrainedMean",par
ams.block_16_depthwise_BN.TrainedMean,"TrainedVariance",params.block_16_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_16_depthwise_relu")
    convolution2dLayer([1
1],320,"Name","block_16_project","Padding","same","Bias",params.block_16_project.Bias,"W
eights",params.block_16_project.Weights)

batchNormalizationLayer("Name","block_16_project_BN","Epsilon",0.001,"Offset",params.blo
ck 16 project BN.Offset,"Scale",params.block 16 project BN.Scale,"TrainedMean",params.bl
ock_16_project_BN.TrainedMean,"TrainedVariance",params.block_16_project_BN.TrainedVarian
ce)
    convolution2dLayer([1
1],1280,"Name","Conv_1","Bias",params.Conv_1.Bias,"Weights",params.Conv_1.Weights)

batchNormalizationLayer("Name","Conv 1 bn","Epsilon",0.001,"Offset",params.Conv 1 bn.Off
set,"Scale",params.Conv_1_bn.Scale,"TrainedMean",params.Conv_1_bn.TrainedMean,"TrainedVa
riance",params.Conv_1_bn.TrainedVariance)
    clippedReluLayer(6,"Name","out_relu")
    globalAveragePooling2dLayer("Name","global_average_pooling2d_1")
```

```
    fullyConnectedLayer(2,"Name","fc")
    regressionLayer("Name","regressionoutput")];
lgraph = addLayers(lgraph,tempLayers);

% clean up helper variable
clear tempLayers;
lgraph = connectLayers(lgraph,"block_1_project_BN","block_2_expand");
lgraph = connectLayers(lgraph,"block_1_project_BN","block_2_add/in2");
lgraph = connectLayers(lgraph,"block_2_project_BN","block_2_add/in1");
lgraph = connectLayers(lgraph,"block_3_project_BN","block_4_expand");
lgraph = connectLayers(lgraph,"block_3_project_BN","block_4_add/in2");
lgraph = connectLayers(lgraph,"block_4_project_BN","block_4_add/in1");
lgraph = connectLayers(lgraph,"block_4_add","block_5_expand");
lgraph = connectLayers(lgraph,"block_4_add","block_5_add/in2");
lgraph = connectLayers(lgraph,"block_5_project_BN","block_5_add/in1");
lgraph = connectLayers(lgraph,"block_6_project_BN","block_7_expand");
lgraph = connectLayers(lgraph,"block_6_project_BN","block_7_add/in2");
lgraph = connectLayers(lgraph,"block_7_project_BN","block_7_add/in1");
lgraph = connectLayers(lgraph,"block_7_add","block_8_expand");
lgraph = connectLayers(lgraph,"block_7_add","block_8_add/in2");
lgraph = connectLayers(lgraph,"block_8_project_BN","block_8_add/in1");
lgraph = connectLayers(lgraph,"block_8_add","block_9_expand");
lgraph = connectLayers(lgraph,"block_8_add","block_9_add/in2");
lgraph = connectLayers(lgraph,"block_9_project_BN","block_9_add/in1");
lgraph = connectLayers(lgraph,"block_10_project_BN","block_11_expand");
lgraph = connectLayers(lgraph,"block_10_project_BN","block_11_add/in2");
lgraph = connectLayers(lgraph,"block_11_project_BN","block_11_add/in1");
lgraph = connectLayers(lgraph,"block_11_add","block_12_expand");
lgraph = connectLayers(lgraph,"block_11_add","block_12_add/in2");
lgraph = connectLayers(lgraph,"block_12_project_BN","block_12_add/in1");
lgraph = connectLayers(lgraph,"block_13_project_BN","block_14_expand");
lgraph = connectLayers(lgraph,"block_13_project_BN","block_14_add/in2");
lgraph = connectLayers(lgraph,"block_14_project_BN","block_14_add/in1");
lgraph = connectLayers(lgraph,"block_14_add","block_15_expand");
lgraph = connectLayers(lgraph,"block_14_add","block_15_add/in2");
lgraph = connectLayers(lgraph,"block_15_project_BN","block_15_add/in1");

plot(lgraph);
```

### c) Entrenar CNN para regresión

El siguiente código sirve para entrenar la CNN anterior para realizar la tarea de regresión. Para ello, en primer lugar, se muestran algunas imágenes del conjunto de datos, y se definen las opciones de entrenamiento. Después, se entrena la CNN con la función *trainNetwork*, para finalmente, realizar su evaluación.

**Mostrar algunas imágenes del conjunto de datos**

```
numTrainImages = numel(Salidas_cnn_linea_labo_train(:,1));
figure
idx = randperm(numTrainImages,24);
for i = 1:numel(idx)
    subplot(4,6,i)
    imshow(array4d_train(:,:,:,idx(i)))
    hold on
    plot(Salidas_cnn_linea_labo_train(idx(i),1),Salidas_cnn_linea_labo_train(idx(i),2),
'r.', 'MarkerSize', 10, 'LineWidth', 2)
end
```

**Opciones de entrenamiento**

```
miniBatchSize  = 128;% 64
validationFrequency = floor(numel(Salidas_cnn_linea_labo_train(:,1))/miniBatchSize);
options = trainingOptions('sgdm', ...
    'MiniBatchSize',miniBatchSize, ...
```

```
    'MaxEpochs',200, ...
    'InitialLearnRate',1e-3, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.1, ...
    'LearnRateDropPeriod',20, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{array4d_test,Salidas_cnn_linea_labo_test}, ...
    'ValidationFrequency',validationFrequency, ...
    'Plots','training-progress', ...
    'ExecutionEnvironment','gpu', ...
    'Verbose',false);
```

**Entrenar CNN para regresión**

```
net = trainNetwork(array4d_train,Salidas_cnn_linea_labo_train,lgraph,options);
```

**Evaluar CNN**

```
YPredicted = predict(net,array4d_test);

predictionError = Salidas_cnn_linea_labo_test - YPredicted;

thr = 5;
numCorrect = sum(abs(predictionError) < thr);
numValidationImages = numel(Salidas_cnn_linea_labo_test(:,1));

accuracy = numCorrect/numValidationImages

squares = predictionError.^2;
rmse = sqrt(mean(squares))

img_prueba = imread('978_xy_190_6.jpg');
img_prueba = imread('imagen_labo_prueba.jpg');
img_prueba = imresize(img_prueba, [61 61]);
img_prueba = im2double(img_prueba);

tic
img_predecida = predict(net,img_prueba);
toc

imshow(img_prueba)
axis on
hold on
plot(img_predecida(1),img_predecida(2), 'r.', 'MarkerSize', 10, 'LineWidth', 2)
plot(190/4,6/4, 'b.', 'MarkerSize', 20, 'LineWidth', 4)
```

d) Automatizar pesos en una red neuronal para clasificación

Como se ha comentado a lo largo del trabajo, para que una CNN pueda realizar dos tareas, hay que modificar los pesos de algunas capas antes de realizar el segundo entrenamiento de la red. El siguiente código realiza este paso automáticamente. Para ello, hace un barrido de todas las capas de la red, y busca las capas que tienen pesos, para modificar sus propiedades y que no aprendan.

```
function layers = automatize_weigths(layers)

    num_Layers = length(layers);

    for i=1:num_Layers

        % CAPAS CONVOLUTION
```

```
        % Propiedad Weigth Learn Rate Factor
        if isprop(layers(i),'WeightLearnRateFactor')
           layers(i).WeightLearnRateFactor = 0;
        end

        % Propiedad Weigth L2 Factor
        if isprop(layers(i),'WeightL2Factor')
           layers(i).WeightL2Factor = 0;
        end

        % Propiedad Bias Learn Rate Factor
        if isprop(layers(i),'BiasLearnRateFactor')
           layers(i).BiasLearnRateFactor = 0;
        end

        % Propiedad Bias L2 Factor
        if isprop(layers(i),'BiasL2Factor')
           layers(i).BiasL2Factor = 0;
        end

        % CAPAS BATCH NORMALIZATION

        % Propiedad Offset Learn Rate Factor
        if isprop(layers(i),'OffsetLearnRateFactor')
           layers(i).OffsetLearnRateFactor = 0;
        end

        % Propiedad Scale L2 Factor
        if isprop(layers(i),'OffsetL2Factor')
           layers(i).OffsetL2Factor = 0;
        end

        % Propiedad Scale Learn Rate Factor
        if isprop(layers(i),'ScaleLearnRateFactor')
           layers(i).ScaleLearnRateFactor = 0;
        end

        % Propiedad Scale L2 Factor
        if isprop(layers(i),'ScaleL2Factor')
           layers(i).ScaleL2Factor = 0;
        end
    end
end
```

e) Crear el conjunto de datos para clasificación

Al igual que se había creado un conjunto para regresión, es necesario generar un conjunto de datos para clasificación que diga si una red es “Línea” o “No Línea”. Para generar este conjunto de datos se emplea el siguiente código.

**Crear el conjunto de datos para la clasificación**

```
folder1 = '.\EntrenamientoCNN\SiLinea\';
folder2 = '.\EntrenamientoCNN\NoLinea\';
filesLinea = dir(fullfile(folder1, '*.jpg'));
filesNoLinea = dir(fullfile(folder2, '*.jpg'));
Salidas_clasificiacion_linea = categorical();

Salidas_clasificiacion_nolinea = categorical();
array4d_clas_linea = zeros(56,56,3,length(filesLinea));
array4d_clas_nolinea = zeros(56,56,3,length(filesNoLinea));
label1='Linea';
label2='NoLinea';
totalFiles = length(filesLinea) + length(filesNoLinea);

for slice = 1 : length(filesLinea)
    filename1 = fullfile(folder1,filesLinea(slice).name);
    thisImage = imread(filename1);
```

```
    thisImage = imresize(thisImage, [56, 56]);
    thisImage = im2double(thisImage);
    array4d_clas_linea(:,:,:,slice) = thisImage;
    Salidas_clasificiacion_linea(slice,1) = label1;
end

for slice = 1 : length(filesNoLinea)
    filename2 = fullfile(folder2,filesNoLinea(slice).name);
    thisImage = imread(filename2);
    thisImage = imresize(thisImage, [56, 56]);
    thisImage = im2double(thisImage);
    array4d_clas_nolinea(:,:,:,slice) = thisImage;
    Salidas_clasificiacion_nolinea(slice,1) = label2;
end
```

**Dividir el conjunto de datos para entrenamiento y validación**

```
trainFilesLinea = floor(length(filesLinea)*0.9);
testFilesLinea = length(filesLinea)-trainFilesLinea;

trainFilesNoLinea = floor(length(filesNoLinea)*0.9);
testFilesNoLinea = length(filesNoLinea)-trainFilesNoLinea;

array4d_clas_train=zeros(56,56,3,trainFilesLinea+trainFilesNoLinea);
array4d_clas_train(:,:,:,1:trainFilesLinea)=array4d_clas_linea(:,:,:,1:trainFilesLinea);
array4d_clas_train(:,:,:,trainFilesLinea+1:trainFilesLinea+trainFilesNoLinea)=array4d_cl
as_nolinea(:,:,:,1:trainFilesNoLinea);

array4d_clas_test=zeros(56,56,3,testFilesLinea+testFilesNoLinea);
array4d_clas_test(:,:,:,1:testFilesLinea)=array4d_clas_linea(:,:,:,trainFilesLinea+1:tra
inFilesLinea+testFilesLinea);
array4d_clas_test(:,:,:,testFilesLinea+1:testFilesLinea+testFilesNoLinea)=array4d_clas_n
olinea(:,:,:,trainFilesNoLinea+1:trainFilesNoLinea+testFilesNoLinea);
```

**Crear los vectores categóricos**

```
Salidas_clasificacion_train=categorical();


for i=1:trainFilesLinea
    Salidas_clasificacion_train(i,1)=label1;
end

for i=trainFilesLinea+1:trainFilesLinea+trainFilesNoLinea
    Salidas_clasificacion_train(i,1)=label2;
end

Salidas_clasificacion_test=categorical();
for i=1:testFilesLinea
    Salidas_clasificacion_test(i,1)=label1;
end

for i=testFilesLinea+1:testFilesLinea+testFilesNoLinea
    Salidas_clasificacion_test(i,1)=label2;
end
```

### f) MobileNet para clasificación

A continuación, se muestra el código correspondiente a la red CNN MobileNet v2 de Matlab, pero modificada para la clasificación de dos clases.

```
params = load(".\params_mobilenet_clasification.mat");
lgraph = layerGraph();
tempLayers = [
```

```
   imageInputLayer([56 56
3],"Name","input_1","Normalization","zscore","Mean",params.input_1.Mean,"StandardDeviati
on",params.input_1.StandardDeviation)
    convolution2dLayer([3 3],32,"Name","Conv1","Padding","same","Stride",[2
2],"Bias",params.Conv1.Bias,"Weights",params.Conv1.Weights)

batchNormalizationLayer("Name","bn_Conv1","Epsilon",0.001,"Offset",params.bn_Conv1.Offse
t,"Scale",params.bn_Conv1.Scale,"TrainedMean",params.bn_Conv1.TrainedMean,"TrainedVarian
ce",params.bn_Conv1.TrainedVariance)
    clippedReluLayer(6,"Name","Conv1_relu")
    groupedConvolution2dLayer([3
3],1,32,"Name","expanded_conv_depthwise","Padding","same","Bias",params.expanded_conv_de
pthwise.Bias,"Weights",params.expanded_conv_depthwise.Weights)

batchNormalizationLayer("Name","expanded_conv_depthwise_BN","Epsilon",0.001,"Offset",par
ams.expanded_conv_depthwise_BN.Offset,"Scale",params.expanded_conv_depthwise_BN.Scale,"T
rainedMean",params.expanded_conv_depthwise_BN.TrainedMean,"TrainedVariance",params.expan
ded_conv_depthwise_BN.TrainedVariance)
    clippedReluLayer(6,"Name","expanded_conv_depthwise_relu")
    convolution2dLayer([1
1],16,"Name","expanded_conv_project","Padding","same","Bias",params.expanded_conv_projec
t.Bias,"Weights",params.expanded_conv_project.Weights)

batchNormalizationLayer("Name","expanded_conv_project_BN","Epsilon",0.001,"Offset",param
s.expanded_conv_project_BN.Offset,"Scale",params.expanded_conv_project_BN.Scale,"Trained
Mean",params.expanded_conv_project_BN.TrainedMean,"TrainedVariance",params.expanded_conv
_project_BN.TrainedVariance)
    convolution2dLayer([1
1],96,"Name","block_1_expand","Padding","same","Bias",params.block_1_expand.Bias,"Weight
s",params.block_1_expand.Weights)

batchNormalizationLayer("Name","block_1_expand_BN","Epsilon",0.001,"Offset",params.block
_1_expand_BN.Offset,"Scale",params.block_1_expand_BN.Scale,"TrainedMean",params.block_1_
expand_BN.TrainedMean,"TrainedVariance",params.block_1_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_1_expand_relu")
    groupedConvolution2dLayer([3
3],1,96,"Name","block_1_depthwise","Padding","same","Stride",[2
2],"Bias",params.block_1_depthwise.Bias,"Weights",params.block_1_depthwise.Weights)

batchNormalizationLayer("Name","block_1_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_1_depthwise_BN.Offset,"Scale",params.block_1_depthwise_BN.Scale,"TrainedMean",params
.block_1_depthwise_BN.TrainedMean,"TrainedVariance",params.block_1_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_1_depthwise_relu")
    convolution2dLayer([1
1],24,"Name","block_1_project","Padding","same","Bias",params.block_1_project.Bias,"Weig
hts",params.block_1_project.Weights)

batchNormalizationLayer("Name","block_1_project_BN","Epsilon",0.001,"Offset",params.bloc
k_1_project_BN.Offset,"Scale",params.block_1_project_BN.Scale,"TrainedMean",params.block
_1_project_BN.TrainedMean,"TrainedVariance",params.block_1_project_BN.TrainedVariance)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],144,"Name","block_2_expand","Padding","same","Bias",params.block_2_expand.Bias,"Weigh
ts",params.block_2_expand.Weights)

batchNormalizationLayer("Name","block_2_expand_BN","Epsilon",0.001,"Offset",params.block
_2_expand_BN.Offset,"Scale",params.block_2_expand_BN.Scale,"TrainedMean",params.block_2_
expand_BN.TrainedMean,"TrainedVariance",params.block_2_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_2_expand_relu")
    groupedConvolution2dLayer([3
3],1,144,"Name","block_2_depthwise","Padding","same","Bias",params.block_2_depthwise.Bia
s,"Weights",params.block_2_depthwise.Weights)

batchNormalizationLayer("Name","block_2_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_2_depthwise_BN.Offset,"Scale",params.block_2_depthwise_BN.Scale,"TrainedMean",params
.block_2_depthwise_BN.TrainedMean,"TrainedVariance",params.block_2_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_2_depthwise_relu")
    convolution2dLayer([1
1],24,"Name","block_2_project","Padding","same","Bias",params.block_2_project.Bias,"Weig
hts",params.block_2_project.Weights)
```

```
batchNormalizationLayer("Name","block_2_project_BN","Epsilon",0.001,"Offset",params.bloc
k_2_project_BN.Offset,"Scale",params.block_2_project_BN.Scale,"TrainedMean",params.block
 2 project BN.TrainedMean,"TrainedVariance",params.block 2 project BN.TrainedVariance)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","block_2_add")
    convolution2dLayer([1
1],144,"Name","block_3_expand","Padding","same","Bias",params.block_3_expand.Bias,"Weigh
ts",params.block_3_expand.Weights)

batchNormalizationLayer("Name","block_3_expand_BN","Epsilon",0.001,"Offset",params.block
_3_expand_BN.Offset,"Scale",params.block_3_expand_BN.Scale,"TrainedMean",params.block_3_
expand BN.TrainedMean,"TrainedVariance",params.block 3 expand BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_3_expand_relu")
    groupedConvolution2dLayer([3
3],1,144,"Name","block_3_depthwise","Padding","same","Stride",[2
2],"Bias",params.block_3_depthwise.Bias,"Weights",params.block_3_depthwise.Weights)

batchNormalizationLayer("Name","block_3_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_3_depthwise_BN.Offset,"Scale",params.block_3_depthwise_BN.Scale,"TrainedMean",params
.block_3_depthwise_BN.TrainedMean,"TrainedVariance",params.block_3_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_3_depthwise_relu")
    convolution2dLayer([1
1],32,"Name","block_3_project","Padding","same","Bias",params.block_3_project.Bias,"Weig
hts",params.block_3_project.Weights)

batchNormalizationLayer("Name","block_3_project_BN","Epsilon",0.001,"Offset",params.bloc
k_3_project_BN.Offset,"Scale",params.block_3_project_BN.Scale,"TrainedMean",params.block
_3_project_BN.TrainedMean,"TrainedVariance",params.block_3_project_BN.TrainedVariance)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],192,"Name","block_4_expand","Padding","same","Bias",params.block_4_expand.Bias,"Weigh
ts",params.block_4_expand.Weights)

batchNormalizationLayer("Name","block_4_expand_BN","Epsilon",0.001,"Offset",params.block
_4_expand_BN.Offset,"Scale",params.block_4_expand_BN.Scale,"TrainedMean",params.block_4_
expand_BN.TrainedMean,"TrainedVariance",params.block_4_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_4_expand_relu")
    groupedConvolution2dLayer([3
3],1,192,"Name","block_4_depthwise","Padding","same","Bias",params.block_4_depthwise.Bia
s,"Weights",params.block_4_depthwise.Weights)

batchNormalizationLayer("Name","block_4_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_4_depthwise_BN.Offset,"Scale",params.block_4_depthwise_BN.Scale,"TrainedMean",params
.block_4_depthwise_BN.TrainedMean,"TrainedVariance",params.block_4_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_4_depthwise_relu")
    convolution2dLayer([1
1],32,"Name","block_4_project","Padding","same","Bias",params.block_4_project.Bias,"Weig
hts",params.block_4_project.Weights)

batchNormalizationLayer("Name","block_4_project_BN","Epsilon",0.001,"Offset",params.bloc
k_4_project_BN.Offset,"Scale",params.block_4_project_BN.Scale,"TrainedMean",params.block
_4_project_BN.TrainedMean,"TrainedVariance",params.block_4_project_BN.TrainedVariance)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = additionLayer(2,"Name","block_4_add");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],192,"Name","block_5_expand","Padding","same","Bias",params.block_5_expand.Bias,"Weigh
ts",params.block_5_expand.Weights)

batchNormalizationLayer("Name","block_5_expand_BN","Epsilon",0.001,"Offset",params.block
_5_expand_BN.Offset,"Scale",params.block_5_expand_BN.Scale,"TrainedMean",params.block_5_
expand_BN.TrainedMean,"TrainedVariance",params.block_5_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_5_expand_relu")
```

```
    groupedConvolution2dLayer([3
3],1,192,"Name","block_5_depthwise","Padding","same","Bias",params.block_5_depthwise.Bia
s,"Weights",params.block_5_depthwise.Weights)

batchNormalizationLayer("Name","block_5_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_5_depthwise_BN.Offset,"Scale",params.block_5_depthwise_BN.Scale,"TrainedMean",params
.block_5_depthwise_BN.TrainedMean,"TrainedVariance",params.block_5_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_5_depthwise_relu")
    convolution2dLayer([1
1],32,"Name","block_5_project","Padding","same","Bias",params.block_5_project.Bias,"Weig
hts",params.block_5_project.Weights)

batchNormalizationLayer("Name","block_5_project_BN","Epsilon",0.001,"Offset",params.bloc
k_5_project_BN.Offset,"Scale",params.block_5_project_BN.Scale,"TrainedMean",params.block
_5_project_BN.TrainedMean,"TrainedVariance",params.block_5_project_BN.TrainedVariance)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","block_5_add")
    convolution2dLayer([1
1],192,"Name","block_6_expand","Padding","same","Bias",params.block_6_expand.Bias,"Weigh
ts",params.block_6_expand.Weights)

batchNormalizationLayer("Name","block_6_expand_BN","Epsilon",0.001,"Offset",params.block
_6_expand_BN.Offset,"Scale",params.block_6_expand_BN.Scale,"TrainedMean",params.block_6_
expand_BN.TrainedMean,"TrainedVariance",params.block_6_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_6_expand_relu")
    groupedConvolution2dLayer([3
3],1,192,"Name","block_6_depthwise","Padding","same","Stride",[2
2],"Bias",params.block_6_depthwise.Bias,"Weights",params.block_6_depthwise.Weights)

batchNormalizationLayer("Name","block_6_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_6_depthwise_BN.Offset,"Scale",params.block_6_depthwise_BN.Scale,"TrainedMean",params
.block_6_depthwise_BN.TrainedMean,"TrainedVariance",params.block_6_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_6_depthwise_relu")
    convolution2dLayer([1
1],64,"Name","block_6_project","Padding","same","Bias",params.block_6_project.Bias,"Weig
hts",params.block_6_project.Weights)

batchNormalizationLayer("Name","block_6_project_BN","Epsilon",0.001,"Offset",params.bloc
k_6_project_BN.Offset,"Scale",params.block_6_project_BN.Scale,"TrainedMean",params.block
_6_project_BN.TrainedMean,"TrainedVariance",params.block_6_project_BN.TrainedVariance)];
tempLayers=automatize_weigths(tempLayers);

lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],384,"Name","block_7_expand","Padding","same","Bias",params.block_7_expand.Bias,"Weigh
ts",params.block_7_expand.Weights)

batchNormalizationLayer("Name","block_7_expand_BN","Epsilon",0.001,"Offset",params.block
_7_expand_BN.Offset,"Scale",params.block_7_expand_BN.Scale,"TrainedMean",params.block_7_
expand_BN.TrainedMean,"TrainedVariance",params.block_7_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_7_expand_relu")
    groupedConvolution2dLayer([3
3],1,384,"Name","block_7_depthwise","Padding","same","Bias",params.block_7_depthwise.Bia
s,"Weights",params.block_7_depthwise.Weights)

batchNormalizationLayer("Name","block_7_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_7_depthwise_BN.Offset,"Scale",params.block_7_depthwise_BN.Scale,"TrainedMean",params
.block_7_depthwise_BN.TrainedMean,"TrainedVariance",params.block_7_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_7_depthwise_relu")
    convolution2dLayer([1
1],64,"Name","block_7_project","Padding","same","Bias",params.block_7_project.Bias,"Weig
hts",params.block_7_project.Weights)

batchNormalizationLayer("Name","block_7_project_BN","Epsilon",0.001,"Offset",params.bloc
k_7_project_BN.Offset,"Scale",params.block_7_project_BN.Scale,"TrainedMean",params.block
_7_project_BN.TrainedMean,"TrainedVariance",params.block_7_project_BN.TrainedVariance)];
tempLayers=automatize_weigths(tempLayers);
```

```
lgraph = addLayers(lgraph,tempLayers);

tempLayers = additionLayer(2,"Name","block 7 add");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],384,"Name","block_8_expand","Padding","same","Bias",params.block_8_expand.Bias,"Weigh
ts",params.block 8 expand.Weights)

batchNormalizationLayer("Name","block_8_expand_BN","Epsilon",0.001,"Offset",params.block
_8_expand_BN.Offset,"Scale",params.block_8_expand_BN.Scale,"TrainedMean",params.block_8_
expand_BN.TrainedMean,"TrainedVariance",params.block_8_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_8_expand_relu")
    groupedConvolution2dLayer([3
3],1,384,"Name","block_8_depthwise","Padding","same","Bias",params.block_8_depthwise.Bia
s,"Weights",params.block_8_depthwise.Weights)

batchNormalizationLayer("Name","block_8_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_8_depthwise_BN.Offset,"Scale",params.block_8_depthwise_BN.Scale,"TrainedMean",params
.block_8_depthwise_BN.TrainedMean,"TrainedVariance",params.block_8_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_8_depthwise_relu")
    convolution2dLayer([1
1],64,"Name","block_8_project","Padding","same","Bias",params.block_8_project.Bias,"Weig
hts",params.block_8_project.Weights)

batchNormalizationLayer("Name","block_8_project_BN","Epsilon",0.001,"Offset",params.bloc
k_8_project_BN.Offset,"Scale",params.block_8_project_BN.Scale,"TrainedMean",params.block
_8_project_BN.TrainedMean,"TrainedVariance",params.block_8_project_BN.TrainedVariance)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = additionLayer(2,"Name","block_8_add");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],384,"Name","block_9_expand","Padding","same","Bias",params.block_9_expand.Bias,"Weigh
ts",params.block_9_expand.Weights)

batchNormalizationLayer("Name","block_9_expand_BN","Epsilon",0.001,"Offset",params.block
_9_expand_BN.Offset,"Scale",params.block_9_expand_BN.Scale,"TrainedMean",params.block_9_
expand_BN.TrainedMean,"TrainedVariance",params.block_9_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_9_expand_relu")
    groupedConvolution2dLayer([3
3],1,384,"Name","block_9_depthwise","Padding","same","Bias",params.block_9_depthwise.Bia
s,"Weights",params.block_9_depthwise.Weights)

batchNormalizationLayer("Name","block_9_depthwise_BN","Epsilon",0.001,"Offset",params.bl
ock_9_depthwise_BN.Offset,"Scale",params.block_9_depthwise_BN.Scale,"TrainedMean",params
.block_9_depthwise_BN.TrainedMean,"TrainedVariance",params.block_9_depthwise_BN.TrainedV
ariance)
    clippedReluLayer(6,"Name","block_9_depthwise_relu")
    convolution2dLayer([1
1],64,"Name","block_9_project","Padding","same","Bias",params.block_9_project.Bias,"Weig
hts",params.block_9_project.Weights)

batchNormalizationLayer("Name","block_9_project_BN","Epsilon",0.001,"Offset",params.bloc
k_9_project_BN.Offset,"Scale",params.block_9_project_BN.Scale,"TrainedMean",params.block
_9_project_BN.TrainedMean,"TrainedVariance",params.block_9_project_BN.TrainedVariance)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","block_9_add")
    convolution2dLayer([1
1],384,"Name","block_10_expand","Padding","same","Bias",params.block_10_expand.Bias,"Wei
ghts",params.block_10_expand.Weights)

batchNormalizationLayer("Name","block 10 expand BN","Epsilon",0.001,"Offset",params.bloc
k_10_expand_BN.Offset,"Scale",params.block_10_expand_BN.Scale,"TrainedMean",params.block
_10_expand_BN.TrainedMean,"TrainedVariance",params.block_10_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_10_expand_relu")
```

```
    groupedConvolution2dLayer([3
3],1,384,"Name","block_10_depthwise","Padding","same","Bias",params.block_10_depthwise.B
ias,"Weights",params.block_10_depthwise.Weights)

batchNormalizationLayer("Name","block_10_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_10_depthwise_BN.Offset,"Scale",params.block_10_depthwise_BN.Scale,"TrainedMean",par
ams.block_10_depthwise_BN.TrainedMean,"TrainedVariance",params.block_10_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_10_depthwise_relu")
    convolution2dLayer([1
1],96,"Name","block_10_project","Padding","same","Bias",params.block_10_project.Bias,"We
ights",params.block_10_project.Weights)

batchNormalizationLayer("Name","block_10_project_BN","Epsilon",0.001,"Offset",params.blo
ck_10_project_BN.Offset,"Scale",params.block_10_project_BN.Scale,"TrainedMean",params.bl
ock_10_project_BN.TrainedMean,"TrainedVariance",params.block_10_project_BN.TrainedVarian
ce)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],576,"Name","block_11_expand","Padding","same","Bias",params.block_11_expand.Bias,"Wei
ghts",params.block_11_expand.Weights)

batchNormalizationLayer("Name","block_11_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_11_expand_BN.Offset,"Scale",params.block_11_expand_BN.Scale,"TrainedMean",params.block
_11_expand_BN.TrainedMean,"TrainedVariance",params.block_11_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_11_expand_relu")
    groupedConvolution2dLayer([3
3],1,576,"Name","block_11_depthwise","Padding","same","Bias",params.block_11_depthwise.B
ias,"Weights",params.block_11_depthwise.Weights)

batchNormalizationLayer("Name","block_11_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_11_depthwise_BN.Offset,"Scale",params.block_11_depthwise_BN.Scale,"TrainedMean",par
ams.block_11_depthwise_BN.TrainedMean,"TrainedVariance",params.block_11_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_11_depthwise_relu")
    convolution2dLayer([1
1],96,"Name","block_11_project","Padding","same","Bias",params.block_11_project.Bias,"We
ights",params.block_11_project.Weights)

batchNormalizationLayer("Name","block_11_project_BN","Epsilon",0.001,"Offset",params.blo
ck_11_project_BN.Offset,"Scale",params.block_11_project_BN.Scale,"TrainedMean",params.bl
ock_11_project_BN.TrainedMean,"TrainedVariance",params.block_11_project_BN.TrainedVarian
ce)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = additionLayer(2,"Name","block_11_add");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],576,"Name","block_12_expand","Padding","same","Bias",params.block_12_expand.Bias,"Wei
ghts",params.block_12_expand.Weights)

batchNormalizationLayer("Name","block_12_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_12_expand_BN.Offset,"Scale",params.block_12_expand_BN.Scale,"TrainedMean",params.block
_12_expand_BN.TrainedMean,"TrainedVariance",params.block_12_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_12_expand_relu")
    groupedConvolution2dLayer([3
3],1,576,"Name","block_12_depthwise","Padding","same","Bias",params.block_12_depthwise.B
ias,"Weights",params.block_12_depthwise.Weights)

batchNormalizationLayer("Name","block_12_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_12_depthwise_BN.Offset,"Scale",params.block_12_depthwise_BN.Scale,"TrainedMean",par
ams.block_12_depthwise_BN.TrainedMean,"TrainedVariance",params.block_12_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_12_depthwise_relu")
    convolution2dLayer([1
1],96,"Name","block_12_project","Padding","same","Bias",params.block_12_project.Bias,"We
ights",params.block_12_project.Weights)

batchNormalizationLayer("Name","block_12_project_BN","Epsilon",0.001,"Offset",params.blo
```

```
ck_12_project_BN.Offset,"Scale",params.block_12_project_BN.Scale,"TrainedMean",params.bl
ock_12_project_BN.TrainedMean,"TrainedVariance",params.block_12_project_BN.TrainedVarian
ce)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","block_12_add")
    convolution2dLayer([1
1],576,"Name","block_13_expand","Padding","same","Bias",params.block_13_expand.Bias,"Wei
ghts",params.block_13_expand.Weights)

batchNormalizationLayer("Name","block_13_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_13_expand_BN.Offset,"Scale",params.block_13_expand_BN.Scale,"TrainedMean",params.block
 13 expand BN.TrainedMean,"TrainedVariance",params.block 13 expand BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_13_expand_relu")
    groupedConvolution2dLayer([3
3],1,576,"Name","block_13_depthwise","Padding","same","Stride",[2
2],"Bias",params.block_13_depthwise.Bias,"Weights",params.block_13_depthwise.Weights)

batchNormalizationLayer("Name","block_13_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_13_depthwise_BN.Offset,"Scale",params.block_13_depthwise_BN.Scale,"TrainedMean",par
ams.block_13_depthwise_BN.TrainedMean,"TrainedVariance",params.block_13_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_13_depthwise_relu")
    convolution2dLayer([1
1],160,"Name","block_13_project","Padding","same","Bias",params.block_13_project.Bias,"W
eights",params.block_13_project.Weights)

batchNormalizationLayer("Name","block_13_project_BN","Epsilon",0.001,"Offset",params.blo
ck_13_project_BN.Offset,"Scale",params.block_13_project_BN.Scale,"TrainedMean",params.bl
ock_13_project_BN.TrainedMean,"TrainedVariance",params.block_13_project_BN.TrainedVarian
ce)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],960,"Name","block_14_expand","Padding","same","Bias",params.block_14_expand.Bias,"Wei
ghts",params.block_14_expand.Weights)

batchNormalizationLayer("Name","block_14_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_14_expand_BN.Offset,"Scale",params.block_14_expand_BN.Scale,"TrainedMean",params.block
_14_expand_BN.TrainedMean,"TrainedVariance",params.block_14_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_14_expand_relu")
    groupedConvolution2dLayer([3
3],1,960,"Name","block_14_depthwise","Padding","same","Bias",params.block_14_depthwise.B
ias,"Weights",params.block_14_depthwise.Weights)

batchNormalizationLayer("Name","block_14_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_14_depthwise_BN.Offset,"Scale",params.block_14_depthwise_BN.Scale,"TrainedMean",par
ams.block_14_depthwise_BN.TrainedMean,"TrainedVariance",params.block_14_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_14_depthwise_relu")
    convolution2dLayer([1
1],160,"Name","block_14_project","Padding","same","Bias",params.block_14_project.Bias,"W
eights",params.block_14_project.Weights)

batchNormalizationLayer("Name","block_14_project_BN","Epsilon",0.001,"Offset",params.blo
ck_14_project_BN.Offset,"Scale",params.block_14_project_BN.Scale,"TrainedMean",params.bl
ock_14_project_BN.TrainedMean,"TrainedVariance",params.block_14_project_BN.TrainedVarian
ce)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = additionLayer(2,"Name","block_14_add");
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],960,"Name","block_15_expand","Padding","same","Bias",params.block_15_expand.Bias,"Wei
ghts",params.block_15_expand.Weights)

batchNormalizationLayer("Name","block_15_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_15_expand_BN.Offset,"Scale",params.block_15_expand_BN.Scale,"TrainedMean",params.block
```

```
_15_expand_BN.TrainedMean,"TrainedVariance",params.block_15_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_15_expand_relu")
    groupedConvolution2dLayer([3
3],1,960,"Name","block_15_depthwise","Padding","same","Bias",params.block_15_depthwise.B
ias,"Weights",params.block_15_depthwise.Weights)

batchNormalizationLayer("Name","block_15_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_15_depthwise_BN.Offset,"Scale",params.block_15_depthwise_BN.Scale,"TrainedMean",par
ams.block 15 depthwise BN.TrainedMean,"TrainedVariance",params.block 15 depthwise BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_15_depthwise_relu")
    convolution2dLayer([1
1],160,"Name","block_15_project","Padding","same","Bias",params.block_15_project.Bias,"W
eights",params.block_15_project.Weights)

batchNormalizationLayer("Name","block_15_project_BN","Epsilon",0.001,"Offset",params.blo
ck_15_project_BN.Offset,"Scale",params.block_15_project_BN.Scale,"TrainedMean",params.bl
ock_15_project_BN.TrainedMean,"TrainedVariance",params.block_15_project_BN.TrainedVarian
ce)];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","block_15_add")
    convolution2dLayer([1
1],960,"Name","block_16_expand","Padding","same","Bias",params.block_16_expand.Bias,"Wei
ghts",params.block_16_expand.Weights)

batchNormalizationLayer("Name","block_16_expand_BN","Epsilon",0.001,"Offset",params.bloc
k_16_expand_BN.Offset,"Scale",params.block_16_expand_BN.Scale,"TrainedMean",params.block
_16_expand_BN.TrainedMean,"TrainedVariance",params.block_16_expand_BN.TrainedVariance)
    clippedReluLayer(6,"Name","block_16_expand_relu")
    groupedConvolution2dLayer([3
3],1,960,"Name","block_16_depthwise","Padding","same","Bias",params.block_16_depthwise.B
ias,"Weights",params.block_16_depthwise.Weights)

batchNormalizationLayer("Name","block_16_depthwise_BN","Epsilon",0.001,"Offset",params.b
lock_16_depthwise_BN.Offset,"Scale",params.block_16_depthwise_BN.Scale,"TrainedMean",par
ams.block_16_depthwise_BN.TrainedMean,"TrainedVariance",params.block_16_depthwise_BN.Tra
inedVariance)
    clippedReluLayer(6,"Name","block_16_depthwise_relu")
    convolution2dLayer([1
1],320,"Name","block_16_project","Padding","same","Bias",params.block_16_project.Bias,"W
eights",params.block_16_project.Weights)

batchNormalizationLayer("Name","block_16_project_BN","Epsilon",0.001,"Offset",params.blo
ck_16_project_BN.Offset,"Scale",params.block_16_project_BN.Scale,"TrainedMean",params.bl
ock_16_project_BN.TrainedMean,"TrainedVariance",params.block_16_project_BN.TrainedVarian
ce)
    convolution2dLayer([1
1],1280,"Name","Conv_1","Bias",params.Conv_1.Bias,"Weights",params.Conv_1.Weights)

batchNormalizationLayer("Name","Conv_1_bn","Epsilon",0.001,"Offset",params.Conv_1_bn.Off
set,"Scale",params.Conv_1_bn.Scale,"TrainedMean",params.Conv_1_bn.TrainedMean,"TrainedVa
riance",params.Conv_1_bn.TrainedVariance)
    clippedReluLayer(6,"Name","out_relu")
    globalAveragePooling2dLayer("Name","global_average_pooling2d_1")];
tempLayers=automatize_weigths(tempLayers);
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    fullyConnectedLayer(2,"Name","fc")
    softmaxLayer("Name","softmax")
    classificationLayer("Name","classoutput")];
lgraph = addLayers(lgraph,tempLayers);

% clean up helper variable
clear tempLayers;

lgraph = connectLayers(lgraph,"block_1_project_BN","block_2_expand");
lgraph = connectLayers(lgraph,"block_1_project_BN","block_2_add/in2");
lgraph = connectLayers(lgraph,"block_2_project_BN","block_2_add/in1");
lgraph = connectLayers(lgraph,"block_3_project_BN","block_4_expand");
lgraph = connectLayers(lgraph,"block_3_project_BN","block_4_add/in2");
lgraph = connectLayers(lgraph,"block_4_project_BN","block_4_add/in1");
```

```
lgraph = connectLayers(lgraph,"block_4_add","block_5_expand");
lgraph = connectLayers(lgraph,"block_4_add","block_5_add/in2");
lgraph = connectLayers(lgraph,"block_5_project_BN","block_5_add/in1");
lgraph = connectLayers(lgraph,"block_6_project_BN","block_7_expand");
lgraph = connectLayers(lgraph,"block_6_project_BN","block_7_add/in2");
lgraph = connectLayers(lgraph,"block_7_project_BN","block_7_add/in1");
lgraph = connectLayers(lgraph,"block_7_add","block_8_expand");
lgraph = connectLayers(lgraph,"block_7_add","block_8_add/in2");
lgraph = connectLayers(lgraph,"block_8_project_BN","block_8_add/in1");
lgraph = connectLayers(lgraph,"block_8_add","block_9_expand");
lgraph = connectLayers(lgraph,"block_8_add","block_9_add/in2");
lgraph = connectLayers(lgraph,"block_9_project_BN","block_9_add/in1");
lgraph = connectLayers(lgraph,"block_10_project_BN","block_11_expand");
lgraph = connectLayers(lgraph,"block_10_project_BN","block_11_add/in2");
lgraph = connectLayers(lgraph,"block_11_project_BN","block_11_add/in1");
lgraph = connectLayers(lgraph,"block_11_add","block_12_expand");
lgraph = connectLayers(lgraph,"block_11_add","block_12_add/in2");
lgraph = connectLayers(lgraph,"block_12_project_BN","block_12_add/in1");
lgraph = connectLayers(lgraph,"block_13_project_BN","block_14_expand");
lgraph = connectLayers(lgraph,"block_13_project_BN","block_14_add/in2");
lgraph = connectLayers(lgraph,"block_14_project_BN","block_14_add/in1");
lgraph = connectLayers(lgraph,"block_14_add","block_15_expand");
lgraph = connectLayers(lgraph,"block_14_add","block_15_add/in2");
lgraph = connectLayers(lgraph,"block_15_project_BN","block_15_add/in1");
lgraph = connectLayers(lgraph,"global_average_pooling2d_1","fc");

plot(lgraph);
```

### g) Entrenar CNN para clasificación

Finalmente, se muestra el código empleado para entrenar la red para clasificación. Tras este entrenamiento, simplemente faltaría unir ambas redes, por ejemplo, en la aplicación *Deep Network Designer* de Matlab.

**Mostrar conjunto de imágenes**

```
numTrainImages = length(Salidas_clasificacion_train);
figure
idx = randperm(numTrainImages,24);
for i = 1:numel(idx)
    subplot(4,6,i)
    imshow(array4d_clas_train(:,:,:,idx(i)))
    if Salidas_clasificacion_train(idx(i))=='Linea'
        title('Linea')
    else
        title('No Linea')
    end
end
```

**Definir opciones de entrenamiento**

```
miniBatchSize  = 128;% 64
validationFrequency = floor(length(Salidas clasificacion train)/miniBatchSize);
options = trainingOptions('sgdm', ...
    'MiniBatchSize',miniBatchSize, ...
    'MaxEpochs',200, ...
    'InitialLearnRate',1e-3, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.1, ...
    'LearnRateDropPeriod',20, ...
    'Shuffle','every-epoch', ...
    'ValidationData',{array4d_clas_test,Salidas_clasificacion_test}, ...
    'ValidationFrequency',validationFrequency, ...
    'Plots','training-progress', ...
    'ExecutionEnvironment','gpu', ...
    'Verbose',false);
```

**Entrenamiento de la CNN**

```
net = trainNetwork(array4d_clas_train,Salidas_clasificacion_train,lgraph,options);
```

**Convertir CNN a dlnetwork**

```
dl_ikernet = dlnetwork(ikernet_graph_mobilenet);
```

**Prueba de la red**

```
im = imread('2165_xy_124_5.jpg');
im = array4d_clas_test(:,:,:,2);
im = imresize(im,[61 61]);
im =im2double(im);
image(im)
X = dlarray(im,"SSCB");
tic
[pos, flag] = predict(dl_ikernet,X);
toc
prediccion_xy=extractdata(pos);
flag=extractdata(flag);

if flag(1)>0.5
    title('Linea')
    hold on
    plot(prediccion_xy(1),prediccion_xy(2), 'r.', 'MarkerSize', 20, 'LineWidth', 4)
else
    title('No Línea')
end
```

**Determinar precisión de la red**

```
YPredicted = classify(net,array4d_clas_test);

num_test = length(YPredicted);
aciertos = 0;

for i=1:num_test
    if YPredicted(i) == Salidas_clasificacion_test(i)
        aciertos = aciertos + 1;
    end
end

precision = (aciertos/num_test)*100;
```

## B.2 Detección de obstáculos

a) Crear red neuronal

Al igual que en el apartado anterior, para entrenar una CNN con Matlab, es necesario crear un grafo con la arquitectura de la red que se va a entrenar. En este caso se muestra el código de una ResNet18 modificada para detectar dos clases: “Con Obstáculo” y “Sin Obstáculo”.

```
params = load(".\params_resnet18_obstaculos.mat");
lgraph = layerGraph();
tempLayers = [
```

```
   imageInputLayer([224 224
3],"Name","data","Normalization","zscore","Mean",params.data.Mean,"StandardDeviation",pa
rams.data.StandardDeviation)
    convolution2dLayer([7 7],64,"Name","conv1","BiasLearnRateFactor",0,"Padding",[3 3 3
3],"Stride",[2 2],"Bias",params.conv1.Bias,"Weights",params.conv1.Weights)

batchNormalizationLayer("Name","bn_conv1","Offset",params.bn_conv1.Offset,"Scale",params
.bn_conv1.Scale,"TrainedMean",params.bn_conv1.TrainedMean,"TrainedVariance",params.bn_co
nv1.TrainedVariance)
    reluLayer("Name","conv1_relu")
    maxPooling2dLayer([3 3],"Name","pool1","Padding",[1 1 1 1],"Stride",[2 2])];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3
3],64,"Name","res2a_branch2a","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res2a_branch2a.Bias,"Weights",params.res2a_branch2a.Weights)

batchNormalizationLayer("Name","bn2a_branch2a","Offset",params.bn2a_branch2a.Offset,"Sca
le",params.bn2a_branch2a.Scale,"TrainedMean",params.bn2a_branch2a.TrainedMean,"TrainedVa
riance",params.bn2a_branch2a.TrainedVariance)
    reluLayer("Name","res2a_branch2a_relu")
    convolution2dLayer([3
3],64,"Name","res2a_branch2b","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res2a_branch2b.Bias,"Weights",params.res2a_branch2b.Weights)

batchNormalizationLayer("Name","bn2a_branch2b","Offset",params.bn2a_branch2b.Offset,"Sca
le",params.bn2a_branch2b.Scale,"TrainedMean",params.bn2a_branch2b.TrainedMean,"TrainedVa
riance",params.bn2a_branch2b.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res2a")
    reluLayer("Name","res2a_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3
3],64,"Name","res2b_branch2a","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res2b_branch2a.Bias,"Weights",params.res2b_branch2a.Weights)

batchNormalizationLayer("Name","bn2b_branch2a","Offset",params.bn2b_branch2a.Offset,"Sca
le",params.bn2b_branch2a.Scale,"TrainedMean",params.bn2b_branch2a.TrainedMean,"TrainedVa
riance",params.bn2b_branch2a.TrainedVariance)
    reluLayer("Name","res2b_branch2a_relu")
    convolution2dLayer([3
3],64,"Name","res2b_branch2b","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res2b_branch2b.Bias,"Weights",params.res2b_branch2b.Weights)

batchNormalizationLayer("Name","bn2b_branch2b","Offset",params.bn2b_branch2b.Offset,"Sca
le",params.bn2b_branch2b.Scale,"TrainedMean",params.bn2b_branch2b.TrainedMean,"TrainedVa
riance",params.bn2b_branch2b.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res2b")
    reluLayer("Name","res2b_relu")];

lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],128,"Name","res3a_branch1","BiasLearnRateFactor",0,"Stride",[2
2],"Bias",params.res3a_branch1.Bias,"Weights",params.res3a_branch1.Weights)

batchNormalizationLayer("Name","bn3a_branch1","Offset",params.bn3a_branch1.Offset,"Scale
",params.bn3a_branch1.Scale,"TrainedMean",params.bn3a_branch1.TrainedMean,"TrainedVarian
ce",params.bn3a_branch1.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3
3],128,"Name","res3a_branch2a","BiasLearnRateFactor",0,"Padding",[1 1 1 1],"Stride",[2
2],"Bias",params.res3a_branch2a.Bias,"Weights",params.res3a_branch2a.Weights)
```

```
batchNormalizationLayer("Name","bn3a_branch2a","Offset",params.bn3a_branch2a.Offset,"Sca
le",params.bn3a_branch2a.Scale,"TrainedMean",params.bn3a_branch2a.TrainedMean,"TrainedVa
riance",params.bn3a_branch2a.TrainedVariance)
    reluLayer("Name","res3a_branch2a_relu")
    convolution2dLayer([3
3],128,"Name","res3a_branch2b","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res3a_branch2b.Bias,"Weights",params.res3a_branch2b.Weights)

batchNormalizationLayer("Name","bn3a_branch2b","Offset",params.bn3a_branch2b.Offset,"Sca
le",params.bn3a_branch2b.Scale,"TrainedMean",params.bn3a_branch2b.TrainedMean,"TrainedVa
riance",params.bn3a_branch2b.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res3a")
    reluLayer("Name","res3a_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3
3],128,"Name","res3b_branch2a","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res3b_branch2a.Bias,"Weights",params.res3b_branch2a.Weights)

batchNormalizationLayer("Name","bn3b_branch2a","Offset",params.bn3b_branch2a.Offset,"Sca
le",params.bn3b_branch2a.Scale,"TrainedMean",params.bn3b_branch2a.TrainedMean,"TrainedVa
riance",params.bn3b_branch2a.TrainedVariance)
    reluLayer("Name","res3b_branch2a_relu")
    convolution2dLayer([3
3],128,"Name","res3b_branch2b","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res3b_branch2b.Bias,"Weights",params.res3b_branch2b.Weights)

batchNormalizationLayer("Name","bn3b_branch2b","Offset",params.bn3b_branch2b.Offset,"Sca
le",params.bn3b_branch2b.Scale,"TrainedMean",params.bn3b_branch2b.TrainedMean,"TrainedVa
riance",params.bn3b_branch2b.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res3b")
    reluLayer("Name","res3b_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],256,"Name","res4a_branch1","BiasLearnRateFactor",0,"Stride",[2
2],"Bias",params.res4a_branch1.Bias,"Weights",params.res4a_branch1.Weights)

batchNormalizationLayer("Name","bn4a_branch1","Offset",params.bn4a_branch1.Offset,"Scale
",params.bn4a_branch1.Scale,"TrainedMean",params.bn4a_branch1.TrainedMean,"TrainedVarian
ce",params.bn4a_branch1.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3
3],256,"Name","res4a_branch2a","BiasLearnRateFactor",0,"Padding",[1 1 1 1],"Stride",[2
2],"Bias",params.res4a_branch2a.Bias,"Weights",params.res4a_branch2a.Weights)

batchNormalizationLayer("Name","bn4a_branch2a","Offset",params.bn4a_branch2a.Offset,"Sca
le",params.bn4a_branch2a.Scale,"TrainedMean",params.bn4a_branch2a.TrainedMean,"TrainedVa
riance",params.bn4a_branch2a.TrainedVariance)
    reluLayer("Name","res4a_branch2a_relu")
    convolution2dLayer([3
3],256,"Name","res4a_branch2b","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res4a_branch2b.Bias,"Weights",params.res4a_branch2b.Weights)

batchNormalizationLayer("Name","bn4a_branch2b","Offset",params.bn4a_branch2b.Offset,"Sca
le",params.bn4a_branch2b.Scale,"TrainedMean",params.bn4a_branch2b.TrainedMean,"TrainedVa
riance",params.bn4a_branch2b.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res4a")
    reluLayer("Name","res4a_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
```

```
    convolution2dLayer([3
3],256,"Name","res4b_branch2a","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res4b_branch2a.Bias,"Weights",params.res4b_branch2a.Weights)

batchNormalizationLayer("Name","bn4b_branch2a","Offset",params.bn4b_branch2a.Offset,"Sca
le",params.bn4b_branch2a.Scale,"TrainedMean",params.bn4b_branch2a.TrainedMean,"TrainedVa
riance",params.bn4b_branch2a.TrainedVariance)
    reluLayer("Name","res4b_branch2a_relu")
    convolution2dLayer([3
3],256,"Name","res4b_branch2b","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res4b_branch2b.Bias,"Weights",params.res4b_branch2b.Weights)

batchNormalizationLayer("Name","bn4b_branch2b","Offset",params.bn4b_branch2b.Offset,"Sca
le",params.bn4b_branch2b.Scale,"TrainedMean",params.bn4b_branch2b.TrainedMean,"TrainedVa
riance",params.bn4b_branch2b.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res4b")
    reluLayer("Name","res4b_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3
3],512,"Name","res5a_branch2a","BiasLearnRateFactor",0,"Padding",[1 1 1 1],"Stride",[2
2],"Bias",params.res5a_branch2a.Bias,"Weights",params.res5a_branch2a.Weights)

batchNormalizationLayer("Name","bn5a_branch2a","Offset",params.bn5a_branch2a.Offset,"Sca
le",params.bn5a_branch2a.Scale,"TrainedMean",params.bn5a_branch2a.TrainedMean,"TrainedVa
riance",params.bn5a_branch2a.TrainedVariance)
    reluLayer("Name","res5a_branch2a_relu")
    convolution2dLayer([3
3],512,"Name","res5a_branch2b","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res5a_branch2b.Bias,"Weights",params.res5a_branch2b.Weights)

batchNormalizationLayer("Name","bn5a_branch2b","Offset",params.bn5a_branch2b.Offset,"Sca
le",params.bn5a_branch2b.Scale,"TrainedMean",params.bn5a_branch2b.TrainedMean,"TrainedVa
riance",params.bn5a_branch2b.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([1
1],512,"Name","res5a_branch1","BiasLearnRateFactor",0,"Stride",[2
2],"Bias",params.res5a_branch1.Bias,"Weights",params.res5a_branch1.Weights)

batchNormalizationLayer("Name","bn5a_branch1","Offset",params.bn5a_branch1.Offset,"Scale
",params.bn5a_branch1.Scale,"TrainedMean",params.bn5a_branch1.TrainedMean,"TrainedVarian
ce",params.bn5a_branch1.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res5a")
    reluLayer("Name","res5a_relu")];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    convolution2dLayer([3
3],512,"Name","res5b_branch2a","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res5b_branch2a.Bias,"Weights",params.res5b_branch2a.Weights)

batchNormalizationLayer("Name","bn5b_branch2a","Offset",params.bn5b_branch2a.Offset,"Sca
le",params.bn5b_branch2a.Scale,"TrainedMean",params.bn5b_branch2a.TrainedMean,"TrainedVa
riance",params.bn5b_branch2a.TrainedVariance)
    reluLayer("Name","res5b_branch2a_relu")
    convolution2dLayer([3
3],512,"Name","res5b_branch2b","BiasLearnRateFactor",0,"Padding",[1 1 1
1],"Bias",params.res5b_branch2b.Bias,"Weights",params.res5b_branch2b.Weights)

batchNormalizationLayer("Name","bn5b_branch2b","Offset",params.bn5b_branch2b.Offset,"Sca
le",params.bn5b_branch2b.Scale,"TrainedMean",params.bn5b_branch2b.TrainedMean,"TrainedVa
riance",params.bn5b_branch2b.TrainedVariance)];
lgraph = addLayers(lgraph,tempLayers);

tempLayers = [
    additionLayer(2,"Name","res5b")
    reluLayer("Name","res5b_relu")
```

```
    globalAveragePooling2dLayer("Name","pool5")
    fullyConnectedLayer(2,"Name","fc")
    softmaxLayer("Name","softmax")
    classificationLayer("Name","classoutput")];
lgraph = addLayers(lgraph,tempLayers);

% clean up helper variable
clear tempLayers;
lgraph = connectLayers(lgraph,"pool1","res2a branch2a");
lgraph = connectLayers(lgraph,"pool1","res2a/in2");
lgraph = connectLayers(lgraph,"bn2a_branch2b","res2a/in1");
lgraph = connectLayers(lgraph,"res2a_relu","res2b_branch2a");
lgraph = connectLayers(lgraph,"res2a_relu","res2b/in2");
lgraph = connectLayers(lgraph,"bn2b_branch2b","res2b/in1");
lgraph = connectLayers(lgraph,"res2b_relu","res3a_branch1");
lgraph = connectLayers(lgraph,"res2b_relu","res3a_branch2a");
lgraph = connectLayers(lgraph,"bn3a_branch1","res3a/in2");
lgraph = connectLayers(lgraph,"bn3a_branch2b","res3a/in1");
lgraph = connectLayers(lgraph,"res3a_relu","res3b_branch2a");
lgraph = connectLayers(lgraph,"res3a_relu","res3b/in2");
lgraph = connectLayers(lgraph,"bn3b_branch2b","res3b/in1");
lgraph = connectLayers(lgraph,"res3b_relu","res4a_branch1");
lgraph = connectLayers(lgraph,"res3b_relu","res4a_branch2a");
lgraph = connectLayers(lgraph,"bn4a_branch2b","res4a/in1");
lgraph = connectLayers(lgraph,"bn4a_branch1","res4a/in2");
lgraph = connectLayers(lgraph,"res4a_relu","res4b_branch2a");
lgraph = connectLayers(lgraph,"res4a_relu","res4b/in2");
lgraph = connectLayers(lgraph,"bn4b_branch2b","res4b/in1");
lgraph = connectLayers(lgraph,"res4b_relu","res5a_branch2a");
lgraph = connectLayers(lgraph,"res4b_relu","res5a_branch1");
lgraph = connectLayers(lgraph,"bn5a_branch2b","res5a/in1");
lgraph = connectLayers(lgraph,"bn5a_branch1","res5a/in2");
lgraph = connectLayers(lgraph,"res5a_relu","res5b_branch2a");
lgraph = connectLayers(lgraph,"res5a_relu","res5b/in2");
lgraph = connectLayers(lgraph,"bn5b_branch2b","res5b/in1");
plot(lgraph);
```

### b) Entrenar red neuronal

Tras generar la red neuronal convolucional, es hora de entrenarla. El siguiente código muestra este entrenamiento. En este caso, en vez de emplear vectores 4d, se emplea el objeto de *imageDatastore* de Matlab para generar la base de datos de imágenes.

**Crear base de datos**

```
folder = '.\ImagenesCNN';
imds = imageDatastore(folder,"IncludeSubfolders",true,"LabelSource","foldernames");
```

**Mostrar el número de imágenes por etiqueta**

```
countEachLabel(imds)
```

**Dividir conjunto de entrenamiento**

```
[trainimds, validationimds] = splitEachLabel(imds,0.90, 'randomize');
```

**Graficar dataset**

```
numTrainImages = length(trainimds.Files);
figure;
```

```
perm = randperm(numTrainImages,20);

for i = 1:12
    subplot(3,4,i);
    imshow(trainimds.Files{perm(i)});
    if trainimds.Labels(perm(i))=='CONobstaculo'
        title('Con Obstaculo')
    else
        title('Sin Obstaculo')
    end
end
```

**Data augmentation**

```
imageAugmenter = imageDataAugmenter( ...
    'RandRotation',[-20,20], ...
    'RandXTranslation',[-10 10], ...
    'RandYTranslation',[-10 10]);
imageSize = [224 224 3];

augimds =
augmentedImageDatastore(imageSize,trainimds,'DataAugmentation',imageAugmenter);
```

**Opciones de entrenamiento**

```
miniBatchSize  = 128;% puede bajarse a 64
validationFrequency = floor(2500/miniBatchSize);
options = trainingOptions('sgdm', ...
    'MiniBatchSize',miniBatchSize, ...
    'MaxEpochs',200, ...
    'InitialLearnRate',1e-3, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.1, ...
    'LearnRateDropPeriod',20, ...
    'Shuffle','every-epoch', ...
    'ValidationData',validationimds, ...
    'ValidationFrequency',validationFrequency, ...
    'Plots','training-progress', ...
    'ExecutionEnvironment','gpu', ...
    'Verbose',false);
```

**Entrenar red**

```
net = trainNetwork(augimds,qq, options);
```

**Precisión Red**

```
YPredicted = classify(net,validationimds);
num test = length(YPredicted);
aciertos = 0;

for i=1:num_test
    if YPredicted(i) == validationimds.Labels(i)


        aciertos = aciertos + 1;
    end
end

precision = (aciertos/num_test)*100;
```

## B.3 Localización y seguimiento

a) Algoritmo de localización (método alternativo al mostrado en la solución final)

El próximo código muestra una de las variantes del algoritmo de localización desarrollado. Esta variante es algo más sencilla que la final, pero igual de efectiva. Simplemente es menos generalista que la última versión, ya que se emplea una ecuación para cada tipo de orientación posible. La última versión aúna estas ecuaciones.

**Configurar cámara**

```
cam = webcam();
cam.Resolution = '1920x1080';
```

**Cargar parámetros cámara**

```
load cameraParamsIker.mat
```

**Definir familia marcadores**

```
tagFamily = "tag36h11";
tagSize = 160.1; % mm
```

**Vector localización Apriltag para pruebas**

```
apriltag_loc=zeros(2,3);
% Apriltag ID=0
apriltag_loc(1,1)=140;
apriltag_loc(1,2)=1190;
apriltag_loc(1,3)=1;
% Apriltag ID=1
apriltag_loc(2,1)=290;
apriltag_loc(2,2)=0;
apriltag_loc(2,3)=0;
```

**Comenzar bucle detección marcadores**

```
h = figure;

while ishandle(h)

    I = snapshot(cam);

    [id,loc,pose,detectedFamily] = readAprilTag(I,tagFamily,params.Intrinsics,tagSize);

    worldPoints = [0 0 0; tagSize/2 0 0; 0 tagSize/2 0; 0 0 tagSize/2];
    if length(pose) >0
        for i = 1:length(pose)
            % Get image coordinates for axes.
            imagePoints = worldToImage(params.Intrinsics,pose(i).Rotation, ...
                          pose(i).Translation,worldPoints);

            % Draw colored axes.
            I = insertShape(I,"Line",[imagePoints(1,:) imagePoints(2,:); ...
                imagePoints(1,:) imagePoints(3,:); imagePoints(1,:) imagePoints(4,:)],
                "Color",["red","green","blue"],"LineWidth",7);

            I = insertText(I,loc(1,:,i),id(i),"BoxOpacity",1,"FontSize",25);
            id_nuevo=id(i)+1;
```

```
            eulers=rad2deg(rotm2eul(pose(i).Rotation));
            thita = eulers(2);
            x = pose(i).Translation(1);
            z = pose(i).Translation(3);

            xv=z*cos(deg2rad(thita))-x*sin(deg2rad(thita));
            yv=abs(z*sin(deg2rad(thita))+x*cos(deg2rad(thita)));

            % Calcular posición absoluta en función de la orientación del
            % marcador
            if apriltag_loc(id_nuevo,3)==0
                aux = xv;
                xv = yv;
                yv = aux;
                if eulers(2)>0
                    x_abs = apriltag_loc(id_nuevo,1)- xv;
                    y_abs = apriltag_loc(id_nuevo,2)+ yv;
                else
                    x_abs = apriltag_loc(id_nuevo,1)+ xv;
                    y_abs = apriltag_loc(id_nuevo,2)+ yv;
                end
            elseif apriltag_loc(id_nuevo,3)==1
                if eulers(2)>0
                    x_abs = apriltag_loc(id_nuevo,1)+ xv;
                    y_abs = apriltag_loc(id_nuevo,2)+ yv;
                else
                    x_abs = apriltag_loc(id_nuevo,1)+ xv;
                    y_abs = apriltag_loc(id_nuevo,2)- yv;
                end
            elseif apriltag_loc(id_nuevo,3)==2
                aux = xv;
                xv = yv;
                yv = auxv;
                if eulers(2)>0
                    x_abs = apriltag_loc(id_nuevo,1)+ xv;
                    y_abs = apriltag_loc(id_nuevo,2)- yv;
                else
                    x_abs = apriltag_loc(id_nuevo,1)- xv;
                    y_abs = apriltag_loc(id_nuevo,2)- yv;
                end
            elseif apriltag_loc(id_nuevo,3)==3

                if eulers(2)>0
                    x_abs = apriltag_loc(id_nuevo,1)- xv;
                    y_abs = apriltag_loc(id_nuevo,2)- yv;
                else
                    x_abs = apriltag_loc(id_nuevo,1)- xv;
                    y_abs = apriltag_loc(id_nuevo,2)+ yv;
                end
            end

            text=append('x: ', string(x), '; z: ', string(z),'; xv: ',string(xv),'; yv:
',string(yv),'; x abs: ',string(x_abs),'; y abs: ',string(y_abs),'; thita: ',
string(eulers(2)));
        end
        imshow(I)
    else
        text=('NADA');
    end

    title(text)
    drawnow

end
```

b) Graficar recorrido

Este código permite graficar el recorrido guardado en el código anterior. Tras cargar el recorrido realiza el filtrado de la señal y lo grafica sobre una imagen previa del entorno donde se está navegando.

**Cargar recorrido vehículo**

```
load('loc_labo_19-05_v1.mat');
```

**Contar número de posiciones del vehículo guardadas**

```
suma = 0;


for i=1:length(posicion_vehiculo)
    if posicion_vehiculo(i,1)~= 0 || posicion_vehiculo(i,2) ~= 0
        suma=suma+1;
    end
end
% Se crea un vector de actualizado de menos puntos
posicion_vehiculo_act=posicion_vehiculo(1:suma,:)/1000;
```

**Se realiza un filtro de media al recorrido**

```
num_ele_media=20;
posicion_vehiculo_act_media=zeros(suma-num_ele_media,2);
for i=1:suma-num_ele_media

posicion_vehiculo_act_media(i,1)=sum(posicion_vehiculo_act(i:i+num_ele_media,1))/num_ele
_media;

posicion_vehiculo_act_media(i,2)=sum(posicion_vehiculo_act(i:i+num_ele_media,2))/num_ele
_media;
end
```

**Calcular número de marcadores en el mapa**

```
load('apriltag_loc.mat');
num_markers=0;
for i=1:length(apriltag_loc)
    if apriltag_loc(i,3) ~= 0
        num_markers=num_markers+1;
    end
end
```

**Cargar imagen del laboratorio**

```
I = imread("LayoutLabo2.PNG");
imagesc([0 9],[6 -4],I)

hold on
```

**Graficar algoritmo localización**

```
% Graficar recorrido
plot(posicion_vehiculo_act_media(1:end,2),posicion_vehiculo_act_media(1:end,1),'LineWidt
h',4)
hold on
%
plot(posicion_vehiculo_act_media(floor(end/2):end,2),posicion_vehiculo_act_media(floor(e
nd/2):end,1),'g','LineWidth',2)

color_marcador_fuera='k';
color_marcador_dentro=[0.5 0.5 0.5];

% Graficar marcadores
```

```
for i=1:length(apriltag_loc)
    if apriltag_loc(i,3) ~= 0
        plot(apriltag loc(i,2)/1000,apriltag loc(i,1)/1000,'s',...
        'LineWidth',2,...
        'MarkerSize',10,...
        'MarkerEdgeColor',color_marcador_fuera,...
        'MarkerFaceColor',color_marcador_dentro) % 'r'
    end
end

% Graficar Inicio
plot(posicion_vehiculo_act_media(1,2),posicion_vehiculo_act_media(1,1),'ko',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor',[0,1,0])
% Graficar Final
plot(posicion_vehiculo_act_media(end,2),posicion_vehiculo_act_media(end,1),'ko',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor',[1,0,0])

% Graficar Origen de coordenadas
plot(0,0,'kp',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','b',...
    'MarkerFaceColor','c')

%
legend('Recorrido','Marcadores','','','','','INICIO','FINAL','Location','southeast',...
%     'FontSize',16,'TextColor','blue')
%
title('Trayectoria Vehículo','FontSize',20)

%Grafico sin filtrar
%plot(posicion_vehiculo_act(1:end,1),posicion_vehiculo_act(1:end,2))

% set the y-axis back to normal.
set(gca,'ydir','normal', 'visible', 'off');
```

c) Filtrar señal y graficar recorrido

Este código permite visualizar los resultados de seguimiento y mapeo del código anterior, pero empleando otros parámetros de filtrado.

**Cargar posiciones del vehículo**

```
load('posicion_vehiculo_labo14.mat');
```

**Filtrar señal**

```
suma = 0;
for i=1:length(posicion_vehiculo)
    if posicion_vehiculo(i,1)~= 0 && posicion_vehiculo(i,2) ~= 0
        suma=suma+1;
    end
end
% Se crea un vector de actualizado de menos puntos
posicion_vehiculo_act=posicion_vehiculo(1:suma,:)/1000;

umbral_diferencia = 0.75;

for i =1:length(posicion_vehiculo_act)
    if i==1
```

```
        pos_anterior_x = posicion_vehiculo_act(i,1);
        pos_anterior_y = posicion_vehiculo_act(i,2);
    else
        diferencia=abs(pos_anterior_x*umbral_diferencia);
        diferencia=1.75;
        if diferencia < abs(pos_anterior_x-posicion_vehiculo_act(i,1)) || diferencia <
abs(pos_anterior_y-posicion_vehiculo_act(i,2))
            disp(i)
            posicion vehiculo act(i,1)=0;
            posicion_vehiculo_act(i,2)=0;
        else
            pos_anterior_x = posicion_vehiculo_act(i,1);
            pos_anterior_y = posicion_vehiculo_act(i,2);
        end
    end
end

suma = 0;
for i=1:length(posicion_vehiculo_act)
    if posicion_vehiculo_act(i,1)~= 0 || posicion_vehiculo_act(i,2) ~= 0
        suma=suma+1;
    end
end

posicion_vehiculo_act_2=zeros(suma,2);
for i=1:length(posicion_vehiculo_act)
    if posicion_vehiculo(i,1)~= 0 || posicion_vehiculo(i,2) ~= 0
        posicion_vehiculo_act_2(i,1)=posicion_vehiculo_act(i,1);
        posicion_vehiculo_act_2(i,2)=posicion_vehiculo_act(i,2);
    end
end

num_ele_media=20;
posicion_vehiculo_act_media=zeros(suma-num_ele_media,2);
for i=1:suma-num_ele_media

posicion_vehiculo_act_media(i,1)=sum(posicion_vehiculo_act_2(i:i+num_ele_media,1))/num_e
le_media;

posicion_vehiculo_act_media(i,2)=sum(posicion_vehiculo_act_2(i:i+num_ele_media,2))/num_e
le_media;
end

load('apriltag_loc.mat');
num_markers=0;
for i=1:length(apriltag_loc)
    if apriltag_loc(i,3) ~= 0
        num_markers=num_markers+1;
    end
end

I = imread("LayoutLabo2.PNG");
imagesc([0 9],[6 -4],I)

hold on

% Graficar recorrido
plot(posicion_vehiculo_act_media(1:end,2),posicion_vehiculo_act_media(1:end,1),'LineWidt
h',4)
hold on
%
plot(posicion_vehiculo_act_media(floor(end/2):end,2),posicion_vehiculo_act_media(floor(e
nd/2):end,1),'g','LineWidth',2)

color marcador fuera='k';
color_marcador_dentro=[0.5 0.5 0.5];

% Graficar marcadores
for i=1:length(apriltag_loc)
    if apriltag_loc(i,3) ~= 0
        plot(apriltag loc(i,2)/1000,apriltag loc(i,1)/1000,'s',...
        'LineWidth',2,...
        'MarkerSize',10,...
        'MarkerEdgeColor',color_marcador_fuera,...
        'MarkerFaceColor',color_marcador_dentro) % 'r'
```

```
    end
end

% Graficar Inicio
plot(posicion_vehiculo_act_media(1,2),posicion_vehiculo_act_media(1,1),'ko',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor',[0,1,0])

% Graficar Final
plot(posicion_vehiculo_act_media(end,2),posicion_vehiculo_act_media(end,1),'ko',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor',[1,0,0])

% Graficar Origen de coordenadas
plot(0,0,'kp',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','b',...
    'MarkerFaceColor','c')

%
legend('Recorrido','Marcadores','','','','','INICIO','FINAL','Location','southeast',...
%     'FontSize',16,'TextColor','blue')
%
title('Trayectoria Vehículo','FontSize',20)

%Grafico sin filtrar
%plot(posicion_vehiculo_act(1:end,1),posicion_vehiculo_act(1:end,2))

% set the y-axis back to normal.
set(gca,'ydir','normal', 'visible', 'off');
```

## B.4 Sistema completo

a) Código completo: 3 algoritmos y movimiento del vehículo

Por último, se muestra el código que ejecuta los tres algoritmos simultáneamente. A medida que se mueve el vehículo, se muestran tres figuras: una correspondiente al algoritmo de seguimiento de líneas, una para la detección de obstáculos, y otra para el seguimiento y mapeo del vehículo.

**Inicialización**

```
clear all
close all
```

**Cargar redes neuronales**

```
% Red neuronal cámara inferior (seguimiento de líneas)
load('dl_ikerresnet18.mat');
% Red neuronal cámara superior (obstáculos)
load('net_obstaculo.mat');
```

**Cargar posición marcadores en el laboratorio**

```
load("apriltag_loc.mat");
```

**Cargar parámetros cámara**

```
load cameraParamsIker.mat
```

**Definir familia y tamaño marcadores**

```
tagFamily = "tag36h11";
tagSize = 160.1; % mm
```

**Vector posiciones**

```
posicion_vehiculo = zeros(10000,2);
j=1;
pause(1)
```

**Parámetros filtrado señal**

```
% Número de elementos de la media móvil para graficar
num_ele_media = 20;

% Umbral para eliminar valor
umbral_diferencia = 1.5;
```

**Parámetros comunicación Beckhoff**

```
% Funciones
addpath('Win64Mex'); %Para 64 bits

% Configuración AGV
IP = '169.254.153.119.1.1'; % Dirección IP
PUERTO = 851; % Puerto
```

**Establecer comunicación con AGV**

```
% Conexión con la cámara inferior
h = Write_BOOL(IP, PUERTO, 'GVL_Matlab.bStart', true);
```

**Conexiones con las cámaras**

```
% Conexión con la cámara inferior
cam1 = webcam(2);
% Conexión con la cámara superior
cam2 = webcam(3);
cam2.Resolution = '1920x1080';
```

**Parámetros de control del Vehículo**

```
Kp = 6;     % Ganancia proporcional
Kd = 12;    % Ganancia derivativa
Kv = 8;    % Ganancia velocidad
v_max = 14;
v min = 4;
error_anterior = 0;
v_R = 0;
v_L = 0;
```

**Bucle principal**

```
h1 = figure('Position', [0 300 640 480], 'Name','Navegacion mediante deteccion de
lineas','NumberTitle','off');
h2 = figure('Position', [640 300 640 480], 'Name','Deteccion de
personas','NumberTitle','off');
h3 = figure('Position', [1280 300 640 480], 'Name','Localic¡zacion y


mapeo','NumberTitle','off');

% Mapear localización
figure(3)
% Cargar imagen laboratorio
I = imread("LayoutLabo2.PNG");
imagesc([0 9],[6 -4],I)
% set the y-axis back to normal.
set(gca,'ydir','normal', 'visible', 'off');
hold on

% Umbrales de deteccion
Num_Obstaculo = 0;
Num_Linea = 0;

color_marcador_fuera='k';
color_marcador_dentro=[0.5 0.5 0.5];
% Graficar marcadores
for i=1:length(apriltag_loc)
    if apriltag_loc(i,2) ~= 0 && apriltag_loc(i,1) ~= 0
        plot(apriltag_loc(i,2)/1000,apriltag_loc(i,1)/1000,'s',...
        'LineWidth',2,...
        'MarkerSize',10,...
        'MarkerEdgeColor',color_marcador_fuera,...
        'MarkerFaceColor',color_marcador_dentro) % 'r'
    end
end

while ishandle(h1)
    if double(get(gcf,'CurrentCharacter')) == 113 % check if 'q' key pressed, change to
27 for escape
        break; % break from while loop
    end

    % Cámara 2
    im2 = snapshot(cam2);
    im2_resized = imresize(im2,[224 224]);
    YPred = classify(net,im2_resized);
    figure(2)
    image(im2)
    drawnow
    figure(1)
    if YPred=='CONobstaculo'
        Num_Obstaculo = Num_Obstaculo +1;
        if Num_Obstaculo >= 5
            h = Write_REAL(IP, PUERTO, 'GVL_Matlab.V_rueda_R', 0);
            h = Write_REAL(IP, PUERTO, 'GVL_Matlab.V_rueda_L', 0);
            figure(2)
            rectangle('Position',[1,1,1910,1075],'EdgeColor', 'r', 'LineWidth', 10)
            title('CON obstaculo')
        else
            h = Write_REAL(IP, PUERTO, 'GVL_Matlab.V_rueda_R', v_R);
            h = Write_REAL(IP, PUERTO, 'GVL_Matlab.V_rueda_L', v_L);
            figure(2)
            title('SIN obstaculo')
        end
    else
        % Cámara 1
        Num_Obstaculo = 0;
        figure(1)
        im = snapshot(cam1);
        im = imresize(im,[56 56]);
        im = im2double(im);
        image(im)
        X = dlarray(im,"SSCB");
```

```
        [pos, flag] = predict(dl_ikerresnet18,X);
        flag=extractdata(flag);
        pos=extractdata(pos);
        prediccion_xy = pos;
        if flag(1)>0.5
            Num_Linea = 0;
            [v_R, v_L, error] = consignas_vehiculo_PD(prediccion_xy, Kp, Kd, Kv, v_max,
v_min, error_anterior);

            error_anterior = error;
            h = Write_REAL(IP, PUERTO, 'GVL_Matlab.V_rueda_R', v_R);
            h = Write_REAL(IP, PUERTO, 'GVL_Matlab.V_rueda_L', v_L);
            title('Linea')
            hold on
            plot(prediccion xy(1),prediccion xy(2), 'r.', 'MarkerSize', 10, 'LineWidth',
2)

            % Algoritmo de localización
            [id,loc,pose,detectedFamily] =
readAprilTag(im2,tagFamily,params.Intrinsics,tagSize);
            worldPoints = [0 0 0; tagSize/2 0 0; 0 tagSize/2 0; 0 0 tagSize/2];

            if length(pose) >0
                for i = 1:length(pose)
                    % Get image coordinates for axes.
                    imagePoints = worldToImage(params.Intrinsics,pose(i).Rotation, ...
                                  pose(i).Translation,worldPoints);

                    % Draw colored axes.
                    I = insertShape(I,"Line",[imagePoints(1,:) imagePoints(2,:); ...
                        imagePoints(1,:) imagePoints(3,:); imagePoints(1,:)
imagePoints(4,:)], "Color",["red","green","blue"],"LineWidth",7);

                    I = insertText(I,loc(1,:,i),id(i),"BoxOpacity",1,"FontSize",25);

                    id_nuevo=id(i);

                    eulers=rad2deg(rotm2eul(pose(i).Rotation));
                    thita = eulers(2);
                    x = pose(i).Translation(1);
                    z = pose(i).Translation(3);

                    % Calcular posición absoluta en función de la orientación del
                    % marcador
                    if apriltag_loc(id_nuevo,3)==0
                        thita = thita + 0;

                    elseif apriltag_loc(id_nuevo,3)==1
                        thita = thita - 90;

                    elseif apriltag_loc(id_nuevo,3)==2
                        thita = thita - 180;

                    elseif apriltag_loc(id_nuevo,3)==3
                        thita = thita - 270;

                    end

                    xv=z*sin(deg2rad(thita))+x*cos(deg2rad(thita));
                    yv=z*cos(deg2rad(thita))-x*sin(deg2rad(thita));

                    x_abs = apriltag_loc(id_nuevo,1)- xv;
                    y_abs = apriltag_loc(id_nuevo,2)+ yv;

                    posicion vehiculo(j,1)=x abs/1000;
                    posicion_vehiculo(j,2)=y_abs/1000;

                    % Filtrado senal - Eliminar valores dispersos
                    if j==1
                        pos_anterior_x = posicion_vehiculo(j,1);
                        pos anterior y = posicion vehiculo(j,2);
                        j=j+1;

                    else
```

```
                        if umbral_diferencia < abs(pos_anterior_x-
posicion_vehiculo(j,1)) || umbral_diferencia < abs(pos_anterior_y-
posicion vehiculo(j,2))

                        else
                            pos_anterior_x = posicion_vehiculo(j,1);
                            pos_anterior_y = posicion_vehiculo(j,2);

                            % FIltrado senal 2 - Media movil de num ele media
                            if j>=num_ele_media
                                posicion_vehiculo_act_media_x=sum(posicion_vehiculo(j-
num_ele_media+1:j,1))/num_ele_media;
                                posicion_vehiculo_act_media_y=sum(posicion_vehiculo(j-
num_ele_media+1:j,2))/num_ele_media;
                                if j ==num ele media
                                    primera_posicion_x=posicion_vehiculo_act_media_x;
                                    primera_posicion_y=posicion_vehiculo_act_media_y;
                                    % Graficar Inicio
                                    figure(3)
                                    plot(primera_posicion_y,primera_posicion_x,'ko',...
                                        'LineWidth',2,...
                                        'MarkerSize',10,...
                                        'MarkerEdgeColor','k',...
                                        'MarkerFaceColor',[0,1,0])
                                    hold on
                                end
                                figure(3)
                                % Graficar recorrido
                    plot(posicion_vehiculo_act_media_y,posicion_vehiculo_act_media_x,'ko',
                                    'LineWidth',1,...
                                    'MarkerSize',2,...
                                    'MarkerEdgeColor','b',...
                                    'MarkerFaceColor',[0,0,1])
                                hold on
                            end
                            j=j+1;
                        end
                    end
                end
            end
        else
            Num_Linea = Num_Linea + 1;
            if Num_Linea >= 10
                h = Write_REAL(IP, PUERTO, 'GVL_Matlab.V_rueda_R', v_R);
                h = Write_REAL(IP, PUERTO, 'GVL_Matlab.V_rueda_L', v_L);
                title('No Linea')
                break;
            end
        end
    end
    drawnow
end

figure(3)
% Graficar Final
plot(posicion_vehiculo_act_media_y,posicion_vehiculo_act_media_x,'ko',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor',[1,0,0])

%
legend('Recorrido','Marcadores','','','','','INICIO','FINAL','Location','southeast',...
%     'FontSize',16,'TextColor','blue')
%
title('Trayectoria Vehículo','FontSize',20)
```

**Cerrar conexión con AGV**

```
h = Write_REAL(IP, PUERTO, 'GVL_Matlab.V_rueda_R', v_R);
h = Write_REAL(IP, PUERTO, 'GVL_Matlab.V_rueda_L', v_L);
h = Write_BOOL(IP, PUERTO, 'GVL_Matlab.bStart', false);
```

b) Calcular consignas del vehículo con controlador PD

Este código muestra la función que calcula las consignas que se envían a las ruedas del AGV. Estos valores se calculan a través de un controlador PD.

```
function [v_R, v_L, error] = consignas_vehiculo_PD(prediccion_xy, Kp, Kd, Kv, v_max,
v_min, error_anterior)


x = prediccion_xy(1);
y = prediccion_xy(2);

% v es la velocidad base del robot, la cual depende de lo lejos que esté el
% punto que debe seguir el robot

v = Kv * (56-y)/56;

if v > v_max
    v = v_max;
elseif v < v_min
    v = v_min;
end

error = atan2(x-28,56-y);

% Controlador PD
PD = Kp*error+Kd*(error-error_anterior);

v_R = min(18, max(v + PD,0));
v_L = min(18, max(v - PD,0));
```

c) Calcular consignas del vehículo con controlador proporcionas (diferente función de error)

Este código muestra una forma alternativa para calcular las consignas que se envían a las ruedas del vehículo.

```
function [vel, ang, signo] = consignas_vehiculo(prediccion_xy, K1, K2, error_max,
vel_max, vel_min)

prediccion_x = prediccion_xy(1);
prediccion_y = prediccion_xy(2);
vel = K1 * abs(prediccion_y-56)/56;

if vel > vel max
    vel = vel_max;
elseif vel < vel_min
    vel = vel_min;
else
    vel = vel;
end

error_angulo = K2 * ((prediccion_x - 28)/28);

if error_angulo > error_max
    error_angulo = error_max;
elseif error_angulo < -(error_max)
    error_angulo = -error_max;
end

ang = - 0.75 * error_angulo;

if ang >= 0
    ang = ang;
    signo = 0;
else
    ang = abs(ang);
    signo = 1;
end
```

d) Filtrar señal (última versión)

Este código es la última versión del código para filtrar la señal visto anteriormente.

**Cargar vector posiciones del vehículo**

```
load('loc_labo_27-05_v4.mat');
```

**Realizar filtrado de la señal**

```
% Contar número de posiciones del vehículo guardadas
suma = 0;
for i=1:length(posicion_vehiculo)
    if posicion_vehiculo(i,1)~= 0 && posicion_vehiculo(i,2) ~= 0
        suma=suma+1;
    end
end
% Se crea un vector de actualizado de menos puntos
posicion_vehiculo_act=posicion_vehiculo(1:suma,:);

umbral_diferencia = 1;
num_ele_media=20;

for i =1:length(posicion_vehiculo_act)
    if i==1
        pos_anterior_x = posicion_vehiculo_act(i,1);
        pos_anterior_y = posicion_vehiculo_act(i,2);
    else
        if umbral_diferencia < abs(pos_anterior_x-posicion_vehiculo_act(i,1)) ||
umbral_diferencia < abs(pos_anterior_y-posicion_vehiculo_act(i,2))
            disp(i)
            posicion_vehiculo_act(i,1)=0;
            posicion_vehiculo_act(i,2)=0;

        else
            pos anterior x = posicion vehiculo act(i,1);
            pos_anterior_y = posicion_vehiculo_act(i,2);
        end
    end
end

suma = 0;
for i=1:length(posicion_vehiculo_act)
    if posicion_vehiculo_act(i,1)~= 0 && posicion_vehiculo_act(i,2) ~= 0
        suma=suma+1;
    end
end

posicion_vehiculo_act_2=zeros(suma,2);
j=1;
for i=1:length(posicion_vehiculo_act)
    if posicion_vehiculo_act(i,1)~= 0 && posicion_vehiculo_act(i,2) ~= 0
        posicion_vehiculo_act_2(j,1)=posicion_vehiculo_act(i,1);
        posicion_vehiculo_act_2(j,2)=posicion_vehiculo_act(i,2);
        j=j+1;
    end
end

posicion_vehiculo_act_media=zeros(suma-num_ele_media,2);
for i=1:suma-num_ele_media
    posicion_vehiculo_act_media(i,1)=sum(posicion_vehiculo_act_2(i:i+num_ele_media-
1,1))/num_ele_media;
    posicion_vehiculo_act_media(i,2)=sum(posicion_vehiculo_act_2(i:i+num_ele_media-
1,2))/num_ele_media;
end

load('apriltag_loc.mat');
num_markers=0;
for i=1:length(apriltag_loc)
    if apriltag_loc(i,3) ~= 0
```

```
        num_markers=num_markers+1;
    end
end

I = imread("LayoutLabo2.PNG");
imagesc([0 9],[6 -4],I)

hold on

% Graficar recorrido
plot(posicion_vehiculo_act_media(1:end,2),posicion_vehiculo_act_media(1:end,1),'LineWidt
h',4)
hold on
%
plot(posicion vehiculo act media(floor(end/2):end,2),posicion vehiculo act media(floor(e
nd/2):end,1),'g','LineWidth',2)

color_marcador_fuera='k';
color_marcador_dentro=[0.5 0.5 0.5];

% Graficar marcadores
for i=1:length(apriltag_loc)
    if apriltag_loc(i,2) ~= 0 || apriltag_loc(i,1) ~=0
        plot(apriltag_loc(i,2)/1000,apriltag_loc(i,1)/1000,'s',...
        'LineWidth',2,...
        'MarkerSize',10,...
        'MarkerEdgeColor',color_marcador_fuera,...
        'MarkerFaceColor',color_marcador_dentro) % 'r'
    end
end

% Graficar Inicio
plot(posicion_vehiculo_act_media(1,2),posicion_vehiculo_act_media(1,1),'ko',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor',[0,1,0])
% Graficar Final
plot(posicion_vehiculo_act_media(end,2),posicion_vehiculo_act_media(end,1),'ko',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','k',...
    'MarkerFaceColor',[1,0,0])

% Graficar Origen de coordenadas
% plot(0,0,'kp',...
%     'LineWidth',2,...
%     'MarkerSize',10,...
%     'MarkerEdgeColor','b',...
%     'MarkerFaceColor','c')

%
legend('Recorrido','Marcadores','','','','','INICIO','FINAL','Location','southeast',...
%     'FontSize',16,'TextColor','blue')
%
title('Trayectoria Vehículo','FontSize',20)

%Grafico sin filtrar
%plot(posicion_vehiculo_act(1:end,1),posicion_vehiculo_act(1:end,2))

% set the y-axis back to normal.
set(gca,'ydir','normal', 'visible', 'off');
```

# Manual de usuario

# Anexo C

Para finalizar con el documento se van a explicar ciertas consideraciones que hay que cumplir para poder emplear el algoritmo de navegación desarrollado.

## C.1 Matlab

Como se ha comentado anteriormente, se está empleado la plataforma de programación Matlab. Sin embargo, para poder ejecutar correctamente los códigos mostrados en el Anexo B será necesario tener ciertas herramientas adicionales:

- Deep Learning Toolbox
- Computer Vision Toolbox
- Image Processing Toolbox
- MATLAB Support Package for USB Webcams

Así como los paquetes correspondientes a las CNN empleadas:

- Deep Learning Toolbox Model for VGG-19 Network
- Deep Learning Toolbox Model for ResNet-18 Network
- Deep Learning Toolbox Model for ResNet-50 Network
- Deep Learning Toolbox Model for DenseNet-201 Network
- Deep Learning ToolboxTM Model for ShuffleNet Network
- Deep Learning Toolbox Model for MobileNet-v2 Network
- Deep Learning Toolbox Model for GoogLeNet Network
- Deep Learning ToolboxTM Model for EfficientNet-b0 Network
- Deep Learning Toolbox Model for Inception-v3 Network
- Deep Learning ToolboxTM Model for Xception Network

Además, se ha empleado una versión de Matlab igual o superior a la 2021(a).

## C.2 Uso de código

Para hacer uso de los algoritmos desarrollados durante este trabajo, será necesario ejecutar los códigos del Anexo B. Sin embargo, para que estos códigos funcionen correctamente, será necesario tener acceso a las imágenes empleadas para el entrenamiento, así como a las redes neuronales convolucionales entrenadas durante este trabajo. Además, es necesario disponer de

unas funciones específicas de comunicación de Beckhoff con Matlab para poder mover el AGV. Por este motivo, se ha creado un repositorio donde están disponibles todos estos archivos.

https://drive.google.com/drive/folders/1aVFrLNuZuqyye3J2R15dUN-JL_z6eM9W?usp=sharing

Por un lado, los códigos de los Anexos B.1, B.2 y B.3 ejecutan cada uno de los algoritmos por separado. Por otro lado, el código correspondiente al Anexo B.4 ejecuta un programa en el que se mueve el AGV y se ejecutan los tres algoritmos juntos. Para que este último funcione correctamente, será necesario tener en la misma carpeta que el código:

- CNNs:
    - Algoritmo para el seguimiento de líneas
    - Algoritmo para la detección de obstáculos
- Funciones de comunicación Beckhoff con Matlab
- Localización de los marcadores fiduciarios
- Parámetros de la cámara
- Función que calcula las consignas del vehículo
- Imagen sobre la que se grafica el recorrido del vehículo (mapa del entorno)