EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de La Laguna

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

"Control de Movimiento de Robots Usando Técnicas de Aprendizaje Automático"

POR

M.C. Sergio López Hernández

TESIS

PRESENTADO COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE DOCTOR EN CIENCIAS EN INGENIERÍA ELÉCTRICA

DIRECTOR DE TESIS

Dr. Miguel Ángel Llama Leal

CODIRECTOR DE TESIS

Dr. Ramón García Hernández

ISSN: 0188-9060

RIITEC: (08)-TDCIE-2022

Torreón, Coahuila. México

Diciembre 2022

Instituto Tecnológico de La Laguna

DEPTO: DIV. EST. POSG. E INVEST.
Dependencia: DEPI/CPCIE
NO. OFICIO: DEPI/CPCIE/492/2022
ASUNTO: Autorización de Impresión
´Torreón Coah., 30. Nov. 2022

M.C. SERGIO LÓPEZ HERNÁNDEZ
C. CANDIDATO AL GRADO DE DOCTOR
EN CIENCIAS EN INGENIERÍA ELÉCTRICA.
PRESENTE.-

Después de haber sometido a revisión su trabajo de tesis titulado**:**

"Control de Movimiento de Robots Usando Técnicas de Aprendizaje Automático"

Habiendo cumplido con todas las indicaciones que el jurado revisor de tesis hizo, se le comunica que se le concede la autorización con número de registro **RIITEC: (08)-TDCIE.2022,** para que proceda a la impresión del mismo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Educación Tecnológica Fuente de Innovación

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLOGICO DE LA LAGUNA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

DR. JOSÉ IRVING HERNÁNDEZ JÁCQUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO
DE INVESTIGACIÓN

JIHJ/*mjsl

Instituto Tecnológico de La Laguna

Torreón Coah., 30/Noviembre/2022

DR. JOSÉ IRVING HERNÁNDEZ JACQUEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Por medio de la presente, hacemos de su conocimiento que después de haber sometido a revisión el trabajo de tesis titulado:

" Control de Movimiento de Robots Usando Técnicas de Aprendizaje Automático"

Desarrollado por el **C. Sergio López Hernández**, con número de control **D1813009** y habiendo cumplido con todas las correcciones que se le indicaron, estamos de acuerdo que se le conceda la autorización de la fecha de examen de grado para que proceda a la impresión de la misma.

ATENTAMENTE
EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN

Dr. Miguel Ángel Llama Leal
Asesor/Director de Tesis

Dr. Ramón García Hernández
Co-Asesor/Co-Director de Tesis

Dr. Juan Sifuentes Mijares
Comité Tutorial

Dr. Francisco Jurado Zamarripa
Comité Tutorial

Dr. Francisco Guillermo Salas Pérez
Comité Tutorial Externo

“No hiciste caso a mis palabras, ¿verdad? Transmite lo que has aprendido. Fuerza. Dominio. Pero también debilidad, locura, fracaso. Sí, el fracaso sobre todo. El mejor maestro, el fracaso es. Luke, somos lo que ellos cultivan más allá. Esa es la verdadera carga de todos los maestros.”

– Maestro Yoda

Dedicatoria

A mis padres, Sergio y Carmen, por apoyarme en cada proyecto y por todo el cariño que me tienen.

A mis hermanos Leticia, Carmen, Sandra y Adrián, por su apoyo incondicional.

Al Dr. Ricardo E. Campa Cocom que en paz descanse, quien siempre fue un gran ejemplo en todo sentido y una gran inspiración.

Agradecimientos

A mis padres, por su orientación y consejo.

Al Dr. Miguel A. Llama Leal, por guiarme en el desarrollo de este trabajo, por su paciencia, consejo, apoyo y sobre todo su confianza.

A mis sinodales, Dr. Ramón García Hernández, Dr. Francisco Jurado Zamarripa y Dr. Juan Sifuentes Mijares, por su consejo y apoyo.

A la señorita María Crystina García Barrera, por toda la paciencia, comprensión y el apoyo.

A todos mis compañeros y excompañeros del posgrado, quienes siempre estuvieron dispuestos a ayudarme.

A mis amigos Carolina y Guillermo, por ser mis amigos incondicionales.

Finalmente agradezco al CONACyT por brindarme las facilidades para el desarrollo de mis estudios de doctorado.

Resumen

En el presente trabajo se diseñan controladores neuro-adaptables de una capa basados en filtrado del error para resolver el problema de seguimiento de trayectoria. Las redes neuronales se utilizan para aproximar las funciones no lineales desconocidas de la planta; posteriormente, los pesos de las redes neuronales se estiman utilizando algoritmos de actualización de pesos en línea. Los controladores diseñados se implementan de manera experimental tanto en un robot manipulador como en un robot móvil omnidireccional. Además de lo anterior, se desarrolla una plataforma de navegación y detección de obstáculos, la cual requiere distintos tipos de sensores y unidades de procesamiento. La tarea de esta plataforma es detectar y ejecutar una ruta utilizando una cámara a bordo, luego se implementan recursos para detectar posibles obstáculos y tomar acciones en consecuencia. Se utilizan recursos como aprendizaje profundo, proyección de coordenadas, generador de trayectorias y control cinemático. La plataforma desarrollada se implementa tanto en simulación como en tiempo real.

Abstract

In this work, single-layer neuro-adaptive controllers based on filtered error are designed to solve the motion trajectory tracking problem. Neural networks are used to approximate the unknown nonlinear plant functions; subsequently, the neural networks weights are estimated using online weight update algorithms. The designed controllers are experimentally implemented on both a manipulator robot and an omnidirectional mobile robot. In addition to the above, a navigation and obstacle detection platform is developed, which requires different types of sensors and processing units. The task of this platform is to detect and execute a route using an onboard camera, then resources are implemented to detect possible obstacles and take actions accordingly. Resources such as deep learning, coordinate projection, trajectory generator, and kinematic control are used. The developed platform is implemented both in simulation and in real-time.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

En general, cuando se quiere controlar un robot se diseña un sistema de control para cada tarea en específico que se quiere que un robot ejecute. Incluso para tareas que una persona promedio puede realizar de manera intuitiva, como agarrar objetos o cortar alimentos, estos controladores pueden ser muy difíciles de diseñar porque no hay manera de traducir fácilmente esta intuición natural en código. Asimismo puede ser extremadamente desafiante escalar estos enfoques a la gran cantidad de desafíos con los que estos robots deben lidiar en un ambiente no controlado [1].

[1]: Lenz (2016), "Deep learning for robotics"

Por lo anterior, los algoritmos de aprendizaje automático han tenido un uso generalizado para aplicaciones de robótica. En lugar de obligar al ingeniero a codificar manualmente un sistema robótico completo, el aprendizaje automático permite aprender partes de este sistema a partir de algunos datos de entrenamiento. Este enfoque nos permite modelar conceptos que pueden ser difíciles o imposibles de modelar matemáticamente. Además de contar con algoritmos adaptables, siempre que la estructura del algoritmo sea general, puede adaptarse a nuevos casos simplemente proporcionando datos de entrenamiento para éstos.

Por su parte, cuando se habla de controladores neuronales, hay que distinguir entre los que identifican la planta y los de realimentación de estados. El control por realimentación de estados ofrece la posibilidad de implementar algoritmos de aprendizaje en línea que no requieren ajustes preliminares fuera de línea [2-4]. Adicionalmente, existe un gran interés por los controladores que no requieren el modelo de la planta para su funcionamiento (*model-free controllers*), esto se puede conseguir con diferentes técnicas, como lógica difusa, imitación de las funciones lingüísticas y de razonamiento, y las redes neuronales (RNs) artificiales [5, 6].

[2]: Lewis y col. (2006), "Neural Networks in Feedback Control Systems"
[3]: Lewis (1999), "Nonlinear network structures for feedback control"
[4]: Dawson y col. (2003), *Robot manipulator control: theory and practice*
[5]: Lewis y col. (1998), *Neural Network Control of Robot Manipulators and Nonlinear Systems*
[6]: Farrell y col. (2006), *Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches*

1.1 Antecedentes

A continuación se mencionan algunos trabajos que se refieren al control neuronal por realimentación que no requieren

etapa de aprendizaje fuera de línea. En [3] se expone una estructura para el diseño de controladores usando RNs y lógica difusa; se presentan arquitecturas de control por realimentación y algoritmos de sintonización de pesos para asegurar la estabilidad en lazo cerrado y el acotamiento de los pesos. En [7] se propone un controlador con precompensación neuronal basado en leyes adaptables para conseguir el control robusto de robots manipuladores de eslabones rígidos; se incluye un estudio de simulación comparativa con diferentes controladores robustos y adaptables. En [8] se diseña un esquema de control con compensación neuro-adaptable (N-A) para un giroscopio de sistema microelectromecánico; el desempeño del controlador se valida en simulación. En [9] se implementan diferentes controladores para seguimiento de trayectorias aplicado a un giroscopio de dos grados de libertad subactuado. Entre los controladores analizados se tiene un controlador por realimentación de estados, un algoritmo neuro-adaptable, un esquema adaptable por modelo del regresor, un control PD clásico y un controlador PID-PID en cascada; las propuestas se evalúan de manera experimental. En [10] se presenta un control N-A para resolver el problema de control de un péndulo de Furuta, también se hace una comparación con otros esquemas de manera experimental. En [11] se propone una estrategia de control N-A para robots manipuladores no lineales, además, se aplican técnicas de control por modos deslizantes. En [12] se propone un esquema de control N-A para una clase de robots manipuladores inciertos con perturbaciones externas y restricciones de salida variables en el tiempo; se utiliza la técnica de *backstepping* para diseñar el controlador. En [13] se presenta un control N-A para el seguimiento de la trayectoria de robots manipuladores, además, se propone una estrategia para obtener el número óptimo de capas ocultas. En [14] se propone un esquema de control de impedancia de seguimiento de fuerza N-A; se aplican observadores no lineales para el diseño del controlador. En [15] se propone una estrategia de control N-A emocional robusto y se implementa en un robot paralelo esférico-prismático-esférico en tiempo real. En [16] se diseña un control N-A para un robot con incertidumbres y restricciones de estado completo; se realizan simulaciones para evaluar el desempeño del control. En [17] se diseña un control adaptable multimodelo basado en RNs difusas; para evaluar su rendimiento se aplicaron múltiples cambios en los parámetros del sistema. En [18]

[7]: Kwan y col. (1995), "Robust adaptive control of robots using neural network: global tracking stability"

[8]: Wang y col. (2019), "Adaptive control of micro-electromechanical system gyroscope using neural network compensator"

[9]: Montoya–Cháirez y col. (2019), "Adaptive control schemes applied to a control moment gyroscope of 2 degrees of freedom"

[10]: Moreno-Valenzuela y col. (2016), "Adaptive Neural Network Control for the Trajectory Tracking of the Furuta Pendulum"

[11]: Luan y col. (2019), "Adaptive neural network control for robotic manipulators with guaranteed finite-time convergence"

[12]: Wu y col. (2019), "Adaptive neural network control of uncertain robotic manipulators with external disturbance and time-varying output constraints"

[13]: Liu y col. (2019), "Adaptive neural network control with optimal number of hidden nodes for trajectory tracking of robot manipulators"

[14]: Yang y col. (2019), "Adaptive neural network force tracking impedance control for uncertain robotic manipulator based on nonlinear velocity observer"

[15]: Baghbani y col. (2020), "Emotional neural networks with universal approximation property for stable direct adaptive nonlinear control systems"

[16]: He y col. (2016), "Adaptive Neural Network Control of an Uncertain Robot With Full-State Constraints"

[17]: Li y col. (2014), "Multimodel adaptive control based on fuzzy neural networks"

se diseña un controlador N-A multicapa y se realizan pruebas en simulación en sistemas no lineales conmutados. En [19] se presenta un control de superficie dinámica adaptable basado en RNs para una clase de sistemas con retrasos de tiempo desconocidos y con histéresis en la entrada. En [20] se propone un controlador neuronal *wavelet* recurrente con retroalimentación de salida basado en el modelo, se realizan simulaciones y experimentos en tiempo real en una viga inteligente piezolaminada sin abrazadera de múltiples entradas y una sola salida. En [21] se propone un método de control adaptable robusto basado en RNs difusas recurrentes para robots manipuladores industriales. En [22] se presenta un control N-A robusto para el alineamiento y acoplamiento de una nave espacial. En [23] se aplican técnicas de linealización por retroalimentación y de control N-A a un giroscopio de dos grados de libertad en tiempo real; se comparó un controlador lineal, un esquema PID-PID en cascada, un controlador N-A y el esquema presentado.

[18]: Yin y col. (2018), "Neural network adaptive tracking control for a class of uncertain switched nonlinear systems"

[19]: Wang y col. (2019), "Neural network based adaptive dynamic surface control of nonaffine nonlinear systems with time delay and input hysteresis nonlinearities"

[20]: Oveisi y col. (2018), "Nonlinear observer-based recurrent wavelet neuro-controller in disturbance rejection control of flexible structures"

[21]: Yen y col. (2019), "Recurrent fuzzy wavelet neural networks based on robust adaptive sliding mode control for industrial robot manipulators"

[22]: Xia y col. (2016), "Robust adaptive backstepping neural networks control for spacecraft rendezvous and docking with uncertainties"

[23]: Moreno–Valenzuela y col. (2020), "Robust trajectory tracking control of an underactuated control moment gyroscope via neural network–based feedback linearization"

Gracias a los avances tecnológicos en el área de aprendizaje automático el interés en los vehículos autónomos se ha incrementado, y esto a su vez ha aumentado el interés en el desarrollo y validación de esquemas de control para los mismos. Para conseguir sistemas de transporte inteligentes seguros y fiables, es necesario desarrollar tecnologías de posicionamiento precisas que consideren incertidumbres, como el comportamiento de los peatones, objetos aleatorios y los tipos de ruta y su configuración [24].

[24]: Parekh y col. (2022), "A Review on Autonomous Vehicles: Progress, Methods and Challenges"

En seguida se mencionan trabajos relacionados con sistemas de navegación autónoma, visión y sistemas de aprendizaje automático aplicados a vehículos móviles. En [25] se presenta un algoritmo de clasificación de obstáculos por regiones, el cual utiliza una sola cámara monocular. El algoritmo considera como obstáculo cualquier objeto que el robot no puede atravesar con seguridad. Todo lo anterior se implementa de manera experimental. En [26] se presenta una RN profunda para aprender el modelo del robot; siendo éste, según el autor, el primer algoritmo propuesto de este tipo en tiempo real. Este enfoque muestra robustez al aplicarlo a métodos de aprendizaje del modelo con señales ruidosas, obtenidos por robots con sensores poco precisos. Además, este esquema es capaz de adaptarse a los cambios del entorno o del propio robot. En [27] se propone una solución para un problema de búsqueda de ruta usando RNs profundas para articulaciones

[25]: Lenz y col. (2012), "Low-power parallel algorithms for single image based obstacle avoidance in aerial robots"

[26]: Polydoros y col. (2015), "Real-time deep learning of robotic manipulator inverse dynamics"

[27]: Aparanji y col. (2017), "Robotic motion control using machine learning techniques"

multisegmentadas. Las capas más altas de la RN almacenan la información requerida para generar rutas óptimas. En [28] se implementa un esquema de visión experimental para la navegación de robots móviles en carreteras urbanas estrechas, los límites de la carretera se detecta mediante la arquitectura *Deep Belief Neural Network*. Los resultados muestran que la arquitectura propuesta es lo suficientemente buena para la detección del camino. En [29] se propone un sistema que fusiona sensores de visión y Lidar (radar láser) para la detección de automóviles utilizando RNs profundas. En [30] se propone un sistema de control basado en RNs profundas que mejora el seguimiento de trayectoria al utilizar datos de historiales de vuelo. Después de un entrenamiento fuera de línea con los datos de vuelo relevantes, se obtiene un modelo generalizado, este modelo se puede evaluar en tiempo real para modificar la señal de referencia dada al controlador. Además de los datos de entrenamiento no se requiere conocimiento previo del sistema. En [31] se desarrolla un drón subacuático, este sistema hace uso de hardware libre, está equipado con una cámara panorámica de 360 grados e implementa un sistema de reconocimiento de peces basado en RNs profundas. Los cálculos se ejecutan en una Raspberry Pi. El reconocimiento de peces alcanza el 87 % de precisión. En [32] se presenta un sistema de navegación autónoma especializado en la evasión de árboles. La detección de obstáculos se realiza utilizando una RNs profunda entrenada con una base de datos limitada, pero que aprovecha el aprendizaje de la arquitectura Alexnet. Además de la detección de árboles también se emplean maniobras de evasión con un único modelo neuronal. Este esquema se evalúa en simulación utilizando el simulador Gazebo y también de forma experimental en un entornos forestal. En [33] se presenta un sistema de detección y clasificación de objetos en 3D en tiempo real, utilizando como recursos un sensor Lidar y el modelo neuronal YOLOv2; el sistema se evalúa con la base de datos KITTI. En [34] se presenta un detector de objetos 3D basado en YOLOv2 ejecutable en tiempo real, el cual utiliza únicamente la información de un sensor Lidar y no requiere cámara. Este esquema se evalúa utilizando la base de datos KITTI en una tarjeta NVIDIA Titan X con un desempeño de más de 50 cuadros por segundo. En [35] se presenta LaserNet, un método detección de objetos en 3D para sistemas de conducción autónoma; este enfoque utiliza una red neuronal convolucional (RNC) y los datos de un sensor Lidar. En [36]

[28]: Khalilullah y col. (2017), "Development of robot navigation method based on single camera vision using deep learning"

[29]: Du y col. (2017), "Car detection for autonomous vehicle: LIDAR and vision fusion approach through deep learning framework"

[30]: Li y col. (2017), "Deep neural networks for improved, impromptu trajectory tracking of quadrotors"

[31]: Meng y col. (2018), "Underwater-Drone With Panoramic Camera for Automatic Fish Recognition Based on Deep Learning"

[32]: Dionisio-Ortega y col. (2018), "A deep learning approach towards autonomous flight in forest environments"

[33]: Ali y col. (2018), *YOLO3D: End-to-end real-time 3D Oriented Object Bounding Box Detection from LiDAR Point Cloud*

[34]: Simon y col. (2018), *Complex-YOLO: Real-time 3D Object Detection on Point Clouds*

[35]: Meyer y col. (2019), "LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving"

[36]: Li y col. (2020), *RTM3D: Real-time Monocular 3D Detection from Object Keypoints for Autonomous Driving*

se propone una técnica de detección de objetos monocular en 3D para sistemas de conducción autónoma; la detección de objetos se plantea como un problema de detección de puntos clave.

1.2 Objetivo general

Estudiar, aplicar y proponer técnicas de aprendizaje automático -*machine learning*- al control de movimiento de robots, tanto manipuladores como móviles.

1.3 Objetivos particulares

- Estudiar los modelos de robots candidatos a ser implementados.
- Estudiar los esquemas de control y detección de objetos que hacen uso de técnicas de aprendizaje automático viables para su aplicación en sistemas robóticos.
- Proponer esquemas de control que hacen uso de técnicas de aprendizaje automático.
- Implementar en tiempo real los esquemas de control presentados.
- Evaluar el desempeño de los esquemas de control presentados.
- Entrenar y aplicar los esquemas de detección de objetos estudiados.
- Desarrollar un sistema de navegación y detección de obstáculos haciendo uso de técnicas de aprendizaje automático.
- Implementar tanto en simulación como en tiempo real el sistema de navegación y detección de obstáculos para diferentes tareas asignadas.

1.4 Motivación

En los últimos años, los avances en el área de aprendizaje automático crecen a pasos agigantados, esto es debido a, los avances tecnológicos, el aumento en la capacidad de cálculo de las computadoras y a técnicas tales como los mecanismos de atención [37]; por ello se ha producido un enorme interés

[37]: Vaswani y col. (2017), *Attention Is All You Need*

en aplicar esta disciplina al control de movimiento de robots. Motivado por los avances y el interés en esta disciplina, este documento desarrolla, implementa y valida técnicas de aprendizaje automático al control de movimiento de robots. Además, este documento responde también a la necesidad de desarrollar y validar algoritmos de control capaces de compensar incertidumbres y perturbaciones.

1.5 Organización del documento

El presente documento de tesis está organizado de la siguiente manera:

- En el capítulo 2 se describe los modelos dinámicos utilizados para implementar diversas técnicas de aprendizaje automático.
- En el capítulo 3 se diseñan controladores N-A de una capa basados en filtrado del error para resolver el problema de seguimiento trayectoria.
- En el capítulo 4 se presentan los experimentos en tiempo real de los esquemas de control N-A diseñados.
- En el capítulo 5 se presentan algunas de las técnicas más importantes de aprendizaje automático orientadas a visión, asimismo se evalúa el desempeño del detector de objetos TinyYOLOv2 especializado en sistemas embebidos.
- En el capítulo 6 se presentan los elementos que conforman al vehículo detector de ruta, así como su implementación tanto en simulación como en tiempo real.
- En el apéndice A se describen los elementos, equipos y herramientas utilizadas para el desarrollo de esta plataforma experimental.
- En el apéndice B se presentan los códigos fuente para la implementación del vehículo detector de ruta en tiempo real.

Módelos dinámicos 2

En este capítulo se describen los modelos dinámicos del robot manipulador CICESE y el robot móvil Nexus utilizados para implementar diversas técnicas de aprendizaje automático.

2.1 Modelo dinámico del robot manipulador CICESE de 2 g.d.l.

El modelo dinámico en el espacio articular de un robot manipulador de n grados de libertad (g.d.l.) con eslabones rígidos, considerando la presencia de fricción en sus uniones, puede escribirse como [38]

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{\delta} = \boldsymbol{\tau}, \qquad (2.1)$$

[38]: Kelly y col. (2006), *Control of Robot Manipulators in Joint Space*

donde los vectores $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} \in \mathbb{R}^n$ denotan la posición, velocidad y aceleración articular respectivamente, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ es la matriz de inercia, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \in \mathbb{R}^n$ es el vector de fuerzas centrífugas y de Coriolis, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ es el vector de pares gravitacionales, $\boldsymbol{F}(\dot{\boldsymbol{q}}) \in \mathbb{R}^n$ es un vector que contiene las fuerzas de fricción viscosa y de Coulomb de cada articulación, $\boldsymbol{\delta} \in \mathbb{R}^n$ es un vector que contiene fuerzas de perturbación acotadas para cada articulación[1], y $\boldsymbol{\tau} \in \mathbb{R}^n$ es el vector de pares aplicados.

1: Las perturbaciones contenidas en este vector pueden ser de naturaleza desvaneciente o no desvaneciente.

Se utilizará el robot manipulador CICESE con dos eslabones rígidos verticales (ver figura 2.1) para realizar experimentos en tiempo real. El diagrama de este robot manipulador de 2 g.d.l. se ilustra en la figura 2.2. El robot cuenta con motores sin escobillas de alto torque sin reducción de engranajes y que opera en modo par. Las posiciones de las articulaciones se obtienen mediante encoders incrementales. El algoritmo de control se ejecuta en un período de muestreo de 0.0025 [s] en una computadora central bajo el ambiente WinMechLab [39].

[39]: Campa y col. (2004), "Windows-based real-time control of direct-drive mechanisms: platform description and experiments"

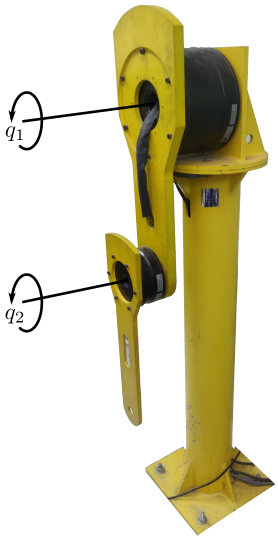

Figura 2.1: Vista lateral del robot manipulador de 2 g.d.l.

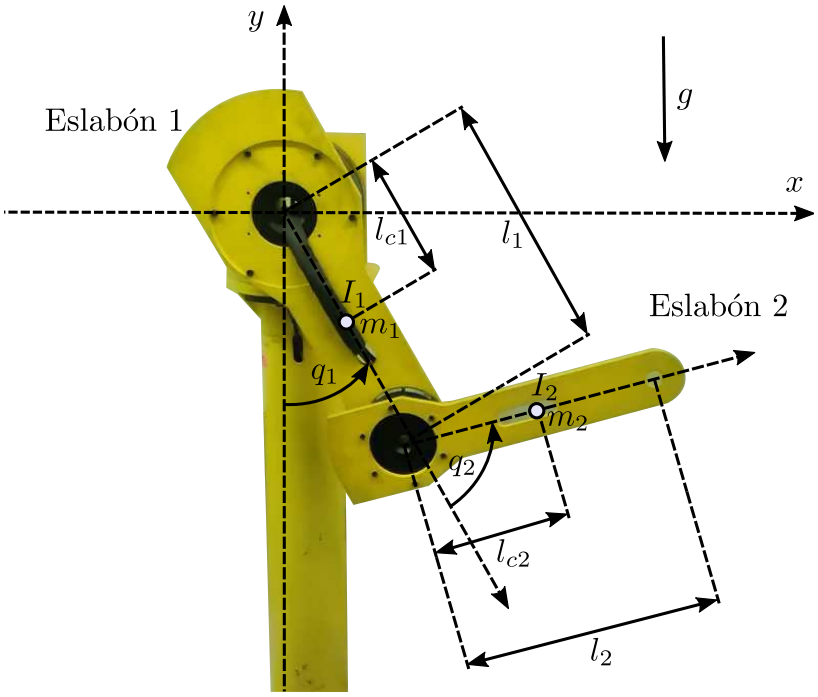

Figura 2.2: Diagrama de un robot manipulador de 2 g.d.l.

Propiedad 2.1.1 *La matriz* $\dot{\boldsymbol{M}}(\boldsymbol{q},\dot{\boldsymbol{q}})/2-\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ *es anti simétrica. Por lo tanto,*

$$\boldsymbol{y}^T\left(\dot{\boldsymbol{M}}(\boldsymbol{q},\dot{\boldsymbol{q}})/2-\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\right)\boldsymbol{y}=\boldsymbol{0},\forall\,\boldsymbol{y},\boldsymbol{q},\dot{\boldsymbol{q}}\in\mathbb{R}^n,$$

donde $\dot{\boldsymbol{M}}(\boldsymbol{q},\dot{\boldsymbol{q}})$ *es la derivada temporal de la matriz de inercias.*

2.2 Modelo dinámico del robot móvil omnidireccional Nexus

[40]: Campion y col. (1996), "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots"

El modelo dinámico de posición y orientación de un robot móvil omnidireccional con respecto al marco inercial está dado por [40]:

$$\boldsymbol{R}^T(\theta)\boldsymbol{M_R}\ddot{\boldsymbol{\xi}}+\boldsymbol{E}^T\boldsymbol{I}\ddot{\boldsymbol{\varphi}}=\boldsymbol{E}^T\boldsymbol{\tau_\varphi} \tag{2.2}$$

donde $\boldsymbol{\xi}=\begin{bmatrix}x & y & \theta\end{bmatrix}^T$ representa las variables de configuración en el espacio de trabajo, $\boldsymbol{\varphi}=\begin{bmatrix}\varphi_1 & \varphi_2 & \varphi_3 & \varphi_4\end{bmatrix}^T$ representa la posición angular de cada rueda,

$$\boldsymbol{\tau_\varphi}=\begin{bmatrix}\tau_{\varphi_1} & \tau_{\varphi_2} & \tau_{\varphi_3} & \tau_{\varphi_4}\end{bmatrix}^T$$

es el vector de pares aplicados a cada rueda,

$$\boldsymbol{M_R} = \text{Diag}\{m_1 + 4m_2, m_1 + 4m_2, 4m_2(l_1^2 + l_2^2) + J_1 + 4J_3\}$$

es la matriz de inercia del robot, $\boldsymbol{I} = \text{Diag}\{J_2, J_2, J_2, J_2\}$ es la matriz de inercia de las cuatro ruedas,

$$\boldsymbol{E} = \frac{1}{r}\begin{bmatrix} 1 & 1 & l_1 + l_2 \\ 1 & -1 & -(l_1 + l_2) \\ 1 & 1 & -(l_1 + l_2) \\ 1 & -1 & l_1 + l_2 \end{bmatrix},$$

representa la matriz jacobiana del sistema, m_1 [kg] es la masa del cuerpo, m_2 [kg] es la masa de cada rueda, J_1 [kg-m^2] es la inercia del cuerpo, J_2 [kg-m^2] es la inercia de las ruedas sobre el eje del motor, J_3 [kg-m^2] es la inercia de las ruedas perpendicular al eje del motor, l_1 [m] es la distancia del centro geométrico al eje de cada una de las ruedas sobre el eje R_1, l_2 [m] es la distancia del centro geométrico al eje de cada una de las ruedas sobre el eje R_2, r [m] es el radio de las ruedas, y

$$\boldsymbol{R}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

es la matriz de rotación para el movimiento planar.

La dinámica de los cuatro motores de CD con imanes permanentes, despreciando la inductancia de armadura L_a y considerando una fricción lineal está dada por:

$$J_m\,\ddot{\boldsymbol{\varphi}} + k_v\dot{\boldsymbol{\varphi}} + \frac{1}{r_e^2}\boldsymbol{\tau_\varphi} + \frac{K_a\,K_b}{R_a}\dot{\boldsymbol{\varphi}} = \frac{K_a}{R_a\,r_e}\boldsymbol{u}. \tag{2.3}$$

donde J_m [kg-m^2] es la inercia del rotor, k_v es la fricción viscosa, r_e [m] es la relación de transmisión, K_a [N-m/A] es una constante de motor-par, K_b [V-s/rad] es la constante de fuerza contraelectromotriz, R_a [Ω] es la resistencia de armadura y $\boldsymbol{u} \in \mathbb{R}^4$ [V] es el vector de voltajes de armadura.

"Desde un punto de vista de sistemas dinámicos, el motor de CD puede verse como un dispositivo cuya entrada es el voltaje v y su salida es el par τ que se aplica después de la caja de engranes." [41, p. 276].

Después de una serie de manipulaciones y considerando la relación $\dot{\boldsymbol{\varphi}} = \boldsymbol{E}\boldsymbol{R}(\theta)^T\dot{\boldsymbol{\xi}}$, es posible mostrar que la dinámica (2.2) considerando los actuadores (2.3) está dada por:

$$\boldsymbol{M}\ddot{\boldsymbol{\xi}} + \boldsymbol{C}(\dot{\theta})\dot{\boldsymbol{\xi}} + \boldsymbol{D}\dot{\boldsymbol{\xi}} = \boldsymbol{\tau} + \boldsymbol{\delta}, \tag{2.4}$$

donde $\boldsymbol{\tau} = \begin{bmatrix}\tau_1 & \tau_2 & \tau_3\end{bmatrix}^T$ es el vector de fuerzas generaliza-

das, $\boldsymbol{\delta} = \begin{bmatrix}\delta_1 & \delta_2 & \delta_3\end{bmatrix}^T$ es un vector que contiene fuerzas de perturbación para cada variable de configuración, $\boldsymbol{M} \in \mathbb{R}^{3\times3}$ es la matriz de inercia y $\boldsymbol{C}(\dot{\theta}) \in \mathbb{R}^{3\times3}$ es la matriz de fuerzas centrífugas y de Coriolis y $\boldsymbol{D} \in \mathbb{R}^{3\times3}$ es una matriz que incluye la fricción y parámetros de los motores; estas últimas están dadas de la siguiente manera

$$\boldsymbol{M} = \boldsymbol{M_R} + (\boldsymbol{I} + J_m r_e^2)\boldsymbol{E^T E},$$

$$\boldsymbol{C}(\dot{\theta}) = \frac{4}{r^2}(\boldsymbol{I} + J_m r_e^2)\dot{\theta}\boldsymbol{B},$$

$$\boldsymbol{D} = r_e^2\left(\frac{K_a\, K_b}{R_a} + K_v\right)\boldsymbol{E^T E}$$

donde

$$\boldsymbol{B} = \begin{bmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}.$$

Se debe observar que $\boldsymbol{M}$ es una matriz diagonal constante. La ecuación

$$\boldsymbol{\tau} = \frac{K_a\, r_e}{R_a}\boldsymbol{R}(\theta)\boldsymbol{E}^T\boldsymbol{u}$$

relaciona las fuerzas generalizadas con los voltajes de armadura de los motores.

Se utilizará el robot móvil Nexus descrito en el apéndice A.1, el cual es omnidireccional y cuenta con ruedas Mecanum; éste se ilustra en la figura 2.3. El robot cuenta con motores con escobillas, reducción de engranajes y opera en modo voltaje. Las posiciones angulares de las ruedas se obtienen mediante encoders incrementales.

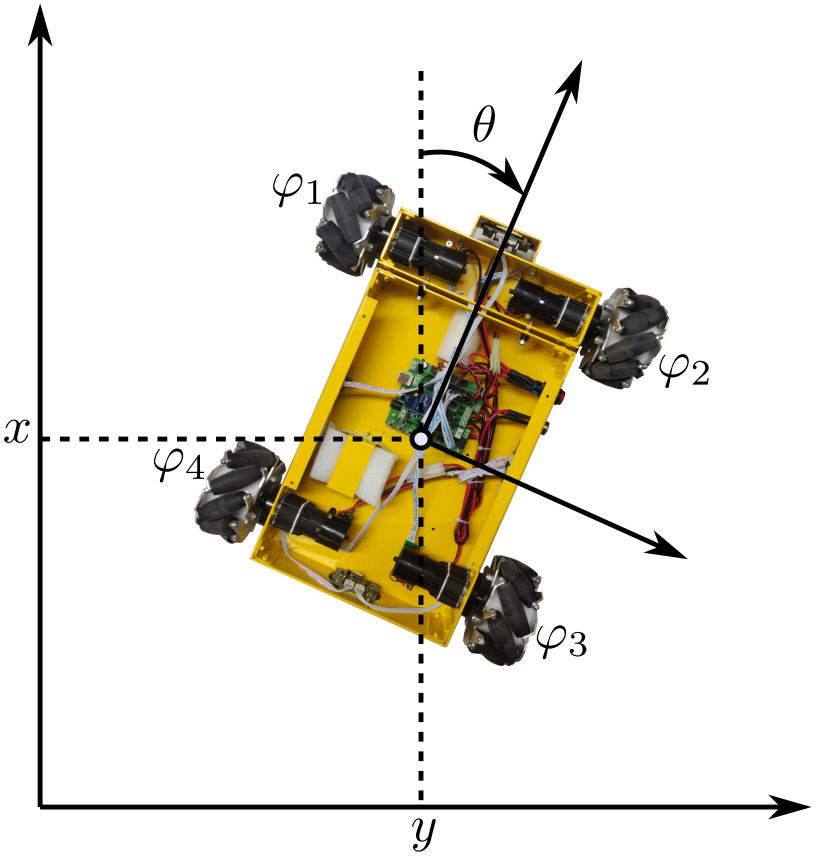

Figura 2.3: Robot móvil omnidireccional Nexus.

Control de robots Neuro-Adaptable 3

En este capítulo se diseñan controladores N-A de una capa basados en filtrado del error para resolver el problema de seguimiento de trayectoria. Las RNs se utilizan para aproximar las funciones no lineales desconocidas de la planta; posteriormente, los pesos de las RNs se estiman utilizando algoritmos de actualización de pesos en línea. Las RNs de una capa son un recurso atractivo, ya que, exigen un menor costo computacional y son más fáciles de entrenar e implementar que las RNs de dos capas. En el capítulo 4 estos controladores se implementan de manera experimental tanto al robot manipulador de 2 g.d.l. como al robot móvil omnidireccional Nexus.

3.1 Planteamiento del problema

Con la finalidad de conseguir el seguimiento de trayectoria se definen los siguientes errores

$$\widetilde{\boldsymbol{q}} = \boldsymbol{q}_d - \boldsymbol{q}, \quad \dot{\widetilde{\boldsymbol{q}}} = \dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}.$$

donde $\boldsymbol{q}_d \in \mathbb{R}^n$ y $\dot{\boldsymbol{q}}_d \in \mathbb{R}^n$ son las posiciones y las velocidades articulares deseadas. Además, también se define el filtrado del error como

$$\boldsymbol{r} = \dot{\widetilde{\boldsymbol{q}}} + \boldsymbol{\Lambda}\widetilde{\boldsymbol{q}} \tag{3.1}$$

donde $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}$ es una matriz de diseño simétrica definida positiva. Se expresa el modelo dinámico del robot manipulador de n g.d.l. (2.1) como una función de $\boldsymbol{r} \in \mathbb{R}^n$ considerando

$$\dot{\boldsymbol{q}} = -\boldsymbol{r} + \dot{\boldsymbol{q}}_d + \boldsymbol{\Lambda}\widetilde{\boldsymbol{q}}, \quad \ddot{\boldsymbol{q}} = -\dot{\boldsymbol{r}} + \ddot{\boldsymbol{q}}_d + \boldsymbol{\Lambda}\dot{\widetilde{\boldsymbol{q}}},$$

entonces,

$$\begin{aligned}\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{r}} = &-\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{r} + \boldsymbol{M}(\boldsymbol{q})(\ddot{\boldsymbol{q}}_d + \boldsymbol{\Lambda}\dot{\widetilde{\boldsymbol{q}}}) \\ &+ \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})(\dot{\boldsymbol{q}}_d + \boldsymbol{\Lambda}\widetilde{\boldsymbol{q}}) + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\delta} - \boldsymbol{\tau}. \end{aligned} \tag{3.2}$$

Ahora se define

$$\boldsymbol{u} \coloneqq \ddot{\boldsymbol{q}}_d + \boldsymbol{\Lambda}\dot{\tilde{\boldsymbol{q}}}, \quad \boldsymbol{w} \coloneqq \dot{\boldsymbol{q}}_d + \boldsymbol{\Lambda}\tilde{\boldsymbol{q}} \quad \text{y}$$

$$\boldsymbol{f} \coloneqq \boldsymbol{M}(\boldsymbol{q})\boldsymbol{u} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{w} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\delta} = \boldsymbol{W}^T\boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{\varepsilon},$$

donde $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ es el error de estimación de la RN acotado por una constante tal que $\| \boldsymbol{\varepsilon} \| < \varepsilon_N$ (el operador $\| \cdot \|$ es la norma euclidiana estándar), de modo que se produce

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{r}} = -\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{r} + \boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{\varepsilon} - \boldsymbol{\tau},$$

donde $\boldsymbol{\Phi}(\boldsymbol{x}) \in \mathbb{R}^L$ es el vector de funciones de activación de la RN, $\boldsymbol{W} \in \mathbb{R}^{L\times n}$ es la matriz de pesos ideales desconocidos de la RN y $\boldsymbol{x} \in \mathbb{R}^L$ es el vector de entradas de la RN que se eligen heurísticamente, en este caso se propone

$$\boldsymbol{x} \equiv \begin{bmatrix} \tilde{\boldsymbol{q}}^T & \dot{\tilde{\boldsymbol{q}}}^T & \boldsymbol{q}_d^T & \dot{\boldsymbol{q}}_d^T & \ddot{\boldsymbol{q}}_d^T & 1 \end{bmatrix}^T \in \mathbb{R}^L. \tag{3.3}$$

Vale la pena señalar que el elemento unitario en la ecuación (3.3) es el polarizador de la RN [42].

[42]: Haykin (1999), *Neural Networks: A Comprehensive Foundation*

3.2 Control Neuro-Adaptable de una capa (CNA)

[5]: Lewis y col. (1998), *Neural Network Control of Robot Manipulators and Nonlinear Systems*

En [5] se propuso la siguiente ley de control N-A de una capa (ver figura 3.1):

$$\boldsymbol{\tau} = \widehat{\boldsymbol{W}}^T\boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{K}_v\boldsymbol{r} + \boldsymbol{v}, \tag{3.4}$$

$$\dot{\widehat{\boldsymbol{W}}} = \boldsymbol{\Gamma}\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{r}^T, \tag{3.5}$$

con

$$\boldsymbol{v} = \begin{cases} \varepsilon_N \dfrac{\boldsymbol{r}}{\| \boldsymbol{r} \|}, & \text{if } \boldsymbol{r} \neq \boldsymbol{0}, \\ \boldsymbol{0}, & \text{if } \boldsymbol{r} = \boldsymbol{0}, \end{cases} \tag{3.6}$$

donde $\boldsymbol{\Gamma} \in \mathbb{R}^{L\times L}$ es una matriz de ganancias ajustables definida positiva que generalmente se selecciona como diagonal, $\boldsymbol{v} \in \mathbb{R}^n$ es una señal auxiliar para proveer robustez, y $\boldsymbol{K}_v \in \mathbb{R}^{n\times n}$ es una matriz simétrica definida positiva ajustable denominada ganancia de velocidad. Definiendo $\boldsymbol{\Lambda} = \boldsymbol{K}_v^{-1}\boldsymbol{K}_p$, donde $\boldsymbol{K}_p \in \mathbb{R}^{n\times n}$ es una matriz simétrica definida positiva ajustable denominada ganancia de posición, esto lleva a que el termino $\boldsymbol{K}_v\boldsymbol{r}$ sea meramente un PD.

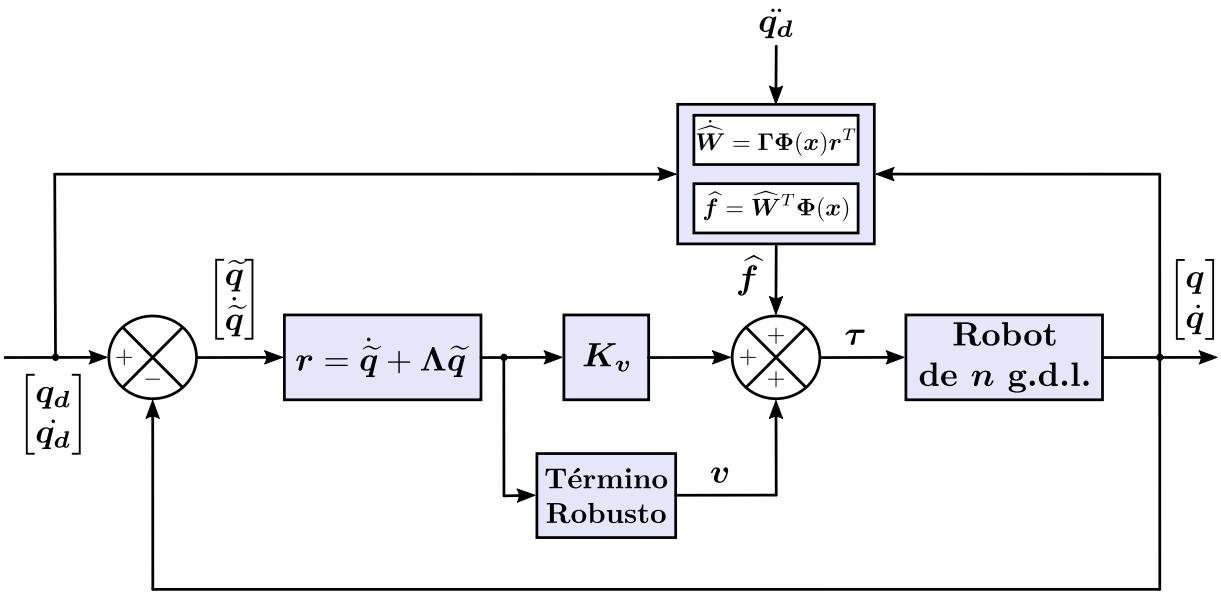

Figura 3.1: Diagrama de bloques del esquema CNA.

Ahora es necesario mostrar cómo la ley adaptable (3.5) ajusta en línea los pesos estimados de la RN $\widehat{W}$ para garantizar el seguimiento de trayectoria; este algoritmo modifica los pesos estimados $\widehat{W}$ para que se aproximen a los pesos ideales W, los cuales son desconocidos.

El estimado de la función no lineal $f \in \mathbb{R}^n$ está dado por

$$\widehat{f} = \widehat{W}^T\Phi(x),$$

y suponiendo que la matriz de pesos desconocidos W es constante, y además definiendo

$$\widetilde{f} = f - \widehat{f} = W^T\Phi(x) + \varepsilon - \widehat{W}^T\Phi(x) = \widetilde{W}^T\Phi(x) + \varepsilon,$$

donde el error de estimación de pesos se define como $\widetilde{W} = W - \widehat{W}$. Por lo tanto, usando la ley de control propuesta (3.4-3.6) resulta la siguiente dinámica en lazo cerrado

$$M(q)\dot{r} = -C(q,\dot{q})r + \widetilde{W}^T\Phi(x) + \varepsilon - K_v r - \varepsilon_N \frac{r}{\| r \|}. \quad (3.7)$$

Eligiendo la función candidata de Lyapunov como

$$V = \frac{1}{2}r^T M(q)r + \tilde{q}^T K_p \tilde{q} + \frac{1}{2}\mathrm{tr}(\widetilde{W}^T\Gamma^{-1}\widetilde{W}), \quad (3.8)$$

donde tr(·) es el operador traza, entonces la derivada tempo-

ral de V está dada por

$$\dot{V} = \boldsymbol{r}^T \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{r}} + \frac{1}{2}\boldsymbol{r}^T \dot{\boldsymbol{M}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{r} + 2\widetilde{\boldsymbol{q}}^T \boldsymbol{K}_p \dot{\widetilde{\boldsymbol{q}}} + \mathrm{tr}(\dot{\widetilde{\boldsymbol{W}}}^T \boldsymbol{\Gamma}^{-1} \widetilde{\boldsymbol{W}}).$$

Sustituyendo (3.7) en la expresión anterior, se produce

$$\begin{aligned}
\dot{V} &= \boldsymbol{r}^T \left(-\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{r} + \widetilde{\boldsymbol{W}}^T \boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{\varepsilon} - \boldsymbol{K}_v \boldsymbol{r} - \varepsilon_N \frac{\boldsymbol{r}}{\| \boldsymbol{r} \|} \right) \\
&\quad + \frac{1}{2}\boldsymbol{r}^T \dot{\boldsymbol{M}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\boldsymbol{r} + 2\widetilde{\boldsymbol{q}}^T \boldsymbol{K}_p \dot{\widetilde{\boldsymbol{q}}} + \mathrm{tr}(\dot{\widetilde{\boldsymbol{W}}}^T \boldsymbol{\Gamma}^{-1} \widetilde{\boldsymbol{W}}), \\
\dot{V} &= \boldsymbol{r}^T \left(\frac{1}{2}\dot{\boldsymbol{M}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \right) \boldsymbol{r} + \boldsymbol{r}^T \widetilde{\boldsymbol{W}}^T \boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{r}^T \boldsymbol{K}_v \boldsymbol{r} \\
&\quad + \boldsymbol{r}^T (\boldsymbol{\varepsilon} - \varepsilon_N \frac{\boldsymbol{r}}{\| \boldsymbol{r} \|}) + 2\widetilde{\boldsymbol{q}}^T \boldsymbol{K}_p \dot{\widetilde{\boldsymbol{q}}} + \mathrm{tr}(\dot{\widetilde{\boldsymbol{W}}}^T \boldsymbol{\Gamma}^{-1} \widetilde{\boldsymbol{W}}), \quad (3.9)
\end{aligned}$$

entonces se utiliza la Propiedad 2.1.1 y

$$\boldsymbol{a}^T \boldsymbol{b} = \mathrm{tr}(\boldsymbol{b}\boldsymbol{a}^T), \quad \boldsymbol{r}^T \widetilde{\boldsymbol{W}}^T \boldsymbol{\Phi}(\boldsymbol{x}) = \mathrm{tr}(\widetilde{\boldsymbol{W}}^T \boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{r}^T),$$

donde $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, así (3.9) resulta en

$$\begin{aligned}
\dot{V} = -\boldsymbol{r}^T \boldsymbol{K}_v \boldsymbol{r} + 2\widetilde{\boldsymbol{q}}^T \boldsymbol{K}_p \dot{\widetilde{\boldsymbol{q}}} + \\
\mathrm{tr}\left(\widetilde{\boldsymbol{W}}^T \left[\boldsymbol{\Gamma}^{-1} \dot{\widetilde{\boldsymbol{W}}} + \boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{r}^T \right] \right) + \boldsymbol{r}^T (\boldsymbol{\varepsilon} - \varepsilon_N \frac{\boldsymbol{r}}{\| \boldsymbol{r} \|}).
\end{aligned}$$

Ahora, de (3.5) se obtiene $\dot{\widetilde{\boldsymbol{W}}} = -\dot{\widehat{\boldsymbol{W}}} = -\boldsymbol{\Gamma}\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{r}^T$, luego la derivada temporal de la función de Lyapunov resulta

$$\begin{aligned}
\dot{V} &= -\boldsymbol{r}^T \boldsymbol{K}_v \boldsymbol{r} + 2\widetilde{\boldsymbol{q}}^T \boldsymbol{K}_p \dot{\widetilde{\boldsymbol{q}}} + \boldsymbol{r}^T (\boldsymbol{\varepsilon} - \varepsilon_N \frac{\boldsymbol{r}}{\| \boldsymbol{r} \|}), \\
\dot{V} &= -\left[\dot{\widetilde{\boldsymbol{q}}}^T + \widetilde{\boldsymbol{q}}^T \boldsymbol{\Lambda}^T \right] \boldsymbol{K}_v \left[\dot{\widetilde{\boldsymbol{q}}} + \boldsymbol{\Lambda}\widetilde{\boldsymbol{q}} \right] \\
&\quad + 2\widetilde{\boldsymbol{q}}^T \boldsymbol{K}_p \dot{\widetilde{\boldsymbol{q}}} + \boldsymbol{r}^T (\boldsymbol{\varepsilon} - \varepsilon_N \frac{\boldsymbol{r}}{\| \boldsymbol{r} \|}), \\
\dot{V} &= -\dot{\widetilde{\boldsymbol{q}}}^T \boldsymbol{K}_v \dot{\widetilde{\boldsymbol{q}}} - \widetilde{\boldsymbol{q}}^T \boldsymbol{\Lambda}^T \boldsymbol{K}_v \boldsymbol{\Lambda}\widetilde{\boldsymbol{q}} - \dot{\widetilde{\boldsymbol{q}}}^T \boldsymbol{K}_v \boldsymbol{\Lambda}\widetilde{\boldsymbol{q}} - \widetilde{\boldsymbol{q}}^T \boldsymbol{\Lambda}^T \boldsymbol{K}_v \dot{\widetilde{\boldsymbol{q}}} \\
&\quad + 2\widetilde{\boldsymbol{q}}^T \boldsymbol{K}_p \dot{\widetilde{\boldsymbol{q}}} + \boldsymbol{r}^T (\boldsymbol{\varepsilon} - \varepsilon_N \frac{\boldsymbol{r}}{\| \boldsymbol{r} \|}), \\
\dot{V} &= -\dot{\widetilde{\boldsymbol{q}}}^T \boldsymbol{K}_v \dot{\widetilde{\boldsymbol{q}}} - \widetilde{\boldsymbol{q}}^T \boldsymbol{\Lambda}^T \boldsymbol{K}_p \widetilde{\boldsymbol{q}} + \boldsymbol{r}^T (\boldsymbol{\varepsilon} - \varepsilon_N \frac{\boldsymbol{r}}{\| \boldsymbol{r} \|}), \\
\dot{V} &= -\begin{bmatrix} \widetilde{\boldsymbol{q}} \\ \dot{\widetilde{\boldsymbol{q}}} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\Lambda}^T \boldsymbol{K}_p & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{K}_v \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{q}} \\ \dot{\widetilde{\boldsymbol{q}}} \end{bmatrix} - \boldsymbol{r}^T (\varepsilon_N \frac{\boldsymbol{r}}{\| \boldsymbol{r} \|} - \boldsymbol{\varepsilon}), \quad (3.10) \\
\dot{V} &\leq -\begin{bmatrix} \widetilde{\boldsymbol{q}} \\ \dot{\widetilde{\boldsymbol{q}}} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{K}_p \boldsymbol{K}_v^{-1} \boldsymbol{K}_p & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{K}_v \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{q}} \\ \dot{\widetilde{\boldsymbol{q}}} \end{bmatrix} \\
&\quad -(\varepsilon_N - \| \boldsymbol{\varepsilon} \|) \| \boldsymbol{r} \| \leq 0. \quad (3.11)
\end{aligned}$$

Recordando que $\| \boldsymbol{\varepsilon} \| < \varepsilon_N$, $\boldsymbol{\Lambda}$ es una matriz no singular, $\boldsymbol{K_v}$ es una matriz simétrica definida positiva, entonces $\boldsymbol{\Lambda}^T \boldsymbol{K_p} = \boldsymbol{K_p}\, \boldsymbol{K_v}^{-1} \boldsymbol{K_p}$ será también una matriz simétrica definida positiva. Lo anterior conlleva a que $\dot{V}$ expresado en (3.10) sea una función semidefinida negativa. Dado que, además, la función candidata de Lyapunov V en (3.8) es definida positiva, radialmente desacotada, y decreciente, esto garantiza que el origen $\widetilde{\boldsymbol{q}}, \dot{\widetilde{\boldsymbol{q}}} = \boldsymbol{0}, \widetilde{\boldsymbol{W}} = \boldsymbol{0}$ es uniformemente estable y $\boldsymbol{r}, \widetilde{\boldsymbol{q}}, \dot{\widetilde{\boldsymbol{q}}}$ y $\widetilde{\boldsymbol{W}}$ están acotados para cualquier condición inicial. Como $\| \boldsymbol{\varepsilon} \| < \varepsilon_N$, la expresión (3.11) se puede reducir a

$$\dot{V} \leq -\widetilde{\boldsymbol{q}}^T \boldsymbol{K_p} \boldsymbol{K_v}^{-1} \boldsymbol{K_p} \widetilde{\boldsymbol{q}} - \dot{\widetilde{\boldsymbol{q}}}^T \boldsymbol{K_v} \dot{\widetilde{\boldsymbol{q}}}, \tag{3.12}$$

entonces integrando ambos lados de (3.12), se puede concluir que

$$\begin{aligned} \int_0^t \widetilde{\boldsymbol{q}}(\sigma)^T \widetilde{\boldsymbol{q}}(\sigma)\, \mathrm{d}\sigma &\leq \frac{V(0, \boldsymbol{r}(0), \widetilde{\boldsymbol{q}}(0), \widetilde{\boldsymbol{W}}(0))}{\lambda_{\min}\{\boldsymbol{K_p}\, \boldsymbol{K_v}^{-1} \boldsymbol{K_p}\}}, \text{ y} \\ \int_0^t \dot{\widetilde{\boldsymbol{q}}}(\sigma)^T \dot{\widetilde{\boldsymbol{q}}}(\sigma)\, \mathrm{d}\sigma &\leq \frac{V(0, \boldsymbol{r}(0), \widetilde{\boldsymbol{q}}(0), \widetilde{\boldsymbol{W}}(0))}{\lambda_{\min}\{\boldsymbol{K_v}\}}, \end{aligned}$$

lo que significa que $\widetilde{\boldsymbol{q}}, \dot{\widetilde{\boldsymbol{q}}} \in \mathrm{L}_2^n$; finalmente, usando el lema de Barbalat [43], se concluye que

[43]: Khalil (2002), *Nonlinear Systems*

$$\lim_{t \to \infty} \widetilde{\boldsymbol{q}}(t) = \boldsymbol{0}, \quad \lim_{t \to \infty} \dot{\widetilde{\boldsymbol{q}}}(t) = \boldsymbol{0} \quad \text{and} \quad \lim_{t \to \infty} \boldsymbol{r}(t) = \boldsymbol{0}.$$

El análisis de estabilidad anterior difiere al mostrado en [5] ya que se utiliza una función candidata de Lyapunov diferente, el cual permite hacer conclusiones directamente sobre $\widetilde{\boldsymbol{q}}$ y $\dot{\widetilde{\boldsymbol{q}}}$; además, los términos $\boldsymbol{\varepsilon}$ y $\boldsymbol{\delta}$ no se desprecian aquí, y se usa un término robusto $\boldsymbol{v}$ diferente.

[5]: Lewis y col. (1998), *Neural Network Control of Robot Manipulators and Nonlinear Systems*

3.3 Control Neuro-Adaptable Modificado de una capa (CN-AM)

Se propusieron algunas modificaciones al controlador CNA, la primera consiste en aumentar la pendiente de la tangente hiperbólica, y la segunda consiste en modificar el algoritmo de actualización de pesos de la red; se propone sustituir el filtrado del error $\boldsymbol{r}$ en la ecuación (3.5) por la potencia signada $\boldsymbol{r}$.

Entonces modificando la ley de control (3.4-3.6) resulta

$$\boldsymbol{\tau} = \widehat{\boldsymbol{W}}^T \boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{K}_v \boldsymbol{r} + \boldsymbol{v}, \tag{3.13}$$

$$\dot{\widehat{\boldsymbol{W}}} = \boldsymbol{\Gamma} \boldsymbol{\Phi}(\boldsymbol{x}) \lceil \boldsymbol{r}^T \rfloor^{\alpha}, \tag{3.14}$$

con

$$\boldsymbol{v} = \begin{cases} \varepsilon_N \dfrac{\boldsymbol{r}}{\| \boldsymbol{r} \|}, & \text{si } \boldsymbol{r} \neq \boldsymbol{0}, \\ \boldsymbol{0}, & \text{si } \boldsymbol{r} = \boldsymbol{0}, \end{cases} \tag{3.15}$$

donde

$$\lceil \boldsymbol{a} \rfloor^{\alpha} = [|a_1|^{\alpha} \text{sign}(a_1), \cdots, |a_n|^{\alpha} \text{sign}(a_n)]^T \tag{3.16}$$

es la función potencia signada para cualquier $\boldsymbol{a} \in \mathbb{R}^n$ y cualquier $\alpha \in (0, 1)$.

Esta propuesta tiene el inconveniente de que no se ha encontrado una función candidata de Lyapunov V con la cual trabajar el análisis de estabilidad.

3.4 Control PD con compensación N-A usando la función Potencia Signada (CPD-CNA)

[5]: Lewis y col. (1998), *Neural Network Control of Robot Manipulators and Nonlinear Systems*

Inspirado en [5], en esta sección se presenta un nuevo controlador N-A utilizando la función de potencia signada.

Reescribiendo el sistema (3.2) de la siguiente manera

$$\dot{\boldsymbol{r}} = \boldsymbol{M}(\boldsymbol{q})^{-1} [-\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \boldsymbol{r} + \boldsymbol{M}(\boldsymbol{q}) \boldsymbol{u} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \boldsymbol{w} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\delta}] - \boldsymbol{M}(\boldsymbol{q})^{-1} \boldsymbol{\tau},$$

entonces se puede definir

$$\boldsymbol{f} \coloneqq \boldsymbol{M}(\boldsymbol{q})^{-1} [-\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \boldsymbol{r} + \boldsymbol{M}(\boldsymbol{q}) \boldsymbol{u} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \boldsymbol{w} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\delta}] = \boldsymbol{W}^T \boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{\varepsilon}, \tag{3.17}$$

donde $\boldsymbol{\varepsilon}$ está acotado por una constante tal que $\| \boldsymbol{\varepsilon} \| < \varepsilon_N$, de modo que se produce

$$\dot{\boldsymbol{r}} = \boldsymbol{W}^T \boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{\varepsilon} - \boldsymbol{M}(\boldsymbol{q})^{-1} \boldsymbol{\tau}.$$

Motivado por la ley de control (3.4-3.6) se propone la siguiente ley de control N-A (ver figura 3.2)

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\left[\widehat{\boldsymbol{W}}^T\boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{K}_v\boldsymbol{r} + \boldsymbol{v}\right], \tag{3.18}$$

$$\dot{\widehat{\boldsymbol{W}}} = \boldsymbol{\Gamma}\boldsymbol{\Phi}(\boldsymbol{x})\lceil \boldsymbol{r}^T \rfloor^{\alpha}, \tag{3.19}$$

con

$$\boldsymbol{v} = \begin{cases} \varepsilon_N \dfrac{\lceil \boldsymbol{r} \rfloor^{\alpha}}{\| \lceil \boldsymbol{r} \rfloor^{\alpha} \|}, & \text{si } \boldsymbol{r} \neq \boldsymbol{0}, \\ \boldsymbol{0}, & \text{si } \boldsymbol{r} = \boldsymbol{0}. \end{cases} \tag{3.20}$$

Por lo tanto, usando la ley de control propuesta (3.18-3.20) resulta la siguiente dinámica en lazo cerrado

$$\dot{\boldsymbol{r}} = \widetilde{\boldsymbol{W}}^T\boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{\varepsilon} - \boldsymbol{K}_v\boldsymbol{r} - \varepsilon_N \frac{\lceil \boldsymbol{r} \rfloor^{\alpha}}{\| \lceil \boldsymbol{r} \rfloor^{\alpha} \|}. \tag{3.21}$$

Eligiendo la función candidata de Lyapunov como

$$V = \frac{1}{\alpha+1}\sum_{i=1}^{n} |r_i|^{\alpha+1} + \frac{1}{2}\mathrm{tr}(\widetilde{\boldsymbol{W}}^T\boldsymbol{\Gamma}^{-1}\widetilde{\boldsymbol{W}}), \tag{3.22}$$

1: La derivada temporal de $|x|^n$ es $n|x|^{n-1}\mathrm{sign}(x)\dot{x}$

entonces la derivada temporal de V está dado por [1]

$$\dot{V} = \lceil \boldsymbol{r}^T \rfloor^{\alpha}\dot{\boldsymbol{r}} + \mathrm{tr}(\widetilde{\boldsymbol{W}}^T\boldsymbol{\Gamma}^{-1}\dot{\widetilde{\boldsymbol{W}}}).$$

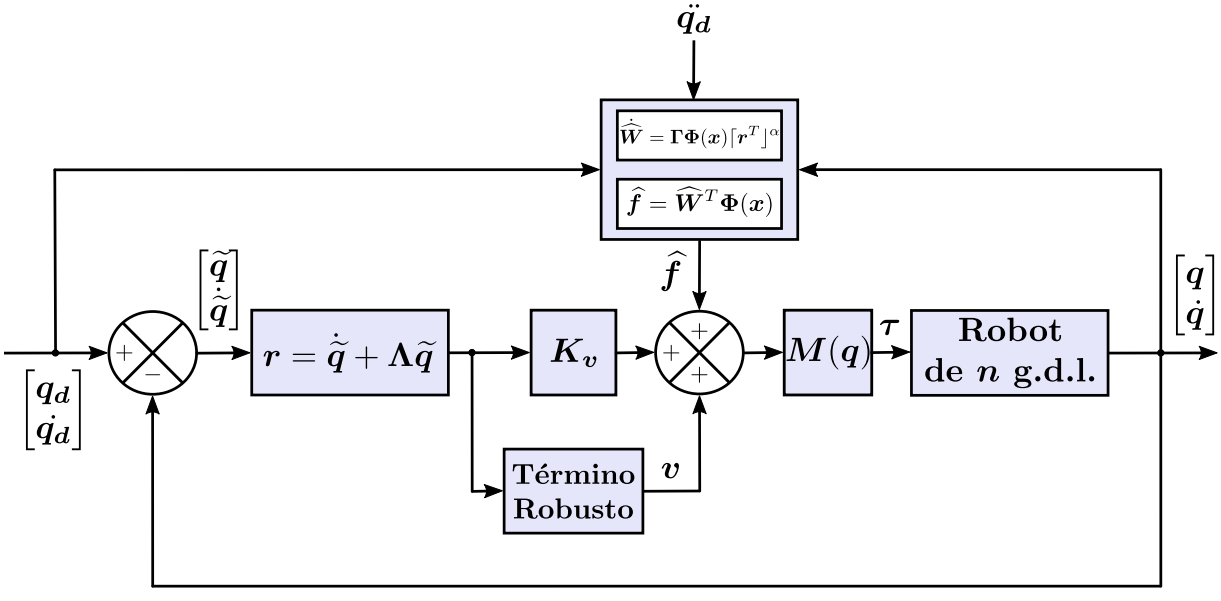

Figura 3.2: Diagrama de bloques del esquema CPD-CNA.

Sustituyendo (3.21) en la expresión anterior se produce

$$\begin{aligned}\dot{V} &= \lceil \boldsymbol{r}^T \rfloor^{\alpha} \left(\widetilde{\boldsymbol{W}}^T \boldsymbol{\Phi}(\boldsymbol{x}) + \boldsymbol{\varepsilon} - \boldsymbol{K}_v \boldsymbol{r} - \varepsilon_N \frac{\lceil \boldsymbol{r} \rfloor^{\alpha}}{\| \lceil \boldsymbol{r} \rfloor^{\alpha} \|} \right) \\ &\quad + \mathrm{tr}(\widetilde{\boldsymbol{W}}^T \boldsymbol{\Gamma}^{-1} \dot{\widetilde{\boldsymbol{W}}}), \\ \dot{V} &= \lceil \boldsymbol{r}^T \rfloor^{\alpha} \widetilde{\boldsymbol{W}}^T \boldsymbol{\Phi}(\boldsymbol{x}) + \lceil \boldsymbol{r}^T \rfloor^{\alpha} (\boldsymbol{\varepsilon} - \varepsilon_N \frac{\lceil \boldsymbol{r} \rfloor^{\alpha}}{\| \lceil \boldsymbol{r} \rfloor^{\alpha} \|}) \\ &\quad - \lceil \boldsymbol{r}^T \rfloor^{\alpha} \boldsymbol{K}_v \boldsymbol{r} + \mathrm{tr}(\widetilde{\boldsymbol{W}}^T \boldsymbol{\Gamma}^{-1} \dot{\widetilde{\boldsymbol{W}}}), \quad (3.23)\end{aligned}$$

entonces se utiliza

$$\boldsymbol{a}^T \boldsymbol{b} = \mathrm{tr}(\boldsymbol{b}\boldsymbol{a}^T), \quad \lceil \boldsymbol{r}^T \rfloor^{\alpha} \widetilde{\boldsymbol{W}}^T \boldsymbol{\Phi}(\boldsymbol{x}) = \mathrm{tr}(\widetilde{\boldsymbol{W}}^T \boldsymbol{\Phi}(\boldsymbol{x}) \lceil \boldsymbol{r}^T \rfloor^{\alpha}),$$

así (3.23) resulta en

$$\begin{aligned}\dot{V} = -\lceil \boldsymbol{r}^T \rfloor^{\alpha} \boldsymbol{K}_v \boldsymbol{r} + \mathrm{tr} \left(\widetilde{\boldsymbol{W}}^T \left[\boldsymbol{\Gamma}^{-1} \dot{\widetilde{\boldsymbol{W}}} + \boldsymbol{\Phi}(\boldsymbol{x}) \lceil \boldsymbol{r}^T \rfloor^{\alpha} \right] \right) \\ + \lceil \boldsymbol{r}^T \rfloor^{\alpha} (\boldsymbol{\varepsilon} - \varepsilon_N \frac{\lceil \boldsymbol{r} \rfloor^{\alpha}}{\| \lceil \boldsymbol{r} \rfloor^{\alpha} \|}).\end{aligned}$$

Ahora, de (3.19) se obtiene $\dot{\widetilde{\boldsymbol{W}}} = -\dot{\widehat{\boldsymbol{W}}} = -\boldsymbol{\Gamma} \boldsymbol{\Phi}(\boldsymbol{x}) \lceil \boldsymbol{r}^T \rfloor^{\alpha}$, luego la derivada temporal de la función de Lyapunov resulta [2]

2: $|r|^{\alpha} \mathrm{sign}(r) r = |r|^{\alpha+1}$

$$\begin{aligned}\dot{V} &= -\lceil \boldsymbol{r}^T \rfloor^{\alpha} \boldsymbol{K}_v \boldsymbol{r} - \lceil \boldsymbol{r}^T \rfloor^{\alpha} (\varepsilon_N \frac{\lceil \boldsymbol{r} \rfloor^{\alpha}}{\| \lceil \boldsymbol{r} \rfloor^{\alpha} \|} - \boldsymbol{\varepsilon}), \quad (3.24) \\ \dot{V} &\leq -\lambda_{\min}\{\boldsymbol{K}_v\} \sum_{i=1}^{n} |r_i|^{\alpha+1} \\ &\quad - (\varepsilon_N - \| \boldsymbol{\varepsilon} \|) \| \lceil \boldsymbol{r} \rfloor^{\alpha} \| \leq 0. \quad (3.25)\end{aligned}$$

Recordando que $\| \boldsymbol{\varepsilon} \| < \varepsilon_N$ y $\boldsymbol{K}_v$ es una matriz simétrica definida positiva, esto conlleva a que $\dot{V}$ expresado en (3.24) sea una función semidefinida negativa. Dado que, además, la función candidata de Lyapunov V en (3.22) es definida positiva, radialmente desacotada, y decreciente, esto garantiza que el origen $\boldsymbol{r} = \boldsymbol{0}, \widetilde{\boldsymbol{W}} = \boldsymbol{0}$ es uniformemente estable y $\boldsymbol{r}$ y $\widetilde{\boldsymbol{W}}$ están acotados para cualquier condición inicial. Como $\| \boldsymbol{\varepsilon} \| < \varepsilon_N$, la expresión (3.25) se puede reducir a

$$\dot{V} \leq -\lambda_{\min}\{\boldsymbol{K}_v\} \sum_{i=1}^{n} |r_i|^{\alpha+1}, \quad (3.26)$$

entonces integrando ambos lados de (3.26), se puede concluir

que

$$\int_0^t \sum_{i=1}^{n} |r_i(\sigma)|^{\alpha+1} \, \mathrm{d}\sigma \quad \leq \quad \frac{V(0, \boldsymbol{r}(0), \widetilde{\boldsymbol{W}}(0))}{\lambda_{\min}\{\boldsymbol{K}_v\}},$$

lo que significa que $\boldsymbol{r}(t) \in \mathrm{L}_1^n$, y junto con $\dot{\boldsymbol{r}}(t) \in \mathrm{L}_\infty^n$, y $\boldsymbol{r}(t) \in \mathrm{L}_\infty$, se puede aplicar el lema de Barbalat [43] para concluir que $\boldsymbol{r}$ tiende a cero asintóticamente, y debido a la ecuación (3.1) $\widetilde{\boldsymbol{q}}$ y $\dot{\widetilde{\boldsymbol{q}}}$ tienden a cero asintóticamente [44]; de hecho $\| \widetilde{\boldsymbol{q}} \| \leq \| \boldsymbol{r} \| / \lambda_{min}(\Lambda)$ y $\| \dot{\widetilde{\boldsymbol{q}}} \| \leq \| \boldsymbol{r} \|$ [5], [45].

[43]: Khalil (2002), *Nonlinear Systems*

[44]: Slotine y col. (1991), *Applied Nonlinear Control*

[5]: Lewis y col. (1998), *Neural Network Control of Robot Manipulators and Nonlinear Systems*

[45]: Slotine y col. (1987), "On the adaptive control of robot manipulators"

Nótese que con el fin de incluir la función potencia signada en la ley de actualización de pesos para el esquema propuesto CPD-CNA, y así aumentar la velocidad de convergencia de los pesos, la nueva estructura de $\boldsymbol{v}$ (3.20) se selecciona de acuerdo con su respectivo análisis de estabilidad. Se puede observar que el esquema propuesto (3.18-3.20) difiere de (3.4-3.6) por el uso de la matriz de inercias $\boldsymbol{M}(\boldsymbol{q})$ y la función potencia signada (3.16). Las propiedades de $\boldsymbol{M}(\boldsymbol{q})$ (pero no su conocimiento preciso) sólo se requieren para el análisis de estabilidad porque $\boldsymbol{M}(\boldsymbol{q})$ se estima implícitamente por la función $\widehat{\boldsymbol{f}} = \widehat{\boldsymbol{W}}^T \boldsymbol{\Phi}(\boldsymbol{x})$, donde $\boldsymbol{f}$ está dada por (3.17). Las posibles imprecisiones de $\boldsymbol{M}(\boldsymbol{q})$ en la ley de control (3.18) son implícitamente compensados por la ley adaptable (3.19).

3.5 Conclusiones y observaciones

Los esquemas N-As presentados funcionan en una gran cantidad de sistemas sin necesidad de un modelado extenso o análisis previo para encontrar un vector regresor. A diferencia de los controladores adaptables convencionales, estos esquemas no condicionan el cumplimiento de requisitos como la linealidad en los parámetros de la planta o el principio de equivalencia cierta.

Se debe destacar que la función candidata de Lyapunov propuesta para el esquema CNA permite hacer conclusiones directas sobre $\widetilde{\boldsymbol{q}}$ y $\dot{\widetilde{\boldsymbol{q}}}$. Además, los controladores N-As propuestos utilizan la función de potencia signada y cuentan con actualización de pesos en línea.

Implementación de controladores Neuro-Adaptables

4

En este capítulo se presentan los experimentos en tiempo real de los esquemas de control neuro-adaptables diseñados en el capítulo 3. Estos controladores se implementan tanto en el robot manipulador de 2 g.d.l. como en el robot móvil omnidireccional Nexus; en el caso del robot manipulador de 2 g.d.l. se presentan también resultados con perturbaciones.

4.1 Resultados experimentales para el robot manipulador de 2 g.d.l.

Para todos los experimentos realizados en el robot manipulador de 2 g.d.l. se han seleccionado las siguientes trayectorias deseadas [41]:

$$\boldsymbol{q}_d = \begin{bmatrix} b_1(1-e^{-2\,t^3}) + c_1(1-e^{-2\,t^3})\sin(\omega_1\,t) \\ b_2(1-e^{-2\,t^3}) + c_2(1-e^{-2\,t^3})\sin(\omega_2\,t) \end{bmatrix} \qquad [\text{rad}],$$

donde $b_1 = \pi/4$, $c_1 = \pi/9$ y $\omega_1 = 4$ son parámetros para la posición deseada del primer eslabón y $b_2 = \pi/3$, $c_2 = \pi/6$, y $\omega_2 = 3$ son parámetros para la posición deseada del segundo eslabón. Las condiciones iniciales $\boldsymbol{q_d}(0) = [0,0]^T$ y $\widehat{\boldsymbol{W}}(0) = \mathbf{0}_{11\times 2}$ se tomaron para todos los controladores.

El procedimiento para sintonizar los controladores se describe a continuación. En primer lugar, se eligen valores pequeños de $\boldsymbol{K_v}$ y $\boldsymbol{K_p}$ para evitar la saturación de los actuadores, pero deben ser suficientemente grandes para superar la fricción estática; todas las ganancias restantes se inicializan en cero. En segundo lugar, $\boldsymbol{\Gamma}$ y ε_N se ajustan a partir de valores muy pequeños y se aumentan hasta el punto en que no se observa una mejora significativa en el rendimiento del controlador. En el caso de los esquemas CN-AM y CPD-CNA, también se debe ajustar α partiendo de uno y decreciendo en decimales sin llegar a cero. Por último, $\boldsymbol{K_v}$ y $\boldsymbol{K_p}$ se ajustan teniendo cuidado de no exceder el umbral crítico de $\boldsymbol{K_v}$ (donde el rendimiento del controlador comienza a empeorar en lugar de mejorar). Las ganancias resultantes para cada controlador

[41]: Kelly y col. (2003), *Control de movimiento de robots manipuladores*

están dadas por:

$$\begin{gathered}\boldsymbol{K_v} = \text{diag}\,\{30, 15\}\ \text{[N-m s/rad]},\\ \boldsymbol{K_p} = \text{diag}\{800, 450\}\ \text{[N-m/rad]},\\ \boldsymbol{\Lambda} = \text{diag}\{26.66, 30\}\ \text{[1/s]},\\ \varepsilon_N = 0.001\ \text{[N-m]}\ \text{y}\ \boldsymbol{\Gamma} = 50\ \boldsymbol{I}_{L\times L}\end{gathered}$$

para el esquema CNA,

$$\begin{gathered}\boldsymbol{K_v} = \text{diag}\,\{30, 15\}\ \text{[N-m s/rad]},\\ \boldsymbol{K_p} = \text{diag}\{800, 450\}\ \text{[N-m/rad]},\\ \boldsymbol{\Lambda} = \text{diag}\{26.66, 30\}\ \text{[1/s]},\\ \varepsilon_N = 0.001\ \text{[N-m]},\ \boldsymbol{\Gamma} = 30\ \boldsymbol{I}_{L\times L}\ \text{y}\ \alpha = 0.7\end{gathered}$$

para el esquema CN-AM, y

$$\begin{gathered}\boldsymbol{K_v} = \text{diag}\,\{30, 35\}\ \text{[1/rad s]},\\ \boldsymbol{K_p} = \text{diag}\{1200, 6000\}\ \text{[1/rad s}^2\text{]},\\ \boldsymbol{\Lambda} = \text{diag}\{26.66, 100\}\ \text{[1/s]},\\ \varepsilon_N = 0.0004\ \text{[1/s}^2\text{]},\ \boldsymbol{\Gamma} = 30\ \boldsymbol{I}_{L\times L}\ \text{y}\ \alpha = 0.7\end{gathered}$$

para el esquema CPD-CNA.

El vector de funciones de activación $\boldsymbol{\Phi}(\boldsymbol{x})$ está dado por

$$\boldsymbol{\Phi}(\boldsymbol{x}) = \begin{bmatrix}\tanh(x_1) & \cdots & \tanh(x_L)\end{bmatrix}^T,$$

donde $\boldsymbol{x}$ se define en (3.3), y

$$\tanh(x) = \frac{2}{1 + e^{-\beta x}} - 1,$$

con $\beta = 2$ para los esquemas CNA y CN-AM, y con $\beta = 4$ para el esquema CPD-CNA. El número de neuronas en la capa oculta es $L = 11$. Se eligió la tangente hiperbólica como función de activación ya que es simétrica y puede tomar valores tanto positivos como negativos. Para medir el desempeño de los controladores se considera el error cuadrático medio (*root mean square error -RMS-*) de posición articular, este criterio está dado por

$$\widetilde{\boldsymbol{q}}_{RMS} = \sqrt{\frac{1}{\Delta T}\int_t^{t+\Delta T} \widetilde{\boldsymbol{q}}(\sigma)^T\ \widetilde{\boldsymbol{q}}(\sigma) d\sigma}.$$

El esquema CPD-CNA requiere de la matriz de inercias $\boldsymbol{M}(\boldsymbol{q})$, la cual está dada por

$$\boldsymbol{M}(\boldsymbol{q}) = \begin{bmatrix} M_{11}(\boldsymbol{q}) & M_{12}(\boldsymbol{q}) \\ M_{21}(\boldsymbol{q}) & M_{22}(\boldsymbol{q}) \end{bmatrix},$$

donde

$$\begin{aligned} M_{11}(\boldsymbol{q}) &= l_1^2 m_2 + 2 l_1 l_{c2} m_2 \cos(q_2) + l_{c2}^2 m_2 + l_{c1}^2 m_1 + I_1 + I_2, \\ M_{12}(\boldsymbol{q}) &= M_{21}(\boldsymbol{q}) = l_{c2}^2 m_2 + l_1 l_{c2} m_2 \cos(q_2) + I_2, \\ M_{22}(\boldsymbol{q}) &= l_{c2}^2 m_2 + I_2, \end{aligned}$$

y sus parámetros aproximados se resumen en la tabla 4.2.

La figura 4.1 muestra la evolución temporal de las posiciones $\boldsymbol{q}$ y las trayectorias de referencia $\boldsymbol{q_d}$ para los tres esquemas. Las figuras 4.3 y 4.4 muestran los errores de posición $\widetilde{\boldsymbol{q}}$ y las señales de control $\boldsymbol{\tau}$ obtenidas experimentalmente. Considerando que si bien todos los controladores lograron el objetivo de seguimiento evitando la saturación de entrada de los actuadores, el error de estado estable del primer eslabón muestra una respuesta más rápida para el esquema CPD-CNA. La señal de entrada del esquema propuesto no presenta exceso de ruido a pesar de incluir funciones de modo deslizante continuo. La figura 4.2 muestra los valores RMS de los errores de seguimiento $\widetilde{\boldsymbol{q}}_{RMS}$ con una variación de tiempo ΔT=5 [s]. La tabla 4.1 muestra los valores RMS de los errores de seguimiento $\widetilde{\boldsymbol{q}}_{RMS}$ durante todo el tiempo del experimento.

A partir de los resultados experimentales es posible observar que el uso de la función potencia signada junto con el pequeño incremento en β aceleran la convergencia del error de estimación de pesos $\widehat{\boldsymbol{W}}$; esto, a su vez, mejora el rendimiento del controlador CPD-CNA. Se puede observar que todos los controladores logran compensar la fricción a pesar de la velocidad relativamente baja de la trayectoria. El esquema propuesto CPD-CNA muestra un mejor desempeño general, a pesar de que el segundo eslabón necesita un período corto de tiempo de aprendizaje para poder desempeñarse mejor que el esquema CNA.

Tabla 4.1: Valores RMS para los errores de posición con una variación de tiempo ΔT=30 [s].

Esquema	$\widetilde{q}_{RMS}$
CNA	0.3208 [rad]
CN-AM	0.4969 [rad]
CPD-CNA	0.0846 [rad]

Tabla 4.2: Parámetros físicos aproximados del robot manipulador de 2 g.d.l.

Descripción	Notación	Valor	Unidades
Longitud del eslabón 1	l_1	0.45	m
Longitud del eslabón 2	l_2	0.45	m
Distancia al centro de masa (eslabón 1)	l_{c1}	0.091	m
Distancia al centro de masa (eslabón 2)	l_{c2}	0.048	m
Masa del eslabón 1	m_1	23.902	Kg
Masa del eslabón 2	m_2	3.88	Kg
Inercia del eslabón 1	I_1	1.266	Kg-m^2
Inercia del eslabón 2	I_2	0.093	Kg-m^2
Aceleración de la gravedad	g	9.81	m/s^2

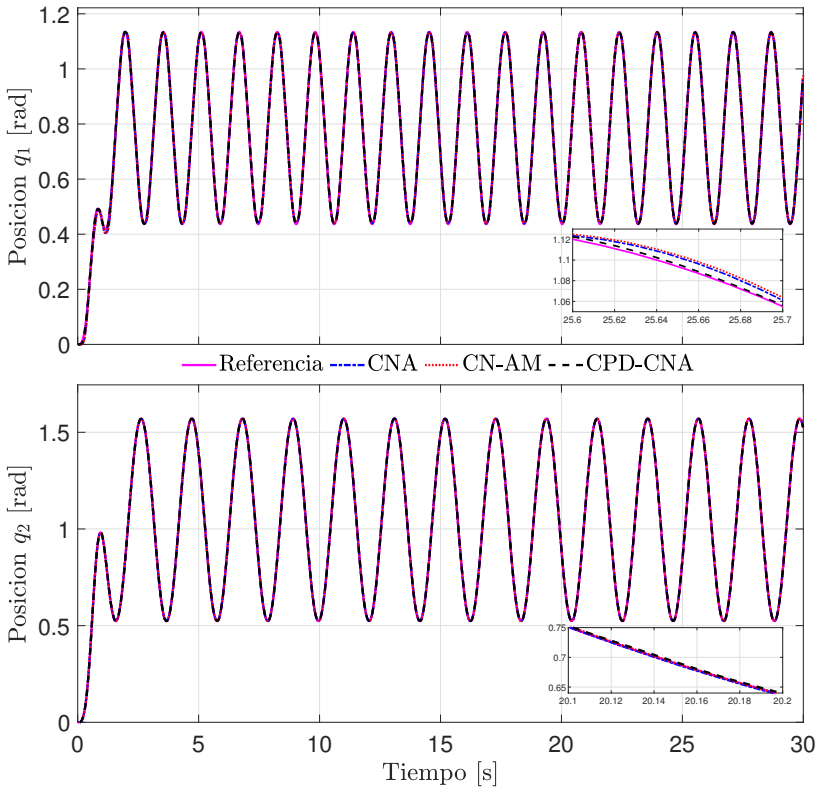

Figura 4.1: Respuesta de posición del sistema.

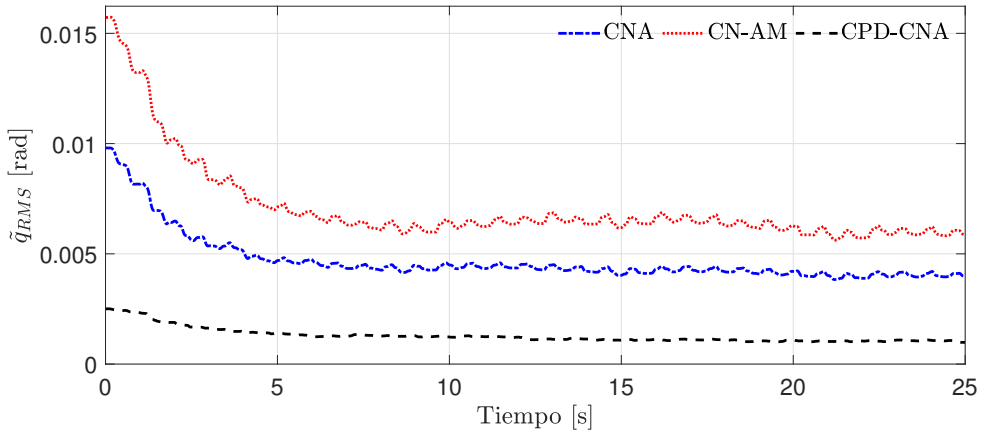

Figura 4.2: Valores RMS para los errores de posición con ΔT=5 [s].

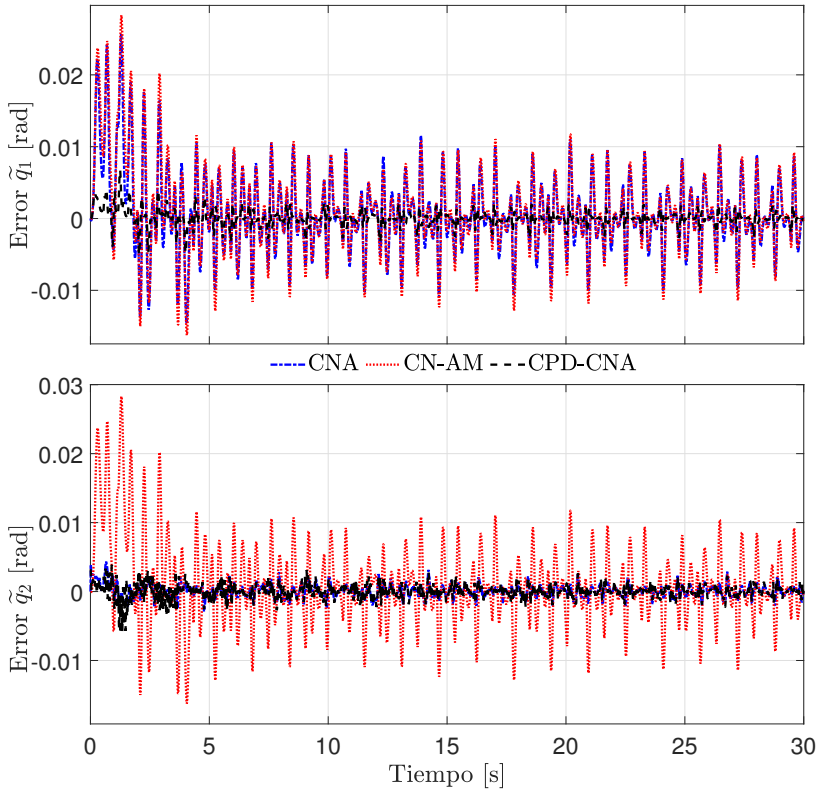

Figura 4.3: Errores de posición del sistema.

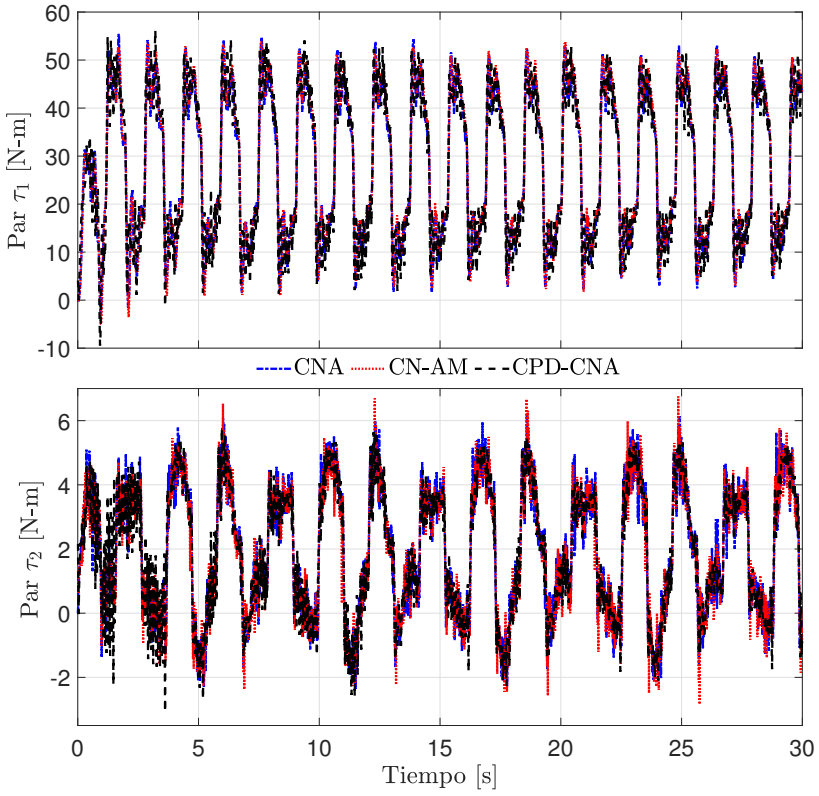

Figura 4.4: Señales de control del sistema.

4.2 Resultados experimentales con perturbaciones para el robot manipulador de 2 g.d.l.

En esta sección, se agregan perturbaciones externas para probar la robustez de los esquemas CNA y CPD-CNA, ya que el esquema CN-AM no tiene prueba de estabilidad. Las perturbaciones externas, para ambos esquemas, tienen la forma:

$$\boldsymbol{\delta} = \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{[N-m], con } 0 \leq t < 10 \tag{4.1}$$

$$\boldsymbol{\delta} = \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} 1 + 30\,\text{sen}(t) \\ 1 + 3\,\text{sen}(t) \end{bmatrix} \text{[N-m], con } t \geq 10 \tag{4.2}$$

La figura 4.5 muestra la evolución temporal de las posiciones $\boldsymbol{q}$ y las trayectorias de referencia $\boldsymbol{q_d}$, las figuras 4.6 y 4.7 muestran los errores de posición $\widetilde{\boldsymbol{q}}$ y las señales de control $\boldsymbol{\tau}$ obtenidas experimentalmente para ambos controladores con la perturbación definida en (4.1-4.2). Tenga en cuenta que mientras el esquema CPD-CNA logra el objetivo de seguimiento sin saturar los actuadores, el esquema CNA satura el segundo eslabón, aproximadamente a los $t = 22$ segundos, por lo que el experimento se tuvo que detener, aproximadamente a los $t = 27$ segundos, para evitar dañar el motor. En la figura 4.5 se puede ver que las posiciones $\boldsymbol{q}$ para el esquema CNA se van a cero después del tiempo $t = 27$ segundos debido al paro de operación por razones de seguridad.

La amplitud de la perturbación es de 30 [N-m] para el primer eslabón y de 3 [N-m] para el segundo eslabón; que están cerca de la mitad del valor de par máximo en los experimentos sin perturbaciones; es decir, las perturbaciones externas son lo suficientemente fuertes. Como se puede notar en las figuras 4.5 a 4.7, a diferencia del esquema CNA, el esquema CPD-CNA tiene un comportamiento robusto frente a perturbaciones externas.

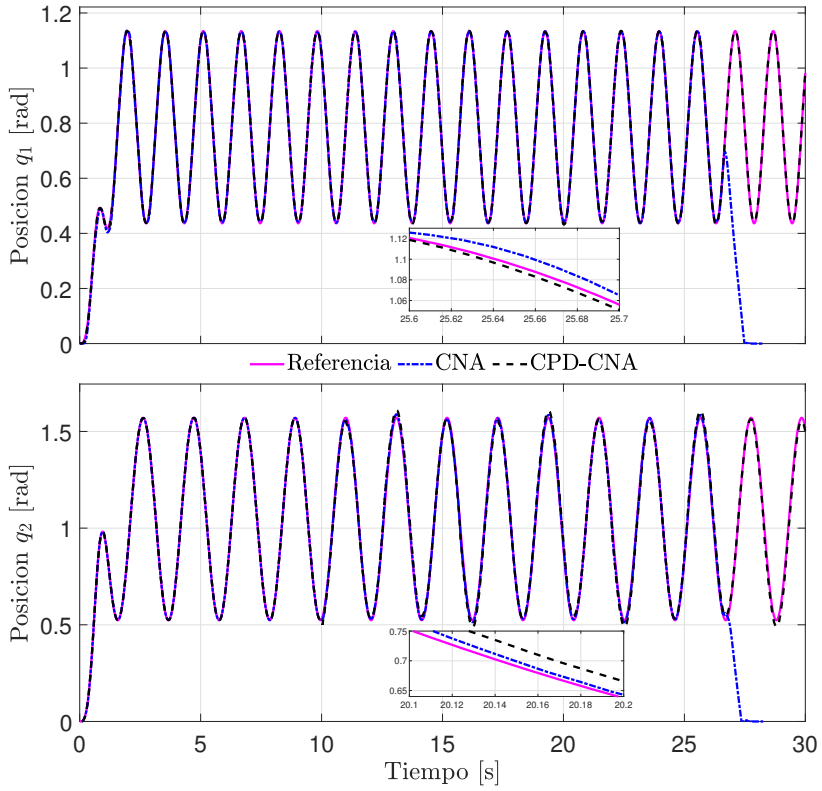

Figura 4.5: Respuesta de posición del sistema con perturbaciones externas.

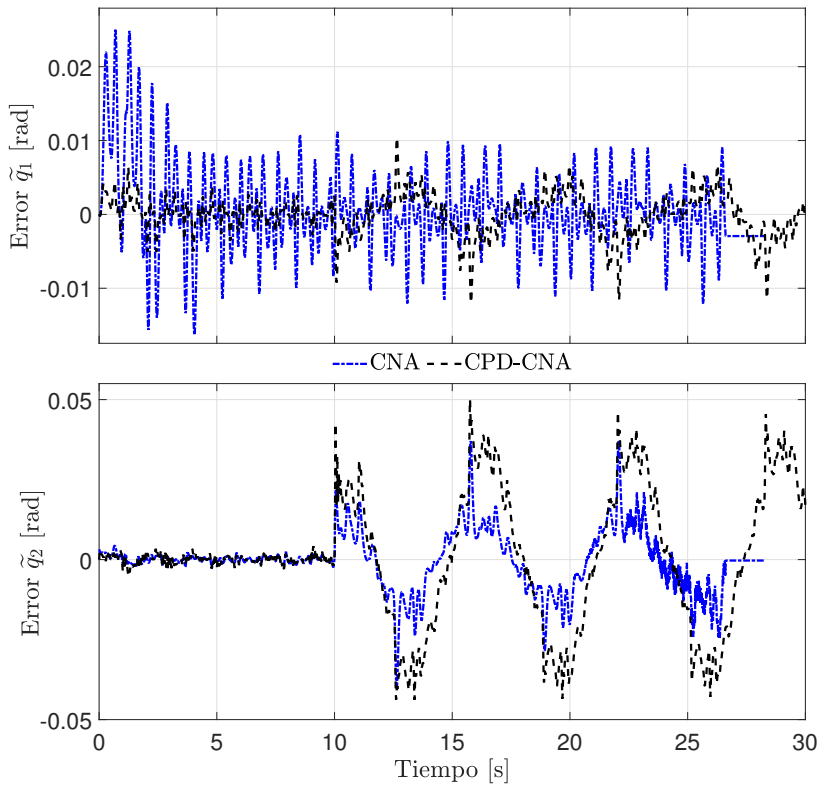

Figura 4.6: Errores de posición del sistema con perturbaciones externas.

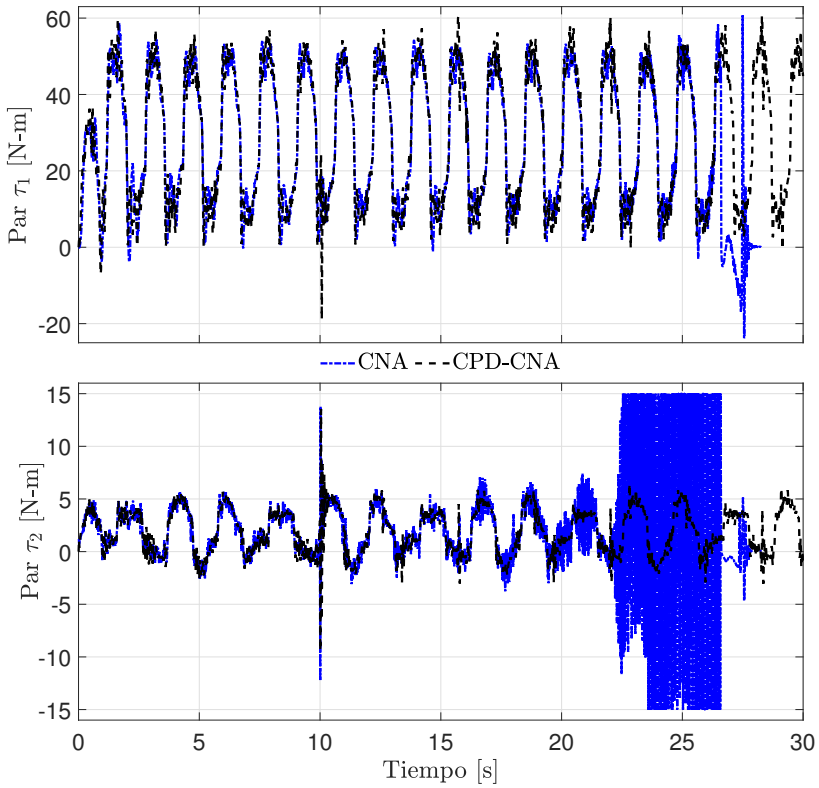

Figura 4.7: Señales de control del sistema con perturbaciones externas.

4.3 Resultados experimentales para el robot móvil omnidireccional Nexus

En esta sección se realiza el control cinemático del robot móvil omnidireccional Nexus (ver figura 4.8), es decir, se controla la posición de las ruedas y las trayectorias deseadas; se calculan haciendo uso de la cinemática inversa del robot. Para este robot se implementa únicamente el esquema CNA, por ser este suficiente para la tarea. Los esquemas vistos en el capítulo 3 se pueden aplicar al control cinemático, ya que, éstos admiten cualquier sistema mecánico Lagrangiano totalmente actuado[1]. Para todos los experimentos realizados en el robot Nexus se han seleccionado las siguientes trayectorias deseadas:

1: Para aplicar estos esquemas en sistemas subactuados se requiere ampliar la metodología.

$$\boldsymbol{q_d} = \begin{bmatrix} \text{sen}\,(\omega\, t) \\ \cos\,(\omega\, t) \\ -\omega\, t \end{bmatrix} \begin{matrix} [\text{m}] \\ [\text{m}] \\ [\text{rad}] \end{matrix} \tag{4.3}$$

donde $\omega = 2\pi/15$ es la frecuencia del circulo y de giro del robot. Las condiciones iniciales $\boldsymbol{q}(0) = [0, 1, 0]^T$ y $\widehat{\boldsymbol{W}}(0) = \boldsymbol{0}_{3\times 4}$ fueron tomadas para los controladores.

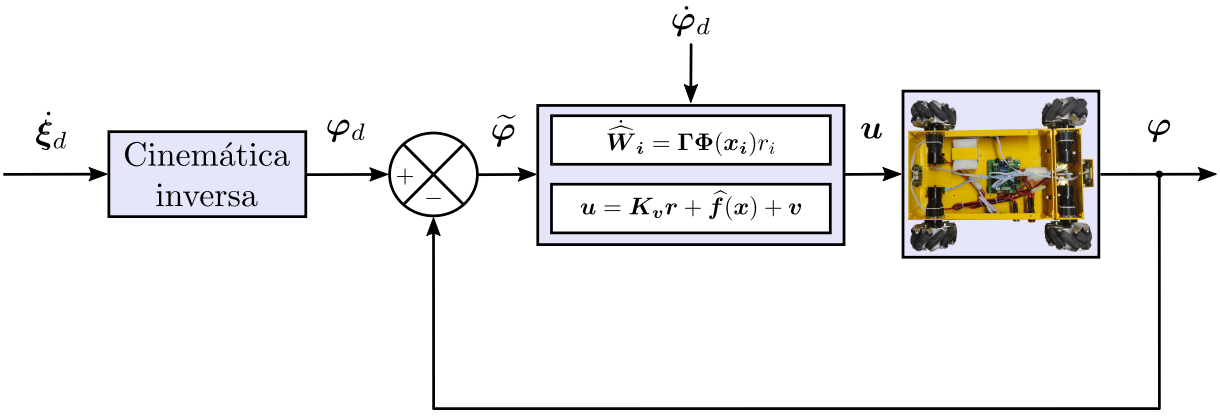

Figura 4.8: Diagrama para control cinemático en espacio articular utilizando el esquema CNA.

Las ganancias resultantes de la sintonización están dadas por:

$$\boldsymbol{K_v} = 25\ \boldsymbol{I}_{4\times4}\ [\mathrm{V\,s/rad}], \quad \boldsymbol{K_p} = 150\ \boldsymbol{I}_{4\times4}\ [\mathrm{V/rad}],$$
$$\boldsymbol{\Lambda} = 6\ \boldsymbol{I}_{4\times4}\ [1/\mathrm{s}], \quad \varepsilon_N = 0.001\ [\mathrm{V}] \quad \text{y} \quad \boldsymbol{\Gamma} = 0.05\ \boldsymbol{I}_{L\times L}.$$

El estimado de la función $\widehat{\boldsymbol{f}}(\boldsymbol{x})$ está dado por

$$\widehat{\boldsymbol{f}}(\boldsymbol{x}) = \left[\widehat{\boldsymbol{W}}_1^T\boldsymbol{\Phi}(\boldsymbol{x}_1) \quad \widehat{\boldsymbol{W}}_2^T\boldsymbol{\Phi}(\boldsymbol{x}_2) \quad \widehat{\boldsymbol{W}}_3^T\boldsymbol{\Phi}(\boldsymbol{x}_3) \quad \widehat{\boldsymbol{W}}_4^T\boldsymbol{\Phi}(\boldsymbol{x}_4)\right]^T,$$

donde $\boldsymbol{x_i} \in \mathbb{R}^L$ se define como

$$\boldsymbol{x_i} \equiv \left[\tilde{\varphi}_i^T \quad \dot{\tilde{\varphi}}_i^T \quad \dot{\varphi}_{di}^T \quad 1\right]^T$$

para cada i-ésima rueda, y $\tanh(x) = \dfrac{2}{1+e^{-\beta x}} - 1$, con $\beta = 2$.

En las siguientes figuras se presentan las gráficas de los resultados experimentales del control cinemático. Las figuras 4.9-4.10 muestran la evolución temporal de las posiciones $\boldsymbol{\xi}$ y las trayectorias de referencia $\boldsymbol{\xi_d}$ para el esquema CNA. La figura 4.11 muestra la evolución temporal de las velocidades de los motores $\dot{\boldsymbol{\varphi}}$ y las velocidades de referencia $\dot{\boldsymbol{\varphi}}_{\boldsymbol{d}}$ para el esquema CNA. Las figuras 4.12-4.13 muestran los errores de velocidad $\dot{\tilde{\boldsymbol{\varphi}}}$ y los voltajes $\boldsymbol{u}$ para el esquema CNA obtenidas experimentalmente.

Note que al implementar un control cinemático no se hace medición directa de las variables de configuración $\boldsymbol{\xi}$, por lo tanto, las figuras 4.9-4.10 se obtienen a través de la cinemática del robot.

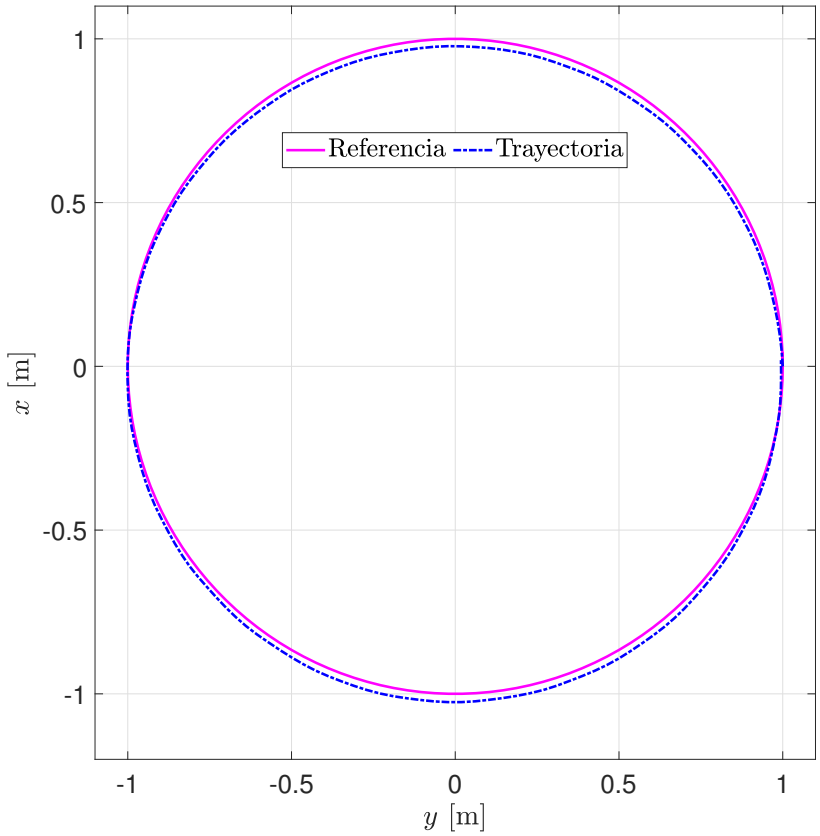

Figura 4.9: Respuesta del sistema en traslación para el esquema CNA.

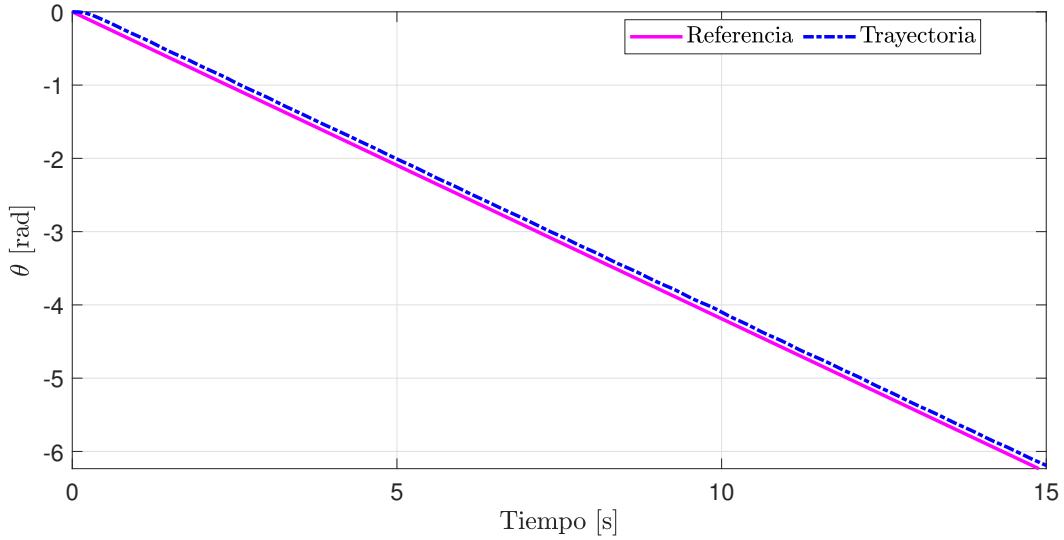

Figura 4.10: Respuesta del sistema en orientación para el esquema CNA.

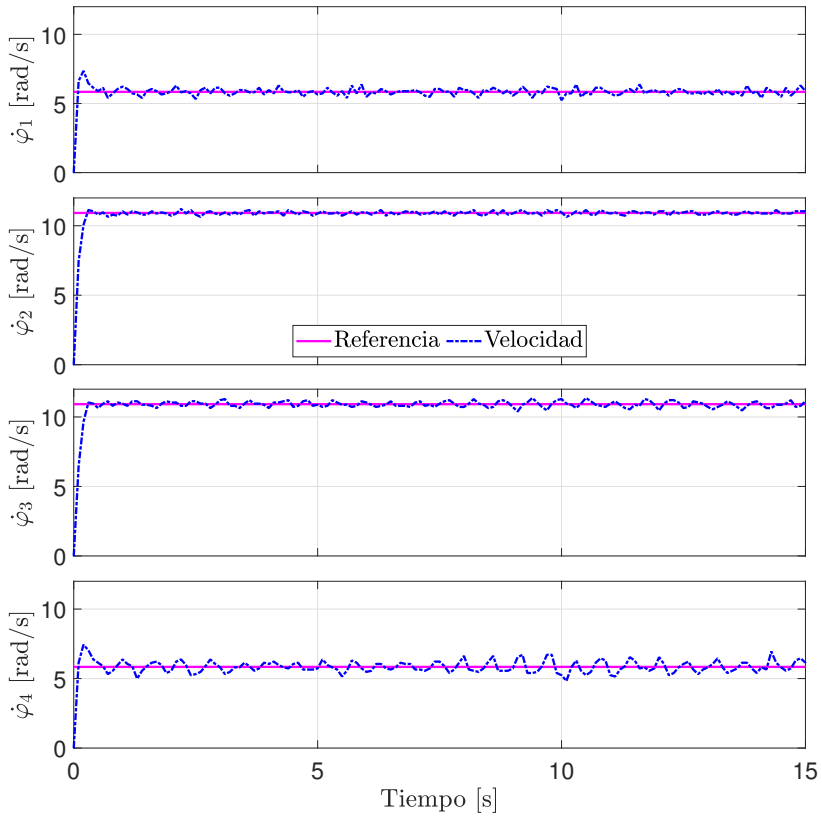

Figura 4.11: Respuesta en velocidad de los motores para el esquema CNA.

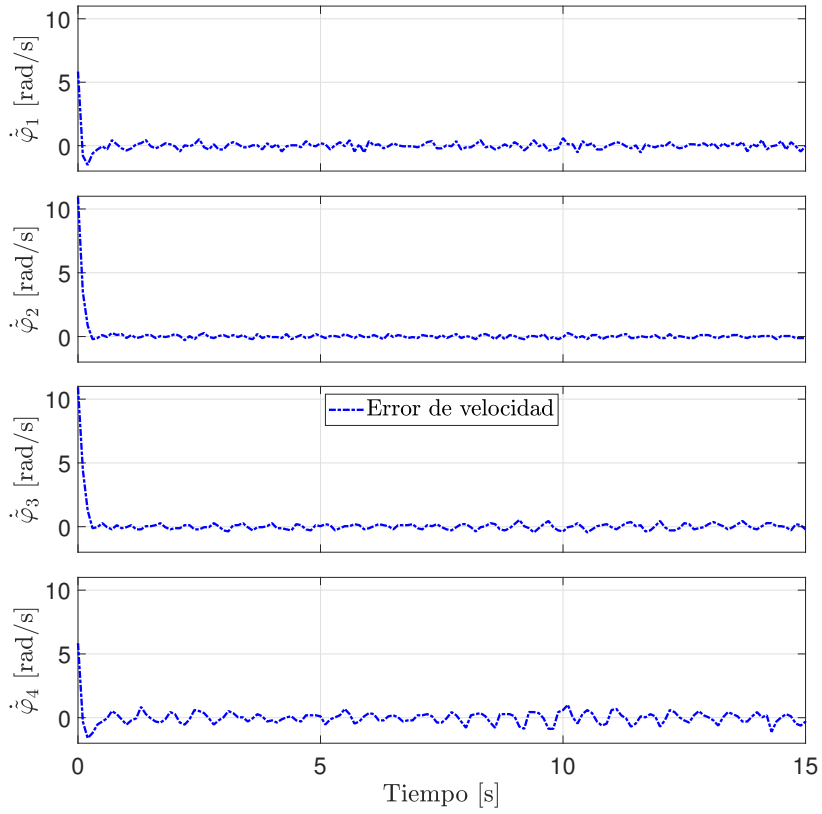

Figura 4.12: Errores de velocidad de los motores para el esquema CNA.

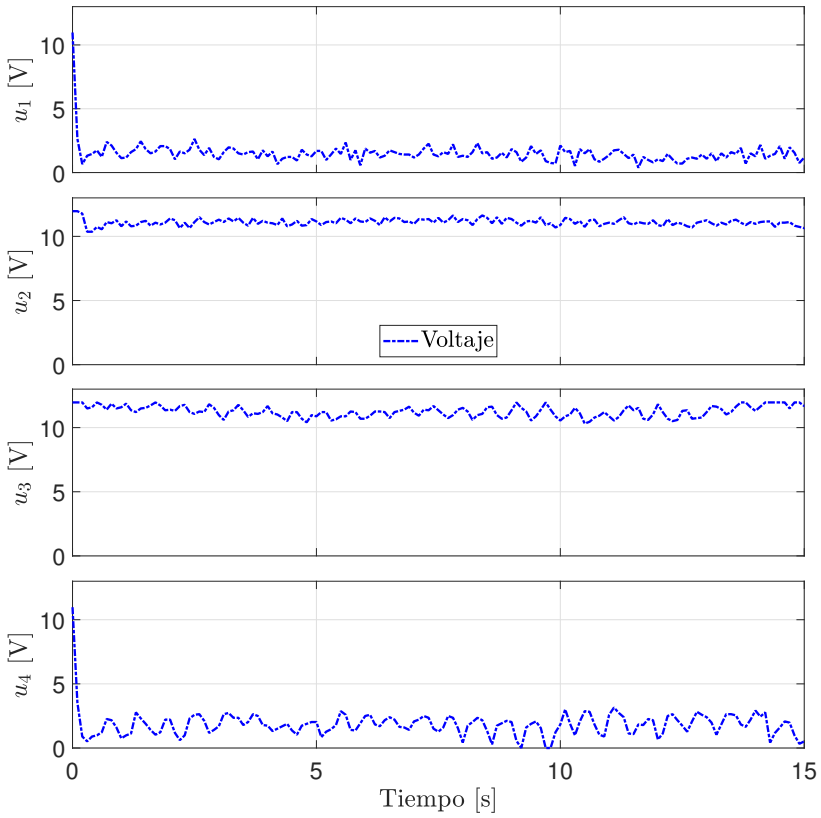

Figura 4.13: Voltajes de los motores para el esquema CNA.

4.4 Conclusiones y observaciones

Los experimentos en el robot manipulador de 2 g.d.l. mostraron que el esquema CPD-CNA fue superior tanto al esquema CNA como al esquema CN-AM. Aunque el esquema CPD-CNA está diseñado con el supuesto de conocimiento de la matriz inercial $\boldsymbol{M}(\boldsymbol{q})$, ésta puede sustituirse por su estimado y funcionará correctamente, por lo tanto, el conocimiento del modelo es no estrictamente necesario.

Los experimentos en el robot omnidireccional Nexus mostraron que el esquema CNA es suficientemente bueno para compensar los distintos coeficientes de fricciones de cada rueda sin saturar los voltajes de los motores. Asimismo, se debe mencionar que para aumentar la tracción de las ruedas se utilizan pliegos de neopreno entre el robot y el piso.

En general los resultados mostraron que los controladores neuronales de una capa diseñados consiguen un buen desempeño sin mostrar excesivo ruido en las señales de control. El esquema propuesto CPD-CNA fue capaz de compensar la fricción, los parámetros inciertos del sistema y tuvo un

comportamiento robusto frente a perturbaciones externas fuertes.

Detección de objetos usando técnicas de aprendizaje automático

5

En este capítulo se presentan algunas de las técnicas más importantes de aprendizaje automático (*machine learning*) orientadas a visión. Asimismo, se abordan los principales exponentes en el rubro de interpretación de imágenes. Por último, se evalúa el desempeño del detector de objetos Tiny-YOLOv2 especializado en sistemas embebidos.

5.1 Transferencia de aprendizaje

La técnica de transferencia de aprendizaje (*transfer learning*) actualmente es muy popular en el ámbito del aprendizaje profundo, ya que, permite entrenar redes neuronales profundas con relativamente pocos datos [46, 47]. La metodología de la transferencia de aprendizaje consiste en utilizar una RN previamente entrenada para facilitar el entrenamiento de una nueva red, la cual resuelve un problema distinto al original (ver figura 5.1). Esta técnica resulta ser de gran utilidad dado que los problemas del mundo real no suelen tener millones de datos etiquetados con el fin de entrenar modelos neuronales complejos.

[46]: Zhuang y col. (2019), *A Comprehensive Survey on Transfer Learning*
[47]: Tan y col. (2018), *A Survey on Deep Transfer Learning*

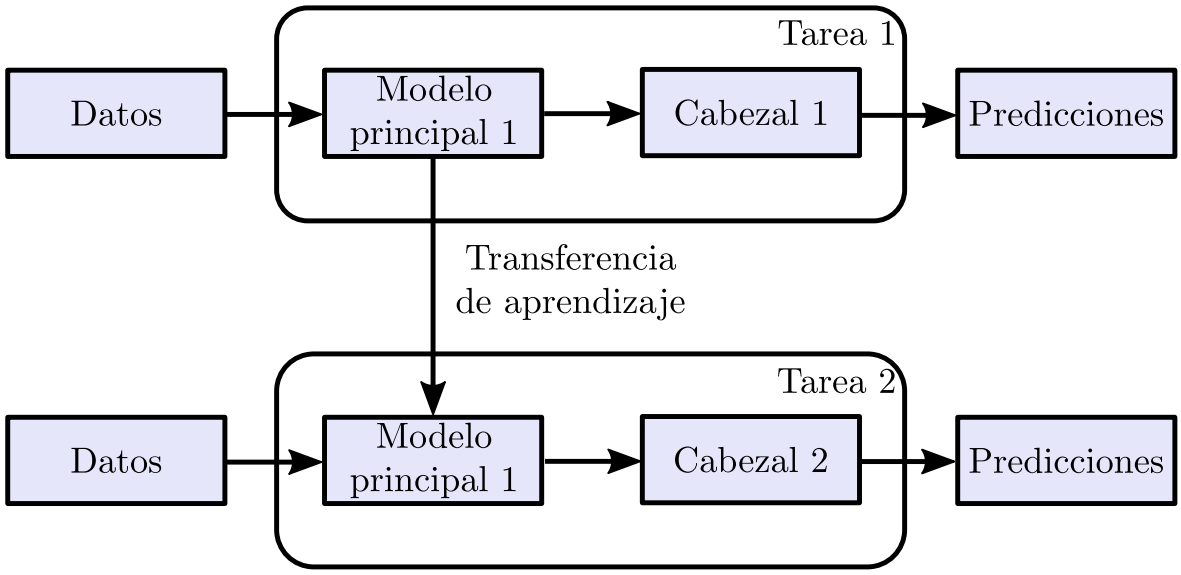

Figura 5.1: Diagrama de la técnica de transferencia de aprendizaje.

5.2 Redes neuronales convolucionales (RNCs)

Las RNCs son una de las categorías más populares del aprendizaje profundo, y toman su nombre de la operación matemática llamada convolución. Las RNCs han demostrado ser especialmente eficientes en áreas de visión como clasificación y detección de imágenes; sobresaliendo en la identificación de caras, objetos y señales de tráfico, además de potenciar la visión de robots y vehículos autónomos .

La función de las RNCs en visión es reducir las imágenes de tal forma que sean más fácil de procesar, sin perder características que son críticas para obtener una buena predicción; por esta razón se les llama también extractores de características (*feature extraction*) [48, 49].

[48]: Albawi y col. (2017), "Understanding of a convolutional neural network"
[49]: Dhillon y col. (2020), "Convolutional neural network: a review of models, methodologies and applications to object detection"

Los algoritmos de detección de objetos, a diferencia de los algoritmos de clasificación de objetos, producen cuadros delimitadores (*bounding boxes*) alrededor de los objetos de interés para localizarlos dentro de la imagen y después clasificarlos; además son capaces de detectar múltiples objetos de una misma clase.

5.3 Redes neuronales convolucionales basadas en regiones

Una manera sencilla de resolver el problema de detección de objetos es proponer una variedad de regiones de interés en una imagen y usar una RNC para clasificar los objetos dentro de cada región; el principal problema de esta metodología consiste en que los objetos de interés pueden tener diferentes ubicaciones y relaciones de aspecto dentro de la imagen. Por lo tanto, este método requiere la selección de una gran cantidad de regiones de interés, esto implica un alto costo computacional.

Para evitar seleccionar una gran cantidad de regiones se han desarrollado sistemas que proponen estas regiones, después las regiones propuestas son procesadas por una RNC; un ejemplo de estos algoritmos de dos etapas son las RNCs basadas en regiones (ver figura 5.2) [50, 51].

[50]: Girshick y col. (2013), "Rich feature hierarchies for accurate object detection and semantic segmentation"
[51]: He y col. (2017), *Mask R-CNN*

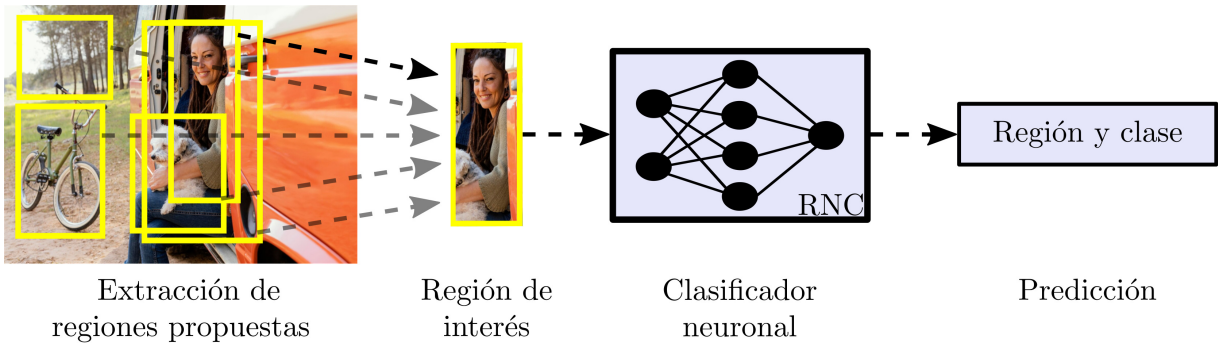

Figura 5.2: Arquitectura de las RNCs basadas en regiones.

5.4 ONNX

ONNX (*Open Neural Network eXchange*) es un formato estándar de código abierto para representar e intercambiar modelos de aprendizaje automático [52]. El objetivo principal de ONNX es simplificar el intercambio de modelos entre plataformas, lo cual permite ensamblar modelos en cualquier tipo de configuración o equipo. Algunos ejemplos de plataformas compatibles son TensorFlow, PyTorch, SciKit-Learn, Keras, Chainer, MXNet, MATLAB, SparkML, entre otros.

[52]: Jin y col. (2020), *Compiling ONNX Neural Network Models Using MLIR*

5.5 YOLO

La red YOLO (*You Only Look Once*) es un sistema detector de objetos orientado a aplicaciones en tiempo real [53-55], este sistema es de una sola etapa a diferencia de los basados en regiones. En la red YOLO una sola RNC predice los cuadros delimitadores y las probabilidades de pertenecer a una clase; esto se logra dividiendo la imagen de entrada en una cuadricula de $S \times S$, y dentro de la cuadricula se toman m cuadros delimitadores (ver figura 5.3). La red YOLO produce los cuadros delimitadores y la probabilidad de pertenecer a una clase siempre y cuando la probabilidad se encuentre por encima de un valor umbral definido.

Actualmente se desarrollan diversas versiones de YOLO, las más recientes incluyendo mecanismos de atención [37, 56-58]; Matlab en sus versiones 2020a, 2021a y 2022a soporta la segunda, tercera y cuarta versión de YOLO respectivamente. Se eligió la red YOLO por su buen desempeño, además de contar con una versión reducida llamada TinyYOLO especializada en sistemas de bajos recursos.

[53]: Redmon y col. (2015), *You Only Look Once: Unified, Real-Time Object Detection*
[54]: Redmon y col. (2016), "YOLO9000: Better, Faster, Stronger"
[55]: Redmon y col. (2018), *YOLOv3: An Incremental Improvement*

[37]: Vaswani y col. (2017), *Attention Is All You Need*
[56]: Zhu y col. (2021), *TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios*
[57]: Wang y col. (2021), *You Only Learn One Representation: Unified Network for Multiple Tasks*
[58]: Bochkovskiy y col. (2020), *YOLOv4: Optimal Speed and Accuracy of Object Detection*

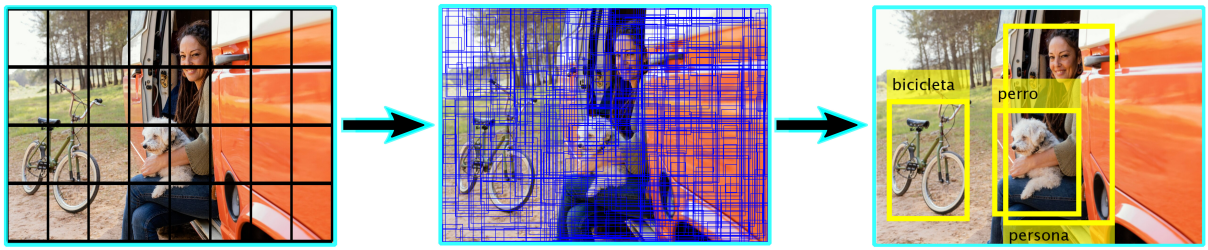

Figura 5.3: Metodología de detección de la red YOLO.

El modelo TinyYOLOv2 cuenta con 34 capas y 20 clases (la versión completa cuenta con 77 capas y 80 clases), las clases de este modelo son: *aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse, motorbike, person, pottedplant, sheep, sofa, train* y *tvmonitor*. La descarga de este modelo en formato ONNX está disponible en [59].

[59]: GitHub repository (2020), *TinyYOLOv2 Model*

5.6 ResNet

ResNet (*Residual Network*) es una RNC que cuenta con diferentes configuraciones (50, 101, 152 y hasta 177 capas), y puede clasificar imágenes entre 1000 categorías de objetos diferentes [60]. En 2015 ResNet consiguió ganar la competencia de clasificación ILSVRC (*ImageNet Large-Scale Visual Recognition Challenge*), siendo así capaz de vencer incluso a GoogLeNet. El modelo ResNet es parte del *Deep Learning Toolbox* de Matlab, y se obtiene con la siguiente función:

[60]: He y col. (2015), *Deep Residual Learning for Image Recognition*

```
net = resnet50
```

5.7 Proyección de coordenadas

Luego de que se tiene entrenada y funcionando la RN, es necesario implementar un método de conversión de coordenadas; este método debe ser capaz de convertir las coordenadas de una imagen a coordenadas en el mundo real y viceversa. La proyección de coordenadas del mundo real a coordenadas de la imagen se muestra en la siguiente ecuación [61]:

$$\begin{bmatrix} \zeta\chi & \zeta\lambda & \zeta \end{bmatrix}^T = \boldsymbol{P} \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T \tag{5.1}$$

donde $\begin{bmatrix} \chi & \lambda \end{bmatrix}^T$ son las coordenadas de la imagen en píxeles, ζ es un factor de escalamiento, $\boldsymbol{P} \in \mathbb{R}^{3\times4}$ es la matriz de

[61]: Juarez-Salazar y col. (2020), "Distorted pinhole camera modeling and calibration"

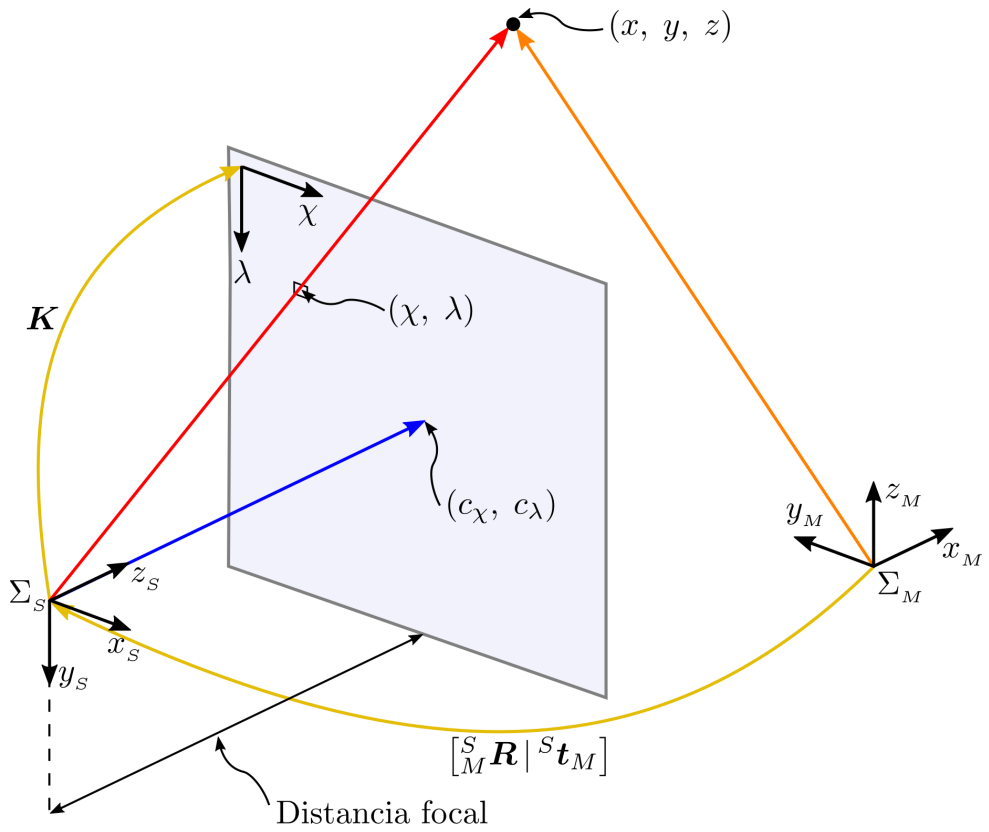

Figura 5.4: Diagrama para la proyección de coordenadas.

proyección y $\begin{bmatrix} x & y & z \end{bmatrix}^T$ son las coordenadas del mundo real en metros, siendo z siempre una constante conocida. La matriz de proyección está definida como:

$$\boldsymbol{P} = \boldsymbol{K}\left[{}^S_M\boldsymbol{R}\,|\,{}^S\boldsymbol{t}_M\right]$$

donde

$$\boldsymbol{K} = \begin{bmatrix} f_\chi & 0 & c_\chi \\ 0 & f_\lambda & c_\lambda \\ 0 & 0 & 1 \end{bmatrix}$$

es la matriz de parámetros intrínsecos de la cámara, $\begin{bmatrix} f_\chi & f_\lambda \end{bmatrix}^T$ es la distancia focal de la cámara en píxeles en dirección de χ y λ respectivamente, $\begin{bmatrix} c_\chi & c_\lambda \end{bmatrix}^T$ es el punto principal de la cámara en píxeles, $\left[{}^S_M\boldsymbol{R}\,|\,{}^S\boldsymbol{t}_M\right]$ es la matriz de parámetros extrínsecos de la cámara, ${}^S_M\boldsymbol{R}$ es la matriz de rotación del marco del sensor Σ_S al marco móvil Σ_M, y ${}^S\boldsymbol{t}_M$ es el vector de traslación del marco móvil Σ_M respecto al marco del sensor Σ_S. La figura 5.4 muestra el diagrama para la proyección de coordenadas del mundo real a coordenadas de la imagen.

La proyección de coordenadas de la imagen a coordenadas del mundo real se muestra en la siguiente ecuación:

$$\begin{bmatrix} x & y & z \end{bmatrix}^T = \boldsymbol{P}^{-1} \begin{bmatrix} \zeta\chi & \zeta\lambda & \zeta & 1 \end{bmatrix}^T \tag{5.2}$$

donde $\boldsymbol{P}^{-1} \in \mathbb{R}^{3\times4}$ es la matriz de proyección inversa, la cual está definida como:

$$\boldsymbol{P}^{-1} = \begin{bmatrix} {}^{M}_{S}\boldsymbol{R} \mid {}^{M}\boldsymbol{t}_S \end{bmatrix} \bar{\boldsymbol{K}}^{-1}$$

donde

$$\bar{\boldsymbol{K}}^{-1} = \begin{bmatrix} 1/f_\chi & 0 & -c_\chi/f_\chi & 0 \\ 0 & 1/f_\lambda & -c_\lambda/f_\lambda & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

es la matriz inversa extendida de parámetros intrínsecos de la cámara[1], $\begin{bmatrix} {}^{M}_{S}\boldsymbol{R} \mid {}^{M}\boldsymbol{t}_S \end{bmatrix}$ es la matriz inversa de parámetros extrínsecos de la cámara, ${}^{M}_{S}\boldsymbol{R}$ es la matriz de rotación del marco móvil Σ_M al marco del sensor Σ_S, y ${}^{M}\boldsymbol{t}_S$ es el vector de traslación del marco del sensor Σ_S respecto al marco móvil Σ_M.

1: Es necesario expandir esta matriz, ya que, de no hacerlo se pierde el vector de traslación ${}^{M}\boldsymbol{t}_S$ en la proyección inversa.

Es posible estimar los parámetros intrínsecos de la cámara con la siguiente función de Matlab:

```
estimateCameraParameters()
```

mientras que los parámetros extrínsecos se pueden estimar con la siguiente función de Matlab:

```
extrinsics()
```

5.8 Aplicación del modelo TinyYOLOv2

Python es el lenguaje con mayor popularidad para el manejo del aprendizaje automático [62]; actualmente Matlab es una excelente opción para implementar detectores de objetos neuronales, ya que, brinda herramientas para entrenar, importar y aplicar redes neuronales de todo tipo. Por todo esto, se decidió trabajar con Matlab para la aplicación de detectores de objetos neuronales.

[62]: Raschka y col. (2020), *Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence*

El procedimiento para ejecutar la red TinyYOLOv2 en Matlab se describe enseguida: en primer lugar se importa el modelo del detector en formato ONNX, posteriormente se deben adecuar tanto la estructura como algunos parámetros del modelo, por último es necesario ensamblar la RN.

En las figuras 5.5-5.10 se muestran algunos resultados de la aplicación del modelo TinyYOLOv2. Para cada predicción se incluye la clase y cuadro delimitador, se omite la probabilidad de pertenencia por cuestiones de visibilidad. Todas las imágenes utilizadas son parte de conjuntos de datos sin derechos de autor.

Asimismo el modelo TinyYOLOv2 se aplica a dos vídeos de un recorrido vial, los vídeos resultantes se encuentran en los siguientes enlaces:

- https://youtu.be/2CzaixJ0Tls.
- https://youtu.be/tTVSKILxnCk.

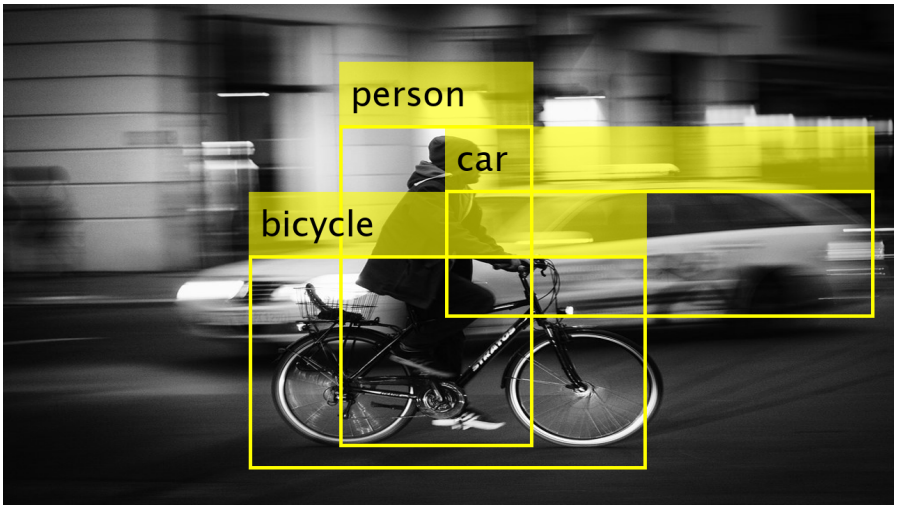

Figura 5.5: Ejemplo 1 de aplicación de TinyYOLOv2 en imágenes.

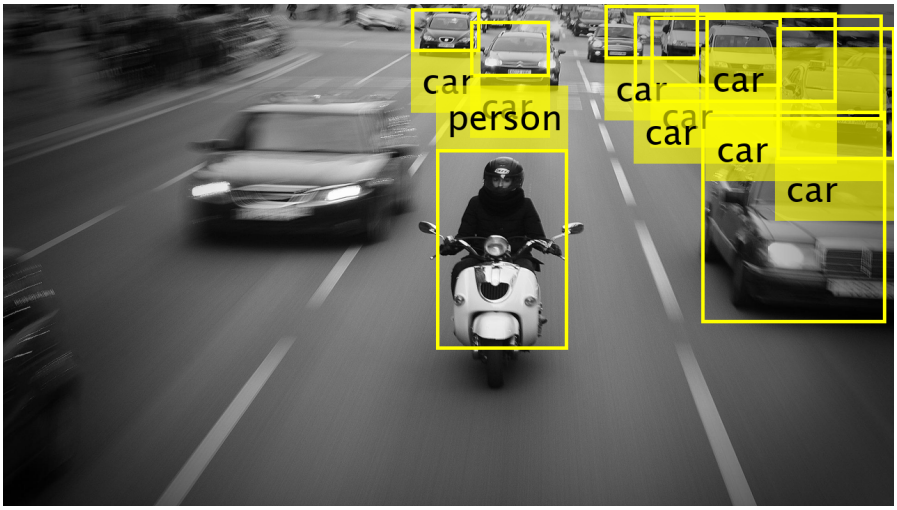

Figura 5.6: Ejemplo 2 de aplicación de TinyYOLOv2 en imágenes.

Figura 5.7: Ejemplo 3 de aplicación de TinyYOLOv2 en imágenes.

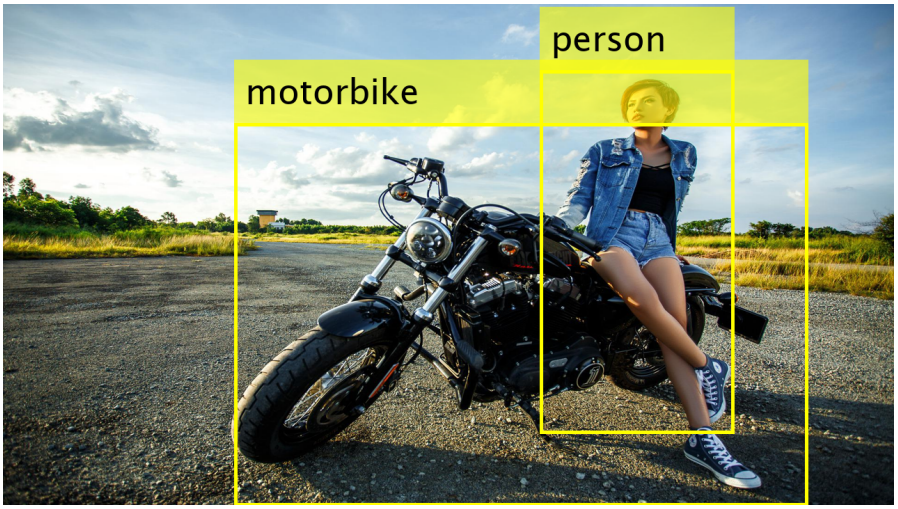

Figura 5.8: Ejemplo 4 de aplicación de TinyYOLOv2 en imágenes.

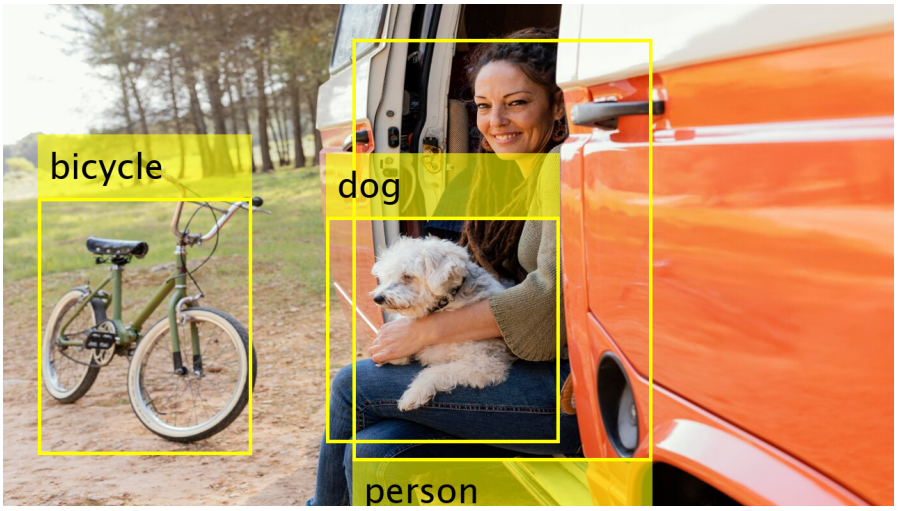

Figura 5.9: Ejemplo 5 de aplicación de TinyYOLOv2 en imágenes.

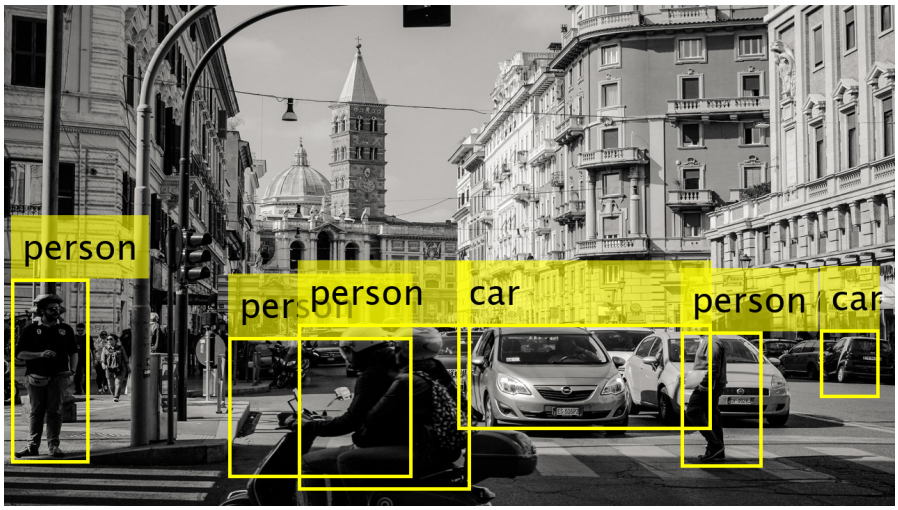

Figura 5.10: Ejemplo 6 de aplicación de TinyYOLOv2 en imágenes.

5.9 Conclusiones y observaciones

Es posible aplicar la técnica de transferencia de aprendizaje tanto a la red TinyYOLOv2 como a ResNet en cualquiera de sus configuraciones. Esta técnica puede ser aplicada aún cuando el modelo base sea un clasificador y el modelo objetivo un detector.

Un aspecto importante para la aplicación de cualquier detector es que, siempre que sea posible, se debe seleccionar una región de interés; esta región de interés luego será la entrada del detector.

La red TinyYOLOv2 mostró ser muy versátil al mostrar muy buenos resultados, además de aceptar una gran variedad de imágenes de diferentes orígenes, tamaños y relaciones de aspecto. Otra ventaja de la red YOLO, a diferencia de otras arquitecturas, es que no restringe el tamaño de las imágenes de entrada a un tamaño en particular.

La baja tasa de error de sus predicciones en conjunto con sus bajos tiempos de inferencia hacen a la red tinyYOLOv2 ideal para aplicaciones en sistemas embebidos.

Vehículo detector de ruta 6

En este capítulo se presentan los elementos que conforman al vehículo detector de ruta. La tarea de esta plataforma es detectar una ruta utilizando una cámara abordo, para luego ejecutarla; la plataforma hace uso de recursos como aprendizaje profundo, proyección de coordenadas, generador de trayectorias y control cinemático. Por último se implementan recursos para detectar posibles obstáculos y posteriormente tomar acciones en consecuencia. El vehículo detector de ruta se implementa tanto en simulación como en tiempo real.

6.1 Detector de ruta usando YOLOv2

Para desarrollar un sistema detector de ruta se emplea la metodología de transferencia de aprendizaje, la cual utiliza como base una RN previamente entrenada y así agilizar el entrenamiento. Antes de poder entrenar cualquier RN es necesario tener un conjunto de imágenes con sus respectivas etiquetas de lo que se desea detectar.

Las imágenes requeridas se obtienen realizando un recorrido de la ruta que se pretende detectar. Luego, las imágenes obtenidas se etiquetan usando la aplicación de etiquetado de imágenes *Ground Truth Labeler* de Matlab que es parte del *Toolbox* llamado *Automated Driving Toolbox*[1].

Una vez obtenido el conjunto de imágenes etiquetadas se procede a entrenar una RN capaz de detectar correctamente la ruta deseada. El entrenamiento de la RN se lleva a cabo con la siguiente función de Matlab:

```
1    trainYOLOv2ObjectDetector()
```

Este mismo procedimiento se puede aplicar para entrenar un detector que desempeñe cualquier otra tarea.

Por último, las detecciones se convierten en coordenadas de la imagen obteniendo los centroides de los cuadros delimitadores resultantes.

1: Matlab cuenta con un *Toolbox* de conducción autónoma llamado *Automated Driving Toolbox* el cual proporciona algoritmos y herramientas para diseñar, simular y probar sistemas de conducción autónoma y sistemas de asistencia al conductor. Este *Toolbox* permite diseñar y probar sistemas de visión (incluyendo lidars), así como fusión de sensores, planificación de rutas y controladores de vehículos.

6.2 Generador de trayectorias deseadas con modulador de velocidad

Después que se detecta la ruta y que las coordenadas de la imagen se proyecten al mundo real, es necesario desarrollar un algoritmo que genere las trayectorias que el robot móvil debe seguir; este algoritmo incluye un modulador de velocidad dependiente de la curvatura de la trayectoria.

El algoritmo generador de trayectorias recibe como entrada las coordenadas de la detección ya proyectadas al mundo real, luego estas coordenadas se interpolan estableciendo la trayectoria deseada, posteriormente a esta trayectoria se le aplica la ecuación de la curvatura, la cual está dada por:

$$\kappa = \frac{|y''|}{[1 + (y')^2]^{3/2}}$$

donde una κ grande implica una curva grande y una κ pequeña cercana a cero implica una línea recta; esta ecuación nos permite parametrizar la ruta y así modificar la velocidad de ejecución.

6.3 Algoritmo de exploración

En el caso particular en el que no se obtengan resultados de la detección, se implementa un algoritmo de búsqueda de ruta, este algoritmo genera una trayectoria deseada con dos variantes. La primera vez que entra en marcha este algoritmo genera un giro de 30° en sentido antihorario, la segunda vez que el algoritmo entra en marcha repite la generación del giro de 30° en sentido antihorario, la tercera y última vez que entra en marcha este algoritmo genera un giro de 120° en sentido horario, para después repetir la secuencia. Este algoritmo se puede modificar según las necesidades de la tarea, incluso agregando traslaciones.

6.4 Algoritmo de control

Como elemento final se requiere de un algoritmo de control, en este caso se opta por implementar el esquema CNA pre-

sentado en el capítulo 3; este esquema utiliza como referencia las trayectorias anteriormente generadas.

6.5 Vehículo detector de ruta en simulación

Ya que se han descrito los algoritmos que integran al vehículo detector de ruta, se procede a implementarlo en simulación. Para la aplicación de la técnica de transferencia de aprendizaje se ha seleccionado como base la red Resnet50, la cual cuenta con 150 capas. Los recorridos para la obtención de las imágenes de entrenamiento se realizan utilizando un mando o *joystick* compatible y una herramienta de simulación 3D que es también parte del *Toolbox* de conducción autónoma ya mencionado. En las figuras 6.1-6.2 se muestran las capturas de pantalla del mundo virtual y de la aplicación de etiquetado respectivamente. El entrenamiento del detector de ruta se realiza con 390 imágenes.

La resolución de las imágenes obtenidas es de 720×1280 pixeles, misma resolución con la cual se entrena el detector. El siguiente vídeo muestra los resultados del detector de ruta en simulación:

- https://youtu.be/KN26kuVnQTE.

Al utilizar un mundo virtual no es necesario estimar los parámetros intrínsecos de la cámara, ya que el bloque de cámara virtual los proporciona. La figura 6.3 muestra la propuesta del diagrama de bloques del vehículo detector de ruta en simulación, el cual incluye algoritmos de control y navegación. Los resultados en simulación para el vehículo detector de ruta se muestran en el siguiente vídeo:

- https://youtu.be/yBpuw_efmrA.

En el vídeo anterior se muestran las imágenes capturadas en vista superior por una cámara fija, las imágenes capturadas en vista superior por una cámara fija al cuerpo del vehículo y las imágenes captadas por la cámara a bordo y procesadas por el detector de ruta.

Figura 6.1: Captura de pantalla del mundo virtual.

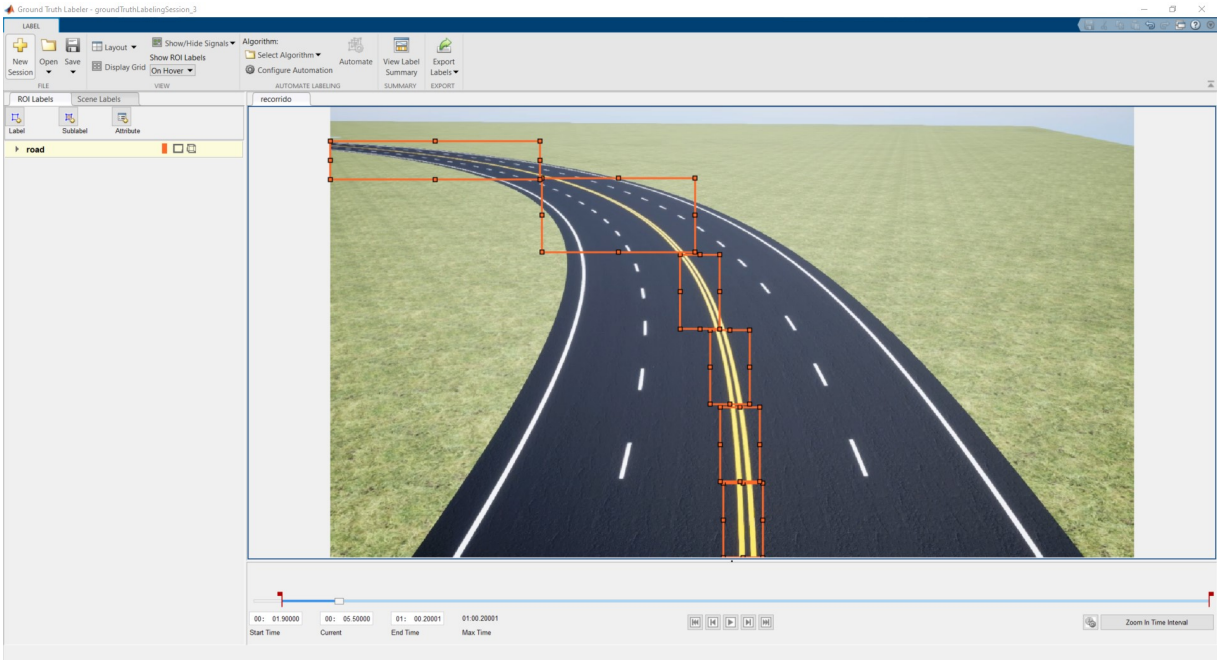

Figura 6.2: Captura de pantalla del etiquetado en simulación.

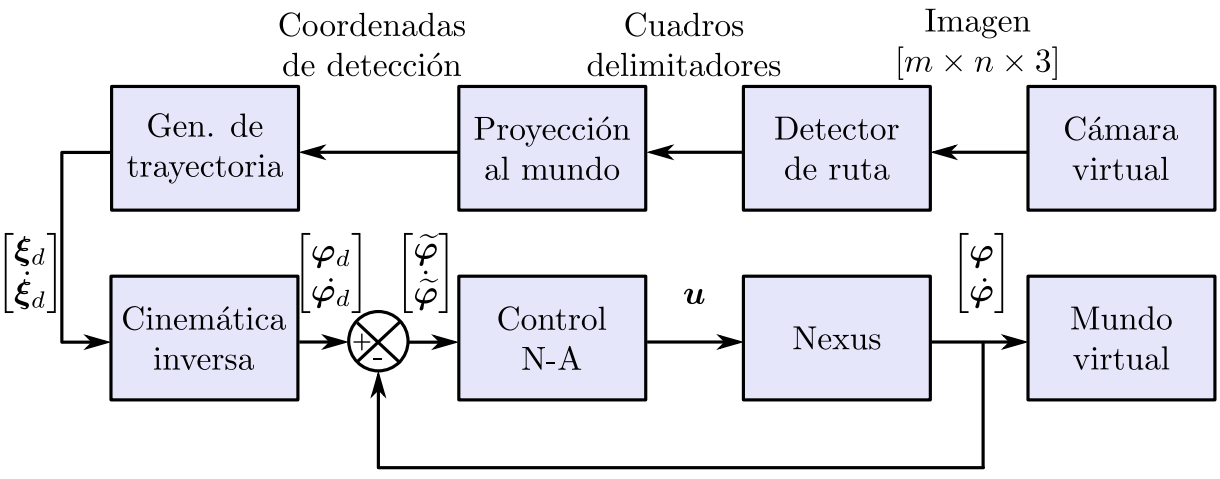

Figura 6.3: Diagrama de bloques del vehículo detector de ruta.

6.6 Vehículo detector de ruta en tiempo real

A continuación, se procede a implementar el vehículo detector de ruta en tiempo real. Para la aplicación de la técnica de transferencia de aprendizaje se ha seleccionado como base la red TinyYOLOv2, descrita en el capítulo anterior. Los recorridos para la obtención de las imágenes de entrenamiento se realizan utilizando un mando o *joystick* compatible, módulos de comunicación inalámbrica (ver apéndice A.3) y el robot Nexus (ver apéndice A.1). En la figura 6.4 se muestra la captura de pantalla de la aplicación de etiquetado. El entrenamiento del detector de ruta se realiza con 806 imágenes.

La resolución de las imágenes obtenidas es de 720×1280 pixeles, misma resolución con la cual se entrena el detector. El circuito hecho sobre neopreno se muestra en el apéndice A.6. El siguiente vídeo muestra los resultados del detector de ruta en tiempo real:

- https://youtu.be/JmG_7r1vwxA.

Los procesos y cálculos son realizados a bordo por el kit de visión UP Squared X, el cual se describe en el apéndice A.5. La figura 6.5 muestra la propuesta del diagrama de flujo del algoritmo de navegación del vehículo detector de ruta en tiempo real. Los resultados en tiempo real para el vehículo detector de ruta se muestran en el siguiente vídeo:

- https://youtu.be/r11t7QTomu8.

En el vídeo anterior se muestra tanto las imágenes captadas por una cámara fija, como las captadas por la cámara a bordo y procesadas por el detector de ruta, además se muestran las evoluciones temporales de las velocidades y los voltajes de los motores.

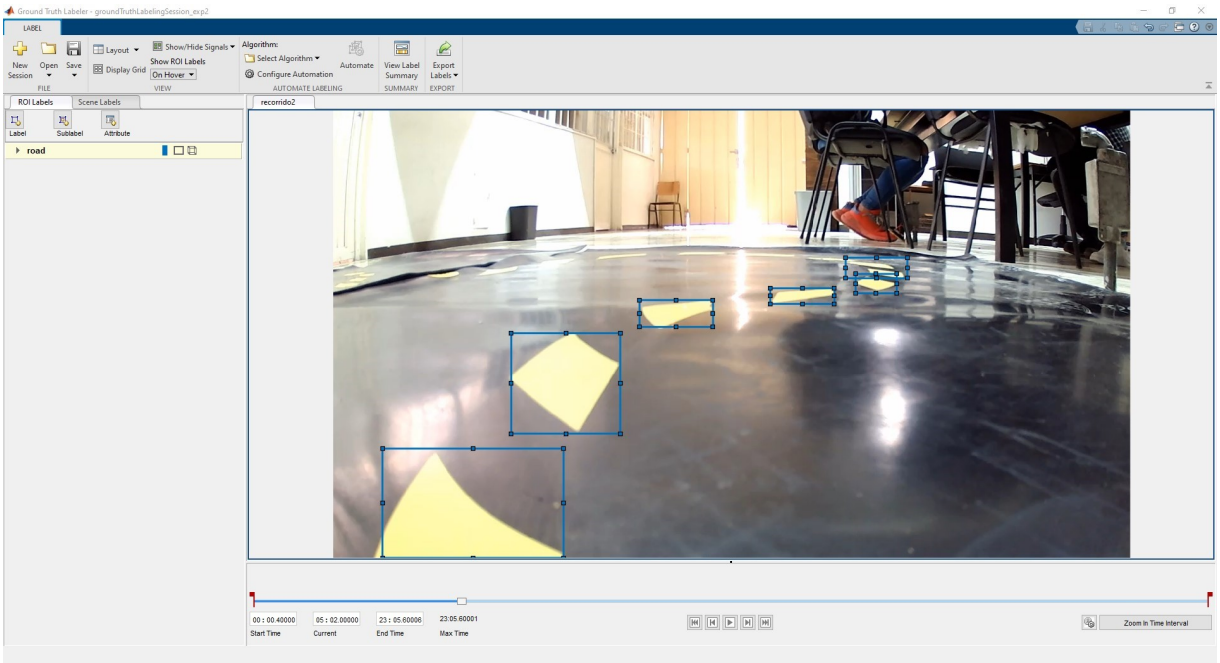

Figura 6.4: Captura de pantalla del etiquetado en tiempo real.

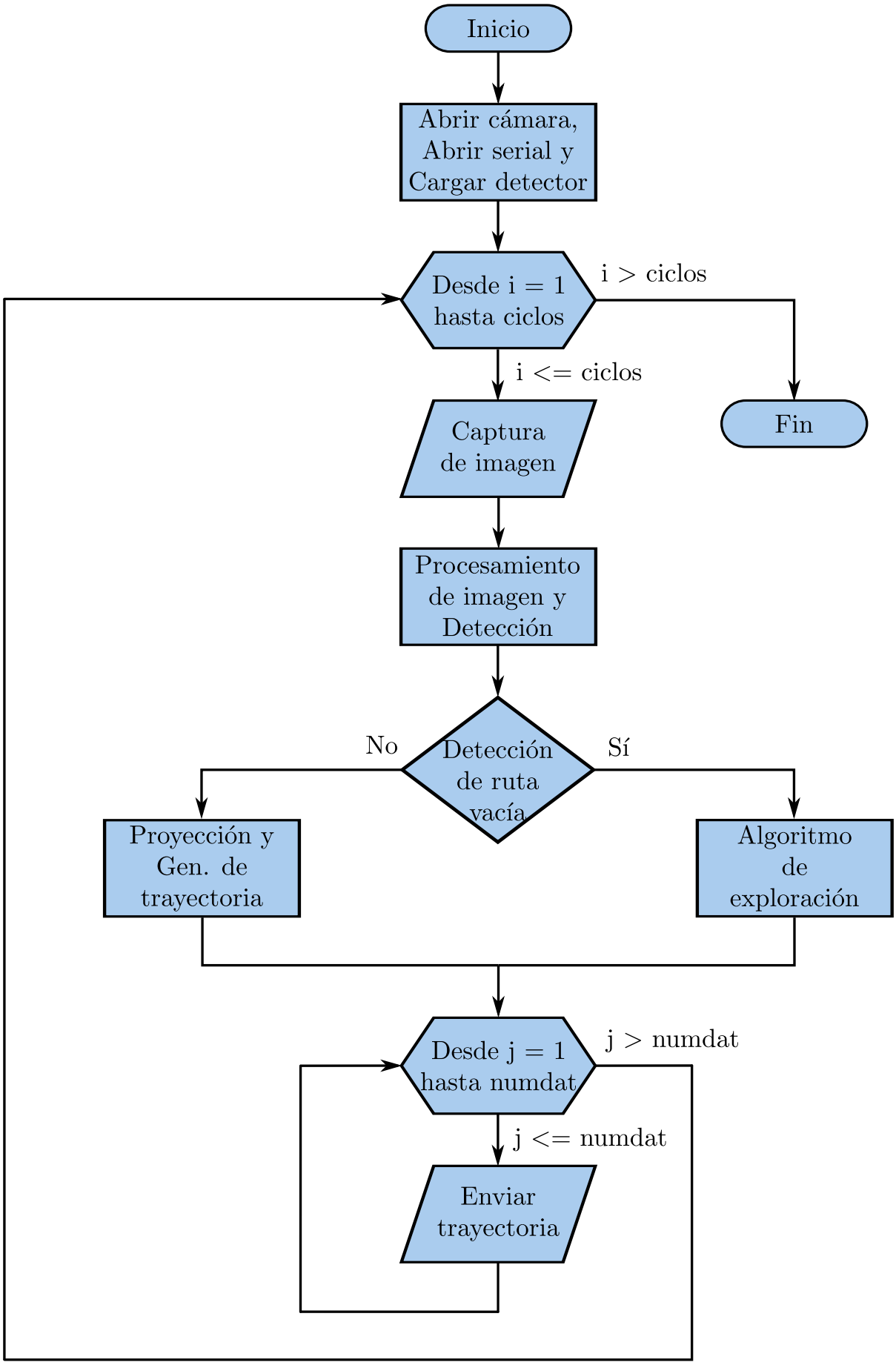

Figura 6.5: Diagrama de flujo del vehículo detector de ruta en tiempo real.

6.7 Vehículo detector de ruta y obstáculos en tiempo real

Tomando en cuenta que se ha desarrollado un sistema de detección y ejecución de ruta en tiempo real, es posible utilizar esta plataforma como base para agregar recursos de detección de obstáculos; estos recursos permiten que la plataforma tome acciones al detectar dichos obstáculos.

Los recursos seleccionados para la detección de obstáculos son la RNC TinyYOLOv2 y el sensor RPLIDAR A2M8, las características de este último se detallan en el apéndice A.4. Se eligió la RNC TinyYOLOv2 por ser un modelo especializado en sistemas embebidos o en tiempo real, además de la utilidad de las clases que utiliza. Para poder integrar la RNC TinyYOLOv2 al sistema, se destacan las clases útiles y se discriminan las que entorpecen el funcionamiento del sistema debido al lugar donde se realizan los experimentos. La condición de detección para la RNC es que arroje resultados sobre las clases destacadas. En lo que concierne el sensor RPLIDAR A2M8 se eligió por sus buenas prestaciones, su protocolo de comunicación y a su relativo bajo costo en comparación con otros sensores. Las condiciones de detección para el sensor Lidar son que los objetos a detectar se encuentren en un rango menor o igual a 1.5 metros y menor o igual a ±15°, ambos con respecto al eje x del marco del robot. En la figura 6.6 se muestra la detección del Lidar en un ciclo.

Los experimentos, además de ejecutarse en un circuito, también se ejecuta en una ruta recta, la cual se muestra en el apéndice A.6. La figura 6.7 muestra la propuesta del diagrama de flujo del algoritmo de navegación del vehículo detector de ruta y obstáculos en tiempo real. La figura 6.8 muestra el diagrama de conexión del vehículo detector de ruta y obstáculos en tiempo real. Los resultados en tiempo real para el vehículo detector de ruta y obstáculos se muestran en los siguientes vídeos:

- https://youtu.be/EGm0CwU7vxQ
- https://youtu.be/xKbFb4r9Smc.

En los vídeos anteriores se muestra tanto las imágenes captadas por una cámara fija, como las captadas por la cámara a

bordo y procesadas por ambos detectores, además se muestran las evoluciones temporales de las velocidades y los voltajes de los motores.

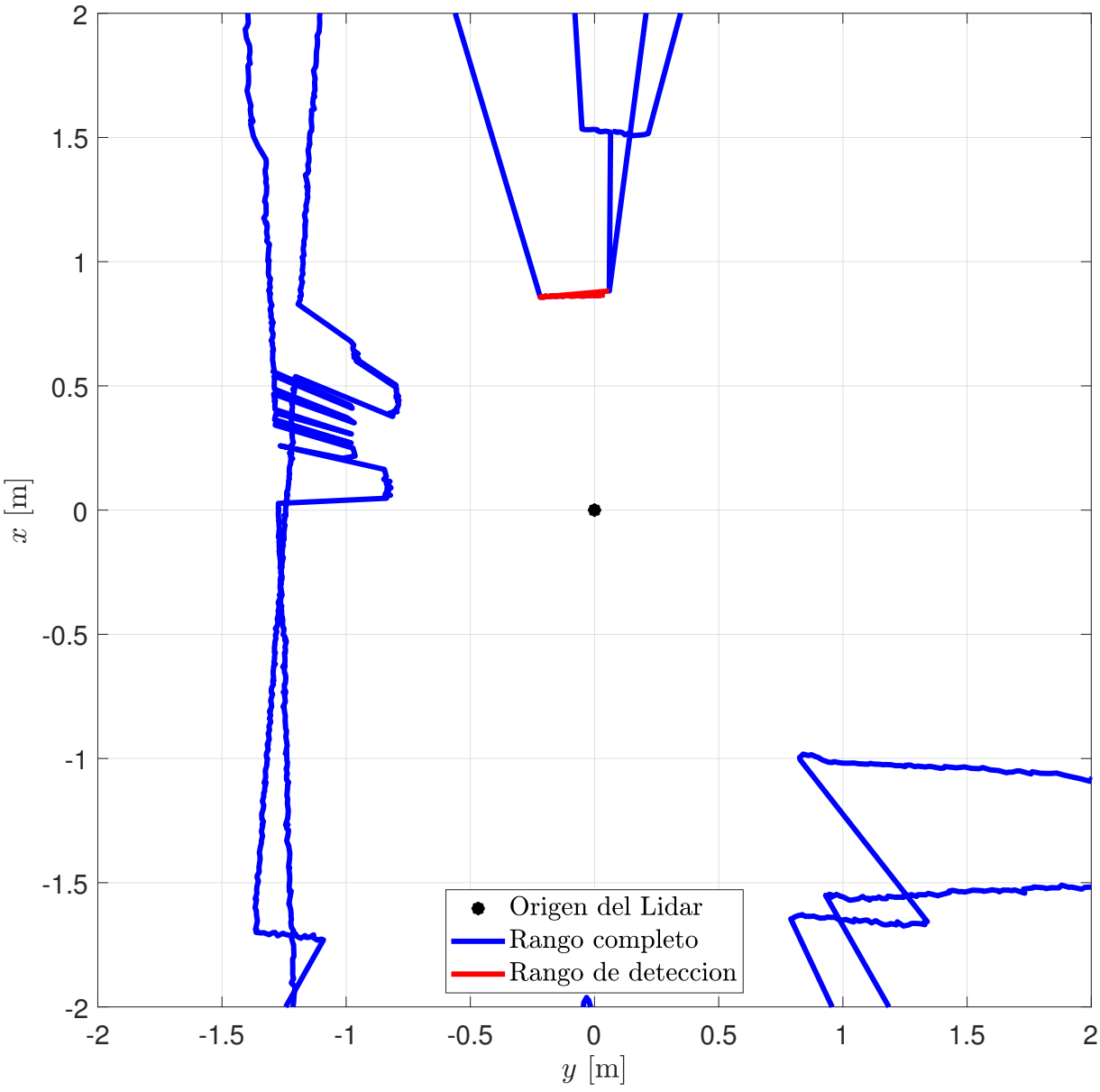

Figura 6.6: Detección del Lidar en un ciclo.

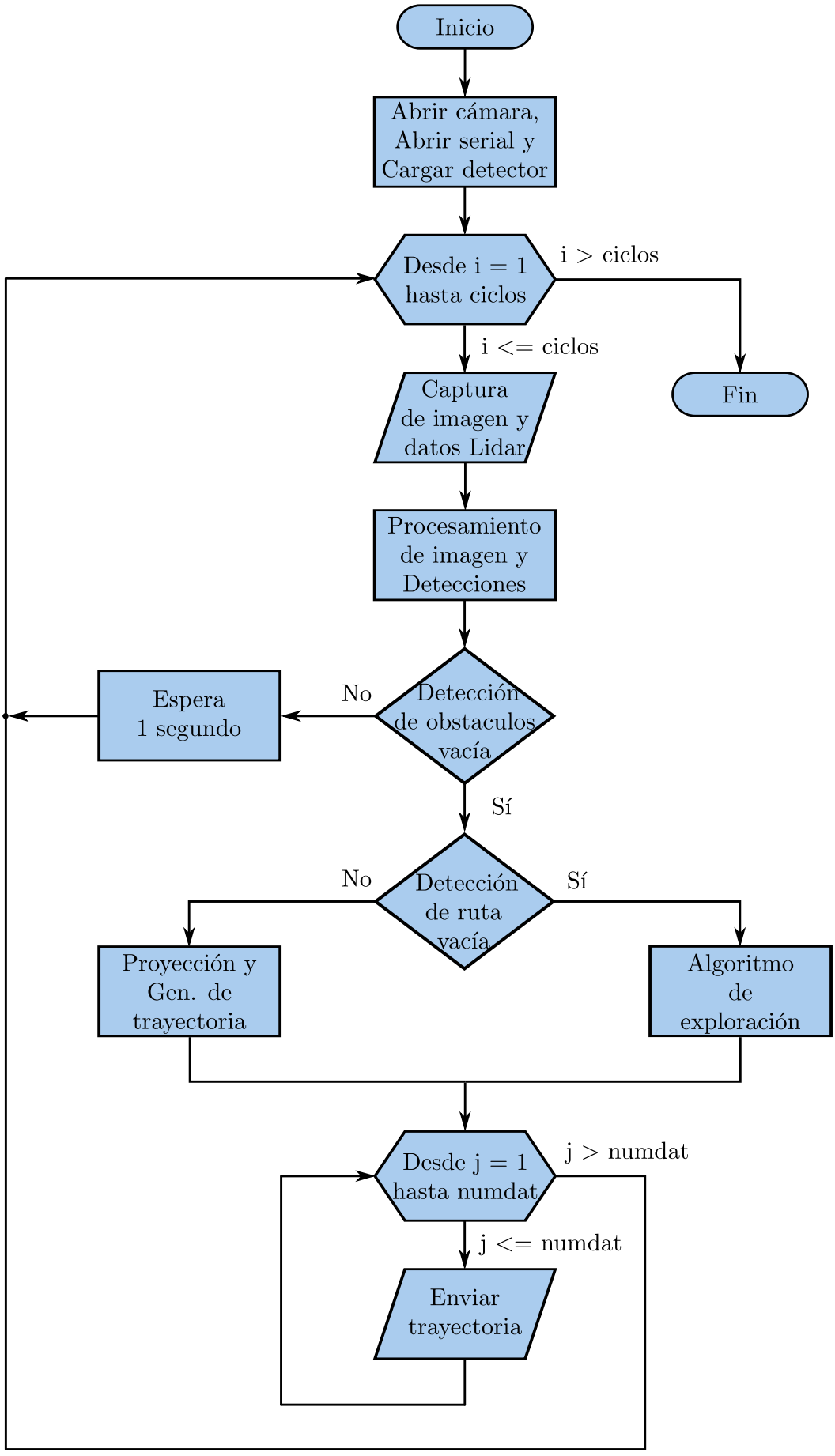

Figura 6.7: Diagrama de flujo del vehículo detector de ruta en tiempo real.

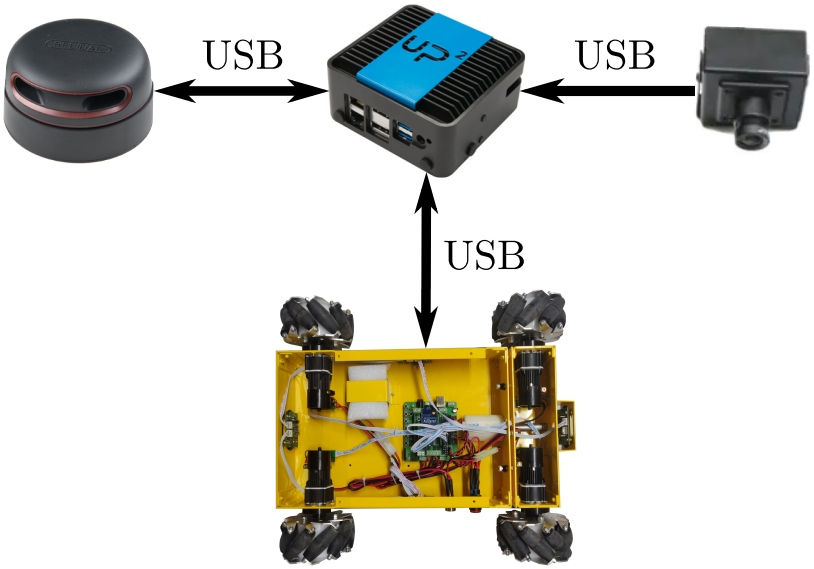

Figura 6.8: Diagrama de conexión del vehículo detector de ruta en tiempo real.

6.8 Conclusiones y observaciones

Antes de proceder a entrenar los detectores neuronales, la arquitectura YOLO requiere del cálculo de los cuadros de referencia (*anchor boxes*), los cuales determinan los rangos de tamaño de los cuadros delimitadores. Es posible calcular los cuadros de referencia a partir de los datos de entrenamiento; la función de Matlab que realiza este cálculo es:

```
1 estimateAnchoreBoxes()
```

Al aplicar el modelo TunyYOLOv2 se implementa una condición simple de detección, la cual consiste en la existencia de objetos de las clases destacadas, pero como ya se tiene el modelo de la cámara es posible agregar una condición de detección sujeta a la distancia del objeto con respecto al marco móvil del robot Σ_M.

El algoritmo de exploración puede ser modificado según las necesidades de la ruta que se desea recorrer, incluyendo cualquier combinación de movimientos de orientación y traslación posibles.

El vehículo detector de ruta y objetos hace uso de dos clases de modelos neuronales, el primero se encuentra en el control de velocidad (esquema CNA) y el segundo se encuentra en el detector de ruta y en el detector de objetos (TinyYOLOv2).

Una buena alternativa para obtener los parámetros de la cámara, es usando la aplicación *Camera Calibration* de Matlab. Para estimar los parámetros de la cámara, la aplicación

requiere de múltiples capturas de un patrón conocido. Este proceso se puede realizar de igual manera utilizando la función de Matlab:

```
1 estimateCameraParameters()
```

Conclusiones 7

7.1 Conclusiones generales

En este trabajo se propone un nuevo análisis de estabilidad para el esquema CNA que difiere al mostrado en [5], el cual permite hacer conclusiones directamente sobre $\widetilde{q}$ y $\dot{\widetilde{q}}$. Se propone el esquema CN-AM, este es una modificación del esquema CNA, entre las modificaciones propuestas se incluyen el cambio de la pendiente en la función tangente hiperbólica y el uso de la función de potencia signada. Se propone un nuevo esquema de control N-A llamado CPD-CNA, dicho esquema utiliza la función de potencia signada y cuentan con actualización de pesos en línea.

[5]: Lewis y col. (1998), *Neural Network Control of Robot Manipulators and Nonlinear Systems*

Además, se presentan experimentos en tiempo real de los esquemas presentados, tanto en el robot manipulador de 2 g.d.l. CICESE como en el robot móvil omnidireccional Nexus. En general, los resultados mostraron que los controladores neuronales de una capa diseñados consiguen un buen desempeño sin mostrar excesivo ruido en las señales de control. El esquema propuesto CPD-CNA fue capaz de compensar la fricción, los parámetros inciertos del sistema y tuvo un comportamiento robusto frente a perturbaciones externas fuertes.

Se desarrolló un sistema de navegación y detección de obstáculos, dicho sistema se implementó tanto en simulación como en tiempo real. En cuanto a los recursos de aprendizaje automático utilizados en la plataforma, se aplicaron técnicas de control neuronal, se implementó el detector de objetos del estado del arte llamado TinnyYOLOv2 y se entrenó e implementó una RN profunda para detectar y ejecutar una ruta determinada.

7.2 Producción académica

Durante el transcurso de este trabajo se produjeron las siguientes publicaciones:

- Sergio López, Miguel Llama, Ramón García-Hernández y Víctor Santibáñez. "PD with neuro-adaptive compensation control using the signed power function". International Journal of Control (2022) [63].
- Sergio López, Miguel Llama y Ramón García-Hernández. "Controlador PD con compensación Neuro-Adaptable aplicado a la dinámica de un RMR Omnidireccional". Memorias del Congreso Nacional de Control Automático (2019) [64].

7.3 Recomendaciones

Uno de los aspectos que requiere mejora en el vehículo detector de ruta es la optimización de los tiempos de inferencia y los tiempos de guardado y procesamiento de datos. La mini PC encargada de los cálculos en tiempo real cuenta con una unidad de procesamiento especializado en ejecutar RNs en tiempo real (ver apéndice A.5), pero desafortunadamante no hay documentación del uso de esta unidad con Matlab y no fue posible utilizarlo; una alternativa a esto es migrar la plataforma a un entorno que admita esta unidad de procesamiento especializada, lo cual es factible gracias al formato ONNX. Una forma de disminuir los tiempos de entrenamiento e inferencia de los detectores neuronales es disminuyendo la resolución de los datos de entrenamiento. Como alternativa para mejorar todos los tiempos de ejecución se propone mejorar el equipo donde se realizan todos los cálculos, de preferencia elegir un equipo con tarjeta gráfica dedicada incluida.

Un factor importante que afecta el desempeño del vehículo detector de ruta es la iluminación, tanto los cambios en la luz natural como la mala iluminación de la luz de interior. Para evitar los malos funcionamientos por cuestiones de iluminación se debe modificar la sensibilidad de detección de la red y se debe garantizar una buena iluminación de interior; es posible implementar un algoritmo automático para modificar la sensibilidad del detector en función de la iluminación.

A

Elementos de la plataforma experimental

Para lograr implementar un sistema de navegación y detección de obstáculos se requiere integrar una plataforma con distintos tipos de sensores y unidades de procesamiento. En este apéndice se describen los elementos, equipos y herramientas utilizadas para el desarrollo de esta plataforma experimental.

A.1 Robot móvil omnidireccional Nexus

Uno de los componentes principales para el desarrollo de esta plataforma es el robot móvil omnidireccional Nexus con ruedas Mecanum (ver figura A.1). EL robot móvil Nexus cuenta con una placa de arduino duemilanove con etapa de potencia. Cada motor del robot cuenta con un tren de engranes que puede suministrar 0.8 [N-m] de par constante y hasta 1.2 [N-m] de par temporal. La hoja de datos del motor se encuentra en el enlace https://www.faulhaber.com/EN_2342_CR_-DFF.pdf, mientras que la hoja de datos del tren de engranes se encuentra en el enlace https://www.faulhaber.com/EN_-26A_DFF.pdf.

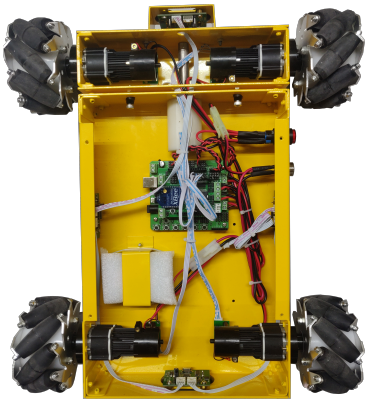

Figura A.1: Robot móvil Nexus.

A.2 Sistema de navegación inercial

La central inercial SparkFun 9DoF Razor (ver figura A.2) combina un microcontrolador SAMD21 con un sensor MPU-9250 de 9 g.d.l. para crear un sistema de navegación inercial reprogramable y multipropósito. El sensor MPU-9250 cuenta con: un acelerómetro, un giroscopio y un magnetómetro, cada uno de tres ejes. Es posible programar este sistema para transmitir ángulos de Euler a través de un puerto serie.

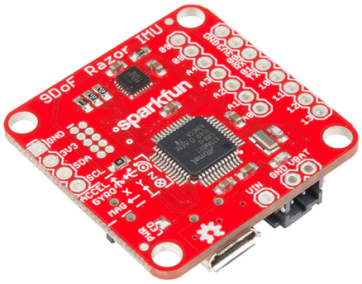

Figura A.2: Central inercial SparkFun 9DoF Razor.

A.3 Módulo de comunicación X-Bee S1

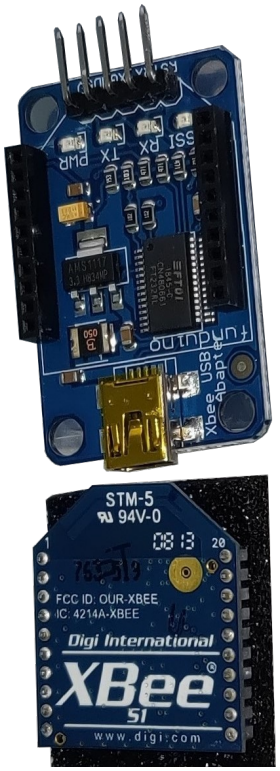

Figura A.3: Módulo explorador para X-Bee y módulo X-Bee S1.

Para obtener las capturas de la ruta para el entrenamiento en tiempo real se requiere un módulo de comunicación inalámbrica, en este caso se selecciono el módulo X-Bee S1; dicho módulo se muestra en la figura A.3. La velocidad de trasmisión más alta sin perdida de datos es de 57600 baudios. Este módulo presenta problemas de desconexión y requiere retardos de aproximadamente 15 milisegundos al enviar datos. La hoja de datos del módulo X-Bee S1 se encuentra en el siguiente enlace https://www.digi.com/resources/library/datasheets/ds_xbeemultipointmodules.

A.4 Sensor RPLIDAR A2M8

Uno de los sensores más importante para la plataforma es el LIDAR (*laser imaging, detection, and ranging*), puesto que permite determinar la distancia a la cual se encuentran ciertos objetos alrededor del mismo en un rango determinado. Se eligió el sensor RPLIDAR A2M8 (ver figura A.4) por su accesibilidad y facilidad de manejo, este sensor puede realizar un escaneo 2D de 360° con un rango de trabajo de entre 0.15 a 12 [m]. La documentación del sensor RPLIDAR A2M8 se encuentra en el enlace https://bucket-download.slamtec.com/LD208_-SLAMTEC_rplidar_datasheet_A2M8_v2.6_en.pdf, mientras que su funcionamiento se pueden ver en el siguiente vídeo: https://youtu.be/rr9n1sLadyM.

Figura A.4: Sensor RPLIDAR A2M8.

A.5 Kit UP Squared AI Vision X

Para agilizar el proceso de interpretación de imágenes se utiliza el kit de desarrollo llamado Up Squared AI Vision X (ver figura A.5), éste cuenta con una mini PC (llamada Up Squared), una cámara USB y una unidad de procesamiento de visión Intel Movidius Myriad X, siendo este último un módulo especializado en implementar RNCs en tiempo real. La mini PC cuenta con un procesador Intel Atom E3940, 8 GB de ram, 64 GB de almacenamiento y Ubuntu 20 LTS. La cámara incluida tiene con resolución máxima de 1920×1080p a 30 cuadros por segundo. La hoja de especificaciones de la mini PC se encuentra en el siguiente enlace: https://up-board.org/wp-content/uploads/2021/06/UP-Squared-Datasheet-v2.pdf.

Figura A.5: Kit UP Squared AI Vision X.

A.6 Superficie de neopreno

Con el fin de aumentar la tracción de los experimentos en tiempo real con el robot Nexus, se utiliza una superficie de neopreno; esta superficie consiste de 3 pliegos de 3 por 1 [m] con un grosor de 3 milímetros. Tanto el circuito como la ruta recta se instalan sobre estos pliegos de neopreno. En la figura A.6 se muestra el circuito, mientras que en la figura A.7 se muestra la ruta recta que el vehículo debe recorrer.

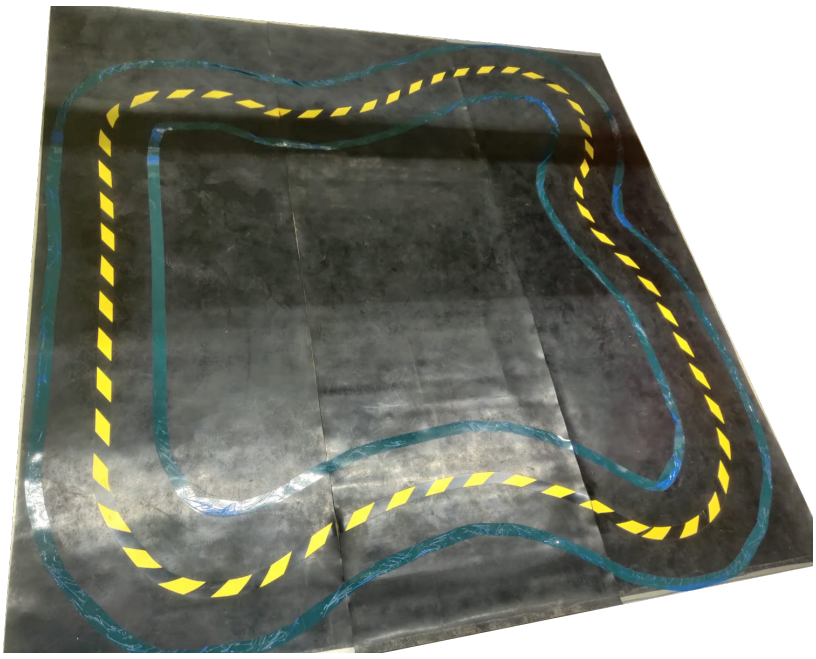

Figura A.6: Circuito para pruebas en tiempo real.

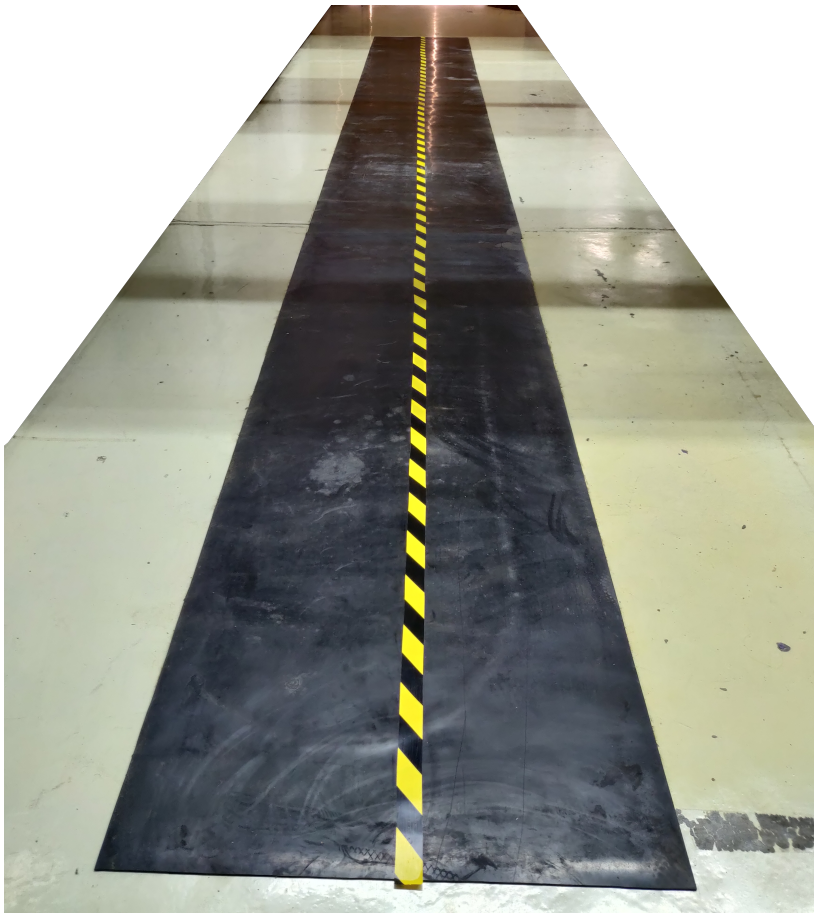

Figura A.7: Ruta recta para pruebas en tiempo real.

B

Códigos fuente

En este apéndice se presentan los códigos fuente para el vehículo detector de ruta y obstáculos en tiempo real, tanto el código principal que corre en el Up Squared, como el código de control que corre en el microcontrolador del Nexus.

B.1 Código principal (Matlab)

```
%%%%%Configuraciones y declaraciones%%%%%
%%%Camara%%%
camera=webcam(1);

%%%Parametros de la camara%%%
focalLength    = [928, 929];
principalPoint = [658,379];
imageSize      = [720, 1280];
height         = 0.071;
pitch          = 13.8;
roll           =  -1.25;
%radialDistortion =
   [-0.435419839396045,0.205646062077845];
camIntrinsics = cameraIntrinsics(focalLength,
   principalPoint, imageSize);%,'
   RadialDistortion', radialDistortion
sensor        = monoCamera(camIntrinsics,
   height, 'Pitch', pitch, 'Roll',roll);

%%%Puertos seriales%%%
s=serialport('/dev/ttyUSB0',57600);%Serial de
   Nexus %115200 57600 38400 19200 9600
s2=serialport('/dev/ttyUSB1',115200);%Serial
   de lidar
%/dev/ttyS5%Serial de IMU
%serialportlist

%%%Cambiar path y carga de archivos%%%
```

```
23 cd '/home/upsquared/Documents/MATLAB/Yolo'
24 load('DetectorTinyYoloV2.mat')
25 load('net.mat')
26
27 %%%Declaraciones de variables y envio de
   inicio%%%
28 ciclos=25;
29 hv=03.7;%Horizonte visible e metros
30 %pv=50;% Puntos de velocidad, mas puntos mas
   lento
31 thho=0.52;%0.5225;
32 vma=[0;0;0;0];
33 tma=zeros([9,1]);
34 ada=[0;0];
35 %%%%%Inicio del loop%%%%%
36 for i=1: ciclos
37 %pause(0.7)
38 [f,ad]=lidar(s2);ada=cat(2,ada,ad);
39 I=snapshot(camera);
40 I=imrotate(I,180);
41 I=imresize(I,[720 1280]);%I=imresize(I,[234
   416]);
42 [bb, sc, lb]=yolov2Detector.detect(I,'
   Threshold',0.25);%isempty(bb)0.25->0.5
43 if isempty(find(lb=='person'|lb=='bicycle'|lb
   =='dog', 1))&&f
44 [bboxes, scores]=detectorTinyYolo2.detect(I,'
   Threshold',thho);
45     if ~isempty(bboxes)
46     impoints=[bboxes(:,1)+bboxes(:,3)/2,
   bboxes(:,2)+bboxes(:,4)/2];
47         di=insertObjectAnnotation(I,'circle',[
   impoints,4*ones(size(bboxes,1),1)],scores,'
   FontSize',30','LineWidth',7);
48         filename=['Muestras','/',sprintf('file
   %02d.png',i)];
49         imwrite(di,filename,'png');
50         % di=insertObjectAnnotation(I,'
   rectangle',bboxes,scores,'FontSize',30','
   LineWidth',7);
51         % imshow(di)
52     vehiclePoints=imageToVehicle(sensor,
   impoints);
53     vehiclePoints=vehiclePoints(vehiclePoints
   (:,1)<hv,:);
54     vehiclePoints(:,1)=vehiclePoints(:,1)
   +0.21;
```

```
55      vehiclePoints=cat(1,vehiclePoints,[0, 0]);
56      % th0 = leer IMU en inicio de ciclo
57      if ~isempty(find(vehiclePoints,1)) %%%Hay
        datos bajo el horizonte sigue la ruta%%%
58      p=polyfit(vehiclePoints(:,1),vehiclePoints
        (:,2),2);
59      %%%%%
60      numpoints=round(abs((vehiclePoints(1,1)-
        vehiclePoints(end,1)))*50)+1;
61      x=linspace(vehiclePoints(end,1),
        vehiclePoints(1,1),numpoints);
62      y=polyval(p,x);
63      yp=gradient(y,0.1);
64      k=max(abs(gradient(yp,0.1))./(1+yp.^2)
        .^(3/2));
65      %pv=30*k/0.35+35;
66      pv=20*k/0.35+30;
67      %%%%%
68      numpoints=round(abs((vehiclePoints(1,1)-
        vehiclePoints(end,1)))*pv)+1;
69      x=linspace(vehiclePoints(end,1),
        vehiclePoints(1,1),numpoints);
70      y=polyval(p,x);
71      xp=gradient(x,0.1);yp=gradient(y,0.1);
72      T=xp./sqrt(xp.^2+yp.^2);
73      th=sign(yp).*acos(T);
74      sth=round(size(th,2)/3);
75      th(1:sth)=linspace(0, th(sth),sth);
76      thp=gradient(th,0.1);
77          %imagePoints=vehicleToImage(sensor,[x
        '-0.21,y']);
78          %di=insertShape(I,'circle',[
        imagePoints, 4*ones(size(imagePoints,1),1)
        ],'LineWidth',7);
79          %imshow(di)
80      %se usa theta real menos el offset de
        inicio de ciclo,
81      %o se usa theta deseada
82          xpr=(xp.*cos(th) + yp.*sin(th))
        .*(255*2);
83          ypr=-(yp.*cos(th) - xp.*sin(th))
        .*(255*3);
84          thpr=-thp.*(170*1);
85          [g,tm]=move(xpr,ypr,thpr,s);
86          vma=cat(2,vma,[xp;yp;thp;th]);
87          tma=cat(2,tma,tm);
```

```
88      else %%%No hay datos bajo el horizonte avanza%%%
89          x=linspace(0,hv-0.1,round((hv-0.1)*pv)+1);
90          xp=gradient(x,0.1)*(255*2);
91          yp=-zeros(size(xp));
92          thp=yp;
93          [g,tm]=move(xp,yp,thp,s);
94          vma=cat(2,vma,[xp;yp;thp;th]);
95          tma=cat(2,tma,tm);
96       end
97       %
98       else %%%En caso de no encontrar la ruta girar%%%
99       filename=['Muestras','/',sprintf('file%02d.png',i)];
100      imwrite(I,filename,'png');
101      th=0:.1:pi/6;
102      thp=gradient(th,0.1);
103      xpr=zeros(size(th));
104      ypr=-xpr;
105      thpr=-thp.*(170);
106      [g,tm]=move(xpr,ypr,thpr,s);
107      vma=cat(2,vma,[xpr;ypr;thp;th]);
108      tma=cat(2,tma,tm);
109      end % end de el ~isempty
110  else
111      di=insertObjectAnnotation(I,'rectangle',bb,lb,'FontSize',30','LineWidth',7,'Color','red');
112      [bboxes, scores]=detectorTinyYolo2.detect(I,'Threshold',thho);
113      if ~isempty(bboxes)
114      impoints=[bboxes(:,1)+bboxes(:,3)/2,bboxes(:,2)+bboxes(:,4)/2];
115      di=insertObjectAnnotation(di,'circle',[impoints,4*ones(size(bboxes,1),1)],scores,'FontSize',30','LineWidth',7);
116      end
117      %imshow(di);
118      filename=['Muestras','/',sprintf('file%02d.png',i)];
119      imwrite(di,filename,'png');
120      pause(1)
121  end
122  end % end del for ciclos
123  % clear camera
```

```
% clear s
```

B.2 Código de control (Arduino)

```
//Para path_exp_6.m
#include <EnableInterrupt.h>
//Llanta 1
#define PWM1 10
#define DIR1 7 //HIGH: POSITIVO, LOW:
   NEGATIVO
#define APIN1 5
#define BPIN1 6
//Llanta 2
#define PWM2 3
#define DIR2 2 //LOW: POSITIVO, HIGH:
   NEGATIVO
#define APIN2 4
#define BPIN2 18
//Llanta3
#define PWM3 11
#define DIR3 12//LOW: POSITIVO, HIGH:
   NEGATIVO
#define APIN3 14
#define BPIN3 15
//Llanta4
#define PWM4 9
#define DIR4 8 //HIGH: POSITIVO, LOW:
   NEGATIVO
#define APIN4 16
#define BPIN4 17

//Declaracion
volatile int16_t intCount1 = 0;
volatile int16_t intCount2 = 0;
volatile int16_t intCount3 = 0;
volatile int16_t intCount4 = 0;
float vel[] = {0, 0, 0, 0};
float veld[] = {9.5, 9.5, 9.5, 9.5};
float velf[] = {0, 0, 0, 0};
float error[4], error_vel[4];
float qpd[]={0.2885, 0, -0.2885};
//float error0[] = {0, 0, 0, 0};
//float dere[] = {0, 0, 0, 0};
```

```
36 int16_t tao[] = {0, 0, 0, 0};
37 float kv = 5, kp=50, kd=0;//kp = 25, kd =
     0, ki=50;
38 float l1=0.1524, l2=0.1505, ra=0.05, G =
     0;
39 uint8_t i;
40 float r[4], phi[4][4], f[4], w[4][4];
41 unsigned long t, t0=0, dt;
42 int palabra[] = {0, 0, 0, 0};
43 int ledState = LOW, c=0;
44 //Funciones de interrupcion
45 void intFunction1() {
46   int b = digitalRead(BPIN1);
47   if(b > 0)       intCount1++;
48   else       intCount1--;
49 }
50 void intFunction2() {
51   int b = digitalRead(BPIN2);
52   if(b > 0)       intCount2--;
53   else       intCount2++;
54 }
55 void intFunction3() {
56   int b = digitalRead(BPIN3);
57   if(b > 0)       intCount3--;
58   else       intCount3++;
59 }
60 void intFunction4() {
61   int b = digitalRead(BPIN4);
62   if(b > 0)       intCount4++;
63   else       intCount4--;
64 }
65
66 void setup() {
67   Serial.begin(57600);//Velocidad de
      comunicacion 115200 57600 38400
68   TCCR1B=TCCR1B&0xf8|0x01; // Pin9,Pin10
      PWM 31250Hz
69   TCCR2B=TCCR2B&0xf8|0x01; // Pin3,Pin11
      PWM 31250Hz
70   pinMode(APIN1, INPUT_PULLUP);//
      Activacion de reistencias pull-up
71   pinMode(APIN2, INPUT_PULLUP);
72   pinMode(APIN3, INPUT_PULLUP);
73   pinMode(APIN4, INPUT_PULLUP);
```

```
74    pinMode(BPIN1, INPUT_PULLUP);//
       Activacion de reistencias pull-up
75    pinMode(BPIN2, INPUT_PULLUP);
76    pinMode(BPIN3, INPUT_PULLUP);
77    pinMode(BPIN4, INPUT_PULLUP);
78    enableInterrupt(APIN1, intFunction1,
       RISING);//Asignacion de interrupciones
       //CHANGE
79    enableInterrupt(APIN2, intFunction2,
       RISING);
80    enableInterrupt(APIN3, intFunction3,
       RISING);
81    enableInterrupt(APIN4, intFunction4,
       RISING);
82  }
83
84  void loop() {
85  //Inicio del loop que se realiza cada
       tiempo de muestreo
86    if(Serial.available() >= 7 && Serial.
       read() == 'L' && Serial.read() == 'M'
       && Serial.read() == 'C'){
87    t=millis();//para calcular tiempos
88  //    dt=t-t0;
89  //    t0=t;
90  //    c++;
91    noInterrupts();
92  //Calculo de velocidades, formula => (
       Cuentas*2*pi)/(ranuras*relacion de
       engrane*tiempo de muestreo)=>Radianes/
       segundo
93    vel[0] = (float)intCount1 / (1.22231*10)
       ;//12.2231
94    vel[1] = (float)intCount2 / (1.22231*10)
       ;
95    vel[2] = (float)intCount3 / (1.22231*10)
       ;
96    vel[3] = (float)intCount4 / (1.22231*10)
       ;
97    intCount1 = 0;//resetear las cuentas
       para el siguiente calculo
98    intCount2 = 0;
99    intCount3 = 0;
100   intCount4 = 0;
```

```
101   interrupts();
102 //Filtro de primer orden para la medicion
      de velocidad
103     velf[0] += 0.1*5*(vel[0] - velf[0]);
104     velf[1] += 0.1*5*(vel[1] - velf[1]);
105     velf[2] += 0.1*5*(vel[2] - velf[2]);
106     velf[3] += 0.1*5*(vel[3] - velf[3]);
107 //Calculo de referencias
108     palabra[0] = Serial.read();
109     palabra[1] = Serial.read();
110     palabra[2] = Serial.read();
111     palabra[3] = Serial.read();
112     qpd[0] =  (float)palabra[0]/(255*2);//
      3
113     qpd[1] =  (float)palabra[1]/(255*3);
114     qpd[2] =  (float)palabra[2]/(170);//
      255
115     if(!bitRead(palabra[3],0))      qpd[0]
       = -qpd[0];
116     if(!bitRead(palabra[3],1))      qpd[1]
       = -qpd[1];
117     if(!bitRead(palabra[3],2))      qpd[2]
       = -qpd[2];
118     if(bitRead(palabra[3],3)){
119       qpd[0]=qpd[0]*2;
120       qpd[1]=qpd[1]*2;
121       qpd[2]=qpd[2]*2;
122       }
123     if(bitRead(palabra[3],5)){
124       vel[0]=0;
125       vel[1]=0;
126       vel[2]=0;
127       vel[3]=0;
128       }
129     //Cinematica => vphipd=r*E*(R')*qpd;
130     veld[0] = (qpd[0] + qpd[1] + qpd[2]*(
      l1 + l2))/ra;
131     veld[1] = (qpd[0] - qpd[1] - qpd[2]*(
      l1 + l2))/ra;
132     veld[2] = (qpd[0] + qpd[1] - qpd[2]*(
      l1 + l2))/ra;
133     veld[3] = (qpd[0] - qpd[1] + qpd[2]*(
      l1 + l2))/ra;
134     //Acotar velocidad deseada
```

```
135     if (veld[0] > 11)    veld[0] = 11;
136     else if(veld[0] < -11)    veld[0] =
    -11;
137     if (veld[1] > 11)     veld[1] = 11;
138     else if(veld[1] < -11)    veld[1] =
    -11;
139     if (veld[2] > 11)    veld[2] = 11;
140     else if(veld[2] < -11)    veld[2] =
    -11;
141     if (veld[3] > 11)    veld[3] = 11;
142     else if(veld[3] < -11)    veld[3] =
    -11;
143 //Calculo de errores de velocidad
144     error_vel[0] = veld[0] - vel[0];
145     error_vel[1] = veld[1] - vel[1];
146     error_vel[2] = veld[2] - vel[2];
147     error_vel[3] = veld[3] - vel[3];
148 //Calculo de la derivada del error
149     /*dere[0] = (error[0] - error0[0]) /
    0.03;
150     dere[1] = (error[1] - error0[1]) /
    0.03;
151     dere[2] = (error[2] - error0[2]) /
    0.03;
152     dere[3] = (error[3] - error0[3]) /
    0.03;*/
153 //Calculo del error
154     error[0] += 0.1*error_vel[0];
155     error[1] += 0.1*error_vel[1];
156     error[2] += 0.1*error_vel[2];
157     error[3] += 0.1*error_vel[3];
158 //Calculo del filtrado del error
159     r[0] = error_vel[0] + kp * error[0] /
    kv; //filtrado del error
160     r[1] = error_vel[1] + kp * error[1] /
    kv;
161     r[2] = error_vel[2] + kp * error[2] /
    kv;
162     r[3] = error_vel[3] + kp * error[3] /
    kv;
163 //Calculo del vector phi (funcion de
    activacion) para cada llanta
164     phi[0][0] = tanh(error_vel[0]);
165     phi[1][0] = tanh(veld[0]);
```

```
166     phi[2][0] = tanh(error[0]);
167     phi[3][0] = tanh(1);
168     phi[0][1] = tanh(error_vel[1]);
169     phi[1][1] = tanh(veld[1]);
170     phi[2][1] = tanh(error[1]);
171     phi[3][1] = tanh(1);
172     phi[0][2] = tanh(error_vel[2]);
173     phi[1][2] = tanh(veld[2]);
174     phi[2][2] = tanh(error[2]);
175     phi[3][2] = tanh(1);
176     phi[0][3] = tanh(error_vel[3]);
177     phi[1][3] = tanh(veld[3]);
178     phi[2][3] = tanh(error[3]);
179     phi[3][3] = tanh(1);
180 //Calculo de los pesos de cada red
    neuronal
181     for (i = 0; i < 4; i++) {
182       w[i][0] += 0.1 * G * phi[i][0] * r
    [0];
183       w[i][1] += 0.1 * G * phi[i][1] * r
    [1];
184       w[i][2] += 0.1 * G * phi[i][2] * r
    [2];
185       w[i][3] += 0.1 * G * phi[i][3] * r
    [3];
186     }
187 //Calculo de la funcion F
188     for (i = 0; i < 4; i++)       f[i] = 0;
189     for (i = 0; i < 4; i++) {
190       f[0] += phi[i][0] * w[i][0];
191       f[1] += phi[i][1] * w[i][1];
192       f[2] += phi[i][2] * w[i][2];
193       f[3] += phi[i][3] * w[i][3];
194     }
195 //Calculo de la ley de control (kv*r  =kv
    *(error[0]+kp*inte[0]/kv))
196     tao[0] = kv * r[0] + f[0];
197     tao[1] = kv * r[1] + f[1];
198     tao[2] = kv * r[2] + f[2];
199     tao[3] = kv * r[3] + f[3];
200     if(bitRead(palabra[3],4)){
201       error[0]=0;
202       error[1]=0;
203       error[2]=0;
```

```
204       error[3]=0;
205       tao[0] = 0;
206       tao[1] = 0;
207       tao[2] = 0;
208       tao[3] = 0;
209       }
210     //Saturacion de la salida (PWM)
211     if (tao[0] > 254)    tao[0] = 254;
212     else if(tao[0] < -254)    tao[0] =
      -254;
213     if (tao[1] > 254)     tao[1] = 254;
214     else if(tao[1] < -254)    tao[1] =
      -254;
215     if (tao[2] > 254)    tao[2] = 254;
216     else if(tao[2] < -254)    tao[2] =
      -254;
217     if (tao[3] > 254)    tao[3] = 254;
218     else if(tao[3] < -254)    tao[3] =
      -254;
219 //Asignacion de valores anteriores
220     /*error0[0] = error[0];
221     error0[1] = error[1];
222     error0[2] = error[2];
223     error0[3] = error[3];*/
224 //Sentido de giro y aplicacion de la
      salida
225     if (tao[0] >= 0)    digitalWrite(DIR1,
      HIGH);
226     else      digitalWrite(DIR1, LOW);
227     if (tao[1] >= 0)    digitalWrite(DIR2,
      LOW);
228     else      digitalWrite(DIR2, HIGH);
229     if (tao[2] >= 0)    digitalWrite(DIR3,
      LOW);
230     else      digitalWrite(DIR3, HIGH);
231     if (tao[3] >= 0)    digitalWrite(DIR4,
      HIGH);
232     else      digitalWrite(DIR4, LOW);
233 //digitalWrite(DIR4, LOW);
234     analogWrite(PWM1, abs(tao[0]));
235     analogWrite(PWM2, abs(tao[1]));
236     analogWrite(PWM3, abs(tao[2]));
237     analogWrite(PWM4, abs(tao[3]));
238     // if the LED is off turn it on and
```

```
    vice-versa:
239     if (ledState == LOW)
240       ledState = HIGH;
241     else
242       ledState = LOW;
243     // set the LED with the ledState of
    the variable:
244     digitalWrite(13, ledState);
245 //Impresion de resultados
246     Serial.print(float(t)/1000);Serial.
    print(", ");
247     Serial.print(String(vel[0]) + ", " +
    String(vel[1]) + ", " + String(vel[2])
    + ", " + String(vel[3]) + ", ");
248     Serial.println(String(tao[0]) + ", " +
     String(tao[1]) + ", " + String(tao[2])
     + ", " + String(tao[3]));
249   }
250 }
```

Acrónimos

RN	Red Neuronal
N-A	Neuro-Adaptable
RNC	Red Neuronal Convolutiva
g.d.l.	grados de libertad
CNA	Control Neuro-Adaptable
CN-AM	Control Neuro-Adaptable Modificado
CPD-CNA	Control PD con Compensación Neuro-Adaptable
RMS	*Root Mean Square Error*
YOLO	*You Only Look Once*
ONNX	*Open Neural Network eXchange*

Bibliografía

[1] Ian Lenz. "Deep learning for robotics". Tesis doct. Cornell University, 2016 (vid. pág. 1).

[2] F. L. Lewis y Shuzhi Sam Ge. "Neural Networks in Feedback Control Systems". En: *Mechanical Engineers' Handbook*. John Wiley & Sons, Ltd, 2006, págs. 791-825. DOI: 10.1002/0471777455.ch19 (vid. pág. 1).

[3] FL Lewis. "Nonlinear network structures for feedback control". En: *Asian Journal of Control* 1.4 (1999), págs. 205-228 (vid. págs. 1, 2).

[4] Darren M Dawson, Chaouki T Abdallah y Frank L Lewis. *Robot manipulator control: theory and practice*. CRC Press, 2003 (vid. pág. 1).

[5] F. L. Lewis, A. Yesildirak y Suresh Jagannathan. *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Bristol, PA, USA: Taylor & Francis, Inc., 1998 (vid. págs. 1, 14, 17, 18, 21, 59).

[6] J.A. Farrell y M.M. Polycarpou. *Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches*. Adaptive and Cognitive Dynamic Systems: Signal Processing, Learning, Communications and Control. Wiley, 2006 (vid. pág. 1).

[7] CM Kwan, DM Dawson y FL Lewis. "Robust adaptive control of robots using neural network: global tracking stability". En: *Proceedings of 1995 34th IEEE Conference on Decision and Control*. Vol. 2. IEEE. 1995, págs. 1846-1851 (vid. pág. 2).

[8] Huan Wang y col. "Adaptive control of micro-electro-mechanical system gyroscope using neural network compensator". En: *Advances in Mechanical Engineering* 11.12 (2019), pág. 1687814019898325 (vid. pág. 2).

[9] Jorge Montoya–Cháirez, Víctor Santibáñez y Javier Moreno–Valenzuela. "Adaptive control schemes applied to a control moment gyroscope of 2 degrees of freedom". En: *Mechatronics* 57 (2019), págs. 73 -85. DOI: https://doi.org/10.1016/j.mechatronics.2018.11.011 (vid. pág. 2).

[10] J. Moreno-Valenzuela y col. "Adaptive Neural Network Control for the Trajectory Tracking of the Furuta Pendulum". En: *IEEE Transactions on Cybernetics* 46.12 (2016), págs. 3439-3452. DOI: 10.1109/TCYB.2015.2509863 (vid. pág. 2).

[11] Fujin Luan y col. "Adaptive neural network control for robotic manipulators with guaranteed finite-time convergence". En: *Neurocomputing* 337 (2019), págs. 153 -164 (vid. pág. 2).

[12] Yuxiang Wu y col. "Adaptive neural network control of uncertain robotic manipulators with external disturbance and time-varying output constraints". En: *Neurocomputing* 323 (2019), págs. 108 -116 (vid. pág. 2).

[13] Chengxiang Liu, Zhijia Zhao y Guilin Wen. "Adaptive neural network control with optimal number of hidden nodes for trajectory tracking of robot manipulators". En: *Neurocomputing* 350 (2019), págs. 136 -145 (vid. pág. 2).

[14] Zeqi Yang, Jinzhu Peng y Yanhong Liu. "Adaptive neural network force tracking impedance control for uncertain robotic manipulator based on nonlinear velocity observer". En: *Neurocomputing* 331 (2019), págs. 263 -280 (vid. pág. 2).

[15] F. Baghbani y col. "Emotional neural networks with universal approximation property for stable direct adaptive nonlinear control systems". En: *Engineering Applications of Artificial Intelligence* 89 (2020), pág. 103447. DOI: https://doi.org/10.1016/j.engappai.2019.103447 (vid. pág. 2).

[16] W. He, Y. Chen y Z. Yin. "Adaptive Neural Network Control of an Uncertain Robot With Full-State Constraints". En: *IEEE Transactions on Cybernetics* 46.3 (2016), págs. 620-629 (vid. pág. 2).

[17] Xiao-li Li y col. "Multi-model adaptive control based on fuzzy neural networks". En: *Journal of Intelligent & Fuzzy Systems* 27.2 (2014), págs. 965-975 (vid. pág. 2).

[18] Qitian Yin y col. "Neural network adaptive tracking control for a class of uncertain switched nonlinear systems". En: *Neurocomputing* 301 (2018), págs. 1 -10 (vid. págs. 2, 3).

[19] Xinjun Wang y col. "Neural network based adaptive dynamic surface control of nonaffine nonlinear systems with time delay and input hysteresis nonlinearities". En: *Neurocomputing* 333 (2019), págs. 53 -63 (vid. pág. 3).

[20] Atta Oveisi, Mateus Bagetti Jeronimo y Tamara Nestorović. "Nonlinear observer-based recurrent wavelet neuro-controller in disturbance rejection control of flexible structures". En: *Engineering Applications of Artificial Intelligence* 69 (2018), págs. 50 -64 (vid. pág. 3).

[21] Vu Thi Yen, Wang Yao Nan y Pham Van Cuong. "Recurrent fuzzy wavelet neural networks based on robust adaptive sliding mode control for industrial robot manipulators". En: *Neural Computing and Applications* 31.11 (2019), págs. 6945-6958 (vid. pág. 3).

[22] Kewei Xia y Wei Huo. "Robust adaptive backstepping neural networks control for spacecraft rendezvous and docking with uncertainties". En: *Nonlinear Dynamics* 84.3 (2016), págs. 1683-1695 (vid. pág. 3).

[23] Javier Moreno–Valenzuela, Jorge Montoya–Cháirez y Víctor Santibáñez. "Robust trajectory tracking control of an underactuated control moment gyroscope via neural network–based feedback linearization". En: *Neurocomputing* 403 (2020), págs. 314 -324 (vid. pág. 3).

[24] Darsh Parekh y col. "A Review on Autonomous Vehicles: Progress, Methods and Challenges". En: *Electronics* 11.14 (2022). DOI: 10.3390/electronics11142162 (vid. pág. 3).

[25] I. Lenz, M. Gemici y A. Saxena. "Low-power parallel algorithms for single image based obstacle avoidance in aerial robots". En: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, págs. 772-779. DOI: 10.1109/IROS.2012.6386146 (vid. pág. 3).

[26] A. S. Polydoros, L. Nalpantidis y V. Krüger. "Real-time deep learning of robotic manipulator inverse dynamics". En: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, págs. 3442-3448. DOI: 10.1109/IROS.2015.7353857 (vid. pág. 3).

[27] V. M. Aparanji, U. V. Wali y R. Aparna. "Robotic motion control using machine learning techniques". En: *2017 International Conference on Communication and Signal Processing (ICCSP)*. 2017, págs. 1241-1245. DOI: 10.1109/ICCSP.2017.8286579 (vid. pág. 3).

[28] K. M. I. Khalilullah y col. "Development of robot navigation method based on single camera vision using deep learning". En: *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. 2017, págs. 939-942. DOI: 10.23919/SICE.2017.8105675 (vid. pág. 4).

[29] X. Du, M. H. Ang y D. Rus. "Car detection for autonomous vehicle: LIDAR and vision fusion approach through deep learning framework". En: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, págs. 749-754. DOI: 10.1109/IROS.2017.8202234 (vid. pág. 4).

[30] Q. Li y col. "Deep neural networks for improved, impromptu trajectory tracking of quadrotors". En: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, págs. 5183-5189. DOI: 10.1109/ICRA.2017.7989607 (vid. pág. 4).

[31] L. Meng, T. Hirayama y S. Oyanagi. "Underwater-Drone With Panoramic Camera for Automatic Fish Recognition Based on Deep Learning". En: *IEEE Access* 6 (2018), págs. 17880-17886. DOI: 10.1109/ACCESS.2018.2820326 (vid. pág. 4).

[32] S. Dionisio-Ortega y col. "A deep learning approach towards autonomous flight in forest environments". En: *2018 International Conference on Electronics, Communications and Computers (CONIELECOMP)*. 2018, págs. 139-144. DOI: 10.1109/CONIELECOMP.2018.8327189 (vid. pág. 4).

[33] Waleed Ali y col. *YOLO3D: End-to-end real-time 3D Oriented Object Bounding Box Detection from LiDAR Point Cloud*. 2018. DOI: 10.48550/ARXIV.1808.02350. URL: https://arxiv.org/abs/1808.02350 (vid. pág. 4).

[34] Martin Simon y col. *Complex-YOLO: Real-time 3D Object Detection on Point Clouds*. 2018. DOI: 10.48550/ARXIV.1803.06199. URL: https://arxiv.org/abs/1803.06199 (vid. pág. 4).

[35] Gregory P. Meyer y col. "LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving". En: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, págs. 12669-12678. DOI: 10.1109/CVPR.2019.01296 (vid. pág. 4).

[36] Peixuan Li y col. *RTM3D: Real-time Monocular 3D Detection from Object Keypoints for Autonomous Driving*. 2020. DOI: 10.48550/ARXIV.2001.03343. URL: https://arxiv.org/abs/2001.03343 (vid. pág. 4).

[37] Ashish Vaswani y col. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762 (vid. págs. 5, 39).

[38] R. Kelly, V.S. Davila y J.A.L. Perez. *Control of Robot Manipulators in Joint Space*. Advanced Textbooks in Control and Signal Processing. Springer London, 2006 (vid. pág. 7).

[39] Ricardo Campa, Rafael Kelly y Victor Santibañez. "Windows-based real-time control of direct-drive mechanisms: platform description and experiments". En: *Mechatronics* 14.9 (2004), págs. 1021-1036. DOI: https://doi.org/10.1016/j.mechatronics.2004.04.004 (vid. pág. 7).

[40] G. Campion, G. Bastin y B. Dandrea-Novel. "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots". En: *IEEE Transactions on Robotics and Automation* 12.1 (1996), págs. 47-62. DOI: 10.1109/70.481750 (vid. pág. 8).

[41] R. Kelly y V. Santibáñez. *Control de movimiento de robots manipuladores*. Automática y Robótica. Pearson Educación, 2003 (vid. págs. 9, 23).

[42] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999 (vid. pág. 14).

[43] H.K. Khalil. *Nonlinear Systems*. Prentice Hall, 2002 (vid. págs. 17, 21).

[44] J.J.E. Slotine y W. Li. *Applied Nonlinear Control*. Prentice Hall, 1991 (vid. pág. 21).

[45] Jean-Jacques E Slotine y Weiping Li. "On the adaptive control of robot manipulators". En: *The international journal of robotics research* 6.3 (1987), págs. 49-59 (vid. pág. 21).

[46] Fuzhen Zhuang y col. *A Comprehensive Survey on Transfer Learning*. 2019. DOI: 10.48550/ARXIV.1911.02685. URL: https://arxiv.org/abs/1911.02685 (vid. pág. 37).

[47] Chuanqi Tan y col. *A Survey on Deep Transfer Learning*. 2018. DOI: 10.48550/ARXIV.1808.01974. URL: https://arxiv.org/abs/1808.01974 (vid. pág. 37).

[48] Saad Albawi, Tareq Abed Mohammed y Saad Al-Zawi. "Understanding of a convolutional neural network". En: *2017 International Conference on Engineering and Technology (ICET)*. 2017, págs. 1-6. DOI: 10.1109/ICEngTechnol.2017.8308186 (vid. pág. 38).

[49] Anamika Dhillon y Gyanendra K Verma. "Convolutional neural network: a review of models, methodologies and applications to object detection". En: *Progress in Artificial Intelligence* 9.2 (2020), págs. 85-112. DOI: https://doi.org/10.1007/s13748-019-00203-0 (vid. pág. 38).

[50] Ross B. Girshick y col. "Rich feature hierarchies for accurate object detection and semantic segmentation". En: *CoRR* abs/1311.2524 (2013) (vid. pág. 38).

[51] Kaiming He y col. *Mask R-CNN*. 2017. DOI: 10.48550/ARXIV.1703.06870. URL: https://arxiv.org/abs/1703.06870 (vid. pág. 38).

[52] Tian Jin y col. *Compiling ONNX Neural Network Models Using MLIR*. 2020. DOI: 10.48550/ARXIV.2008.08272. URL: https://arxiv.org/abs/2008.08272 (vid. pág. 39).

[53] Joseph Redmon y col. *You Only Look Once: Unified, Real-Time Object Detection*. 2015. DOI: 10.48550/ARXIV.1506.02640. URL: https://arxiv.org/abs/1506.02640 (vid. pág. 39).

[54] Joseph Redmon y Ali Farhadi. "YOLO9000: Better, Faster, Stronger". En: *arXiv preprint arXiv:1612.08242* (2016) (vid. pág. 39).

[55] Joseph Redmon y Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. DOI: 10.48550/ARXIV.1804.02767. URL: https://arxiv.org/abs/1804.02767 (vid. pág. 39).

[56] Xingkui Zhu y col. *TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios*. 2021. DOI: 10.48550/ARXIV.2108.11539. URL: https://arxiv.org/abs/2108.11539 (vid. pág. 39).

[57] Chien-Yao Wang, I-Hau Yeh y Hong-Yuan Mark Liao. *You Only Learn One Representation: Unified Network for Multiple Tasks*. 2021. DOI: 10.48550/ARXIV.2105.04206. URL: https://arxiv.org/abs/2105.04206 (vid. pág. 39).

[58] Alexey Bochkovskiy, Chien-Yao Wang y Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. DOI: 10.48550/ARXIV.2004.10934. URL: https://arxiv.org/abs/2004.10934 (vid. pág. 39).

[59] GitHub repository. *TinyYOLOv2 Model*. Online. 2020. URL: https://github.com/onnx/models/tree/master/vision/object_detection_segmentation/tiny_yolov2 (vid. pág. 40).

[60] Kaiming He y col. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385 (vid. pág. 40).

[61] Rigoberto Juarez-Salazar, Juan Zheng y Victor H. Diaz-Ramirez. "Distorted pinhole camera modeling and calibration". En: *Appl. Opt.* 59.36 (2020), págs. 11310-11318. DOI: 10.1364/AO.412159 (vid. pág. 40).

[62] Sebastian Raschka, Joshua Patterson y Corey Nolet. *Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence*. 2020. DOI: Redmon2016yolo. URL: https://arxiv.org/abs/2002.04803 (vid. pág. 42).

[63] Sergio López y col. "PD with neuro-adaptive compensation control using the signed power function". En: *International Journal of Control* 0.0 (2022), págs. 1-12. DOI: 10.1080/00207179.2022.2062452 (vid. pág. 60).

[64] Sergio López, Miguel Llama y Ramón García-Hernández. *Controlador PD con compensación Neuro-Adaptable aplicado a la dinámica de un RMR Omnidireccional*. 2019 (vid. pág. 60).