Estimación de la pose mediante sensores inerciales, técnicas de aprendizaje profundo y mediciones de un sistema óptico de captura de movimiento

José Gabriel Matheu Escobar

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Estimación de la pose mediante sensores inerciales, técnicas de aprendizaje profundo y mediciones de un sistema óptico de captura de movimiento

Trabajo de graduación presentado por José Gabriel Matheu Escobar
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2025

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Estimación de la pose mediante sensores inerciales, técnicas de aprendizaje profundo y mediciones de un sistema óptico de captura de movimiento

Trabajo de graduación presentado por José Gabriel Matheu Escobar
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2025

Vo.Bo.:

(f) _____

M. Sc. Carlos Esquit

Tribunal Examinador:

(f) _____

M.Sc. Carlos Esquit

(f) _____

M. Sc. Miguel Enrique Zea Arenales

(f) _____

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

En primer lugar, quiero agradecer a Dios por la oportunidad de gozar de esta gran experiencia rodeado de familia y amigos. Quiero dar gracias por los conocimientos adquiridos, las risas, desvelos e increíbles personajes conocidos a lo largo del camino. En particular, quiero agradecer a mis padres, Luisa y José Rodolfo, quienes han hecho todo esto posible.

Del mismo modo, a toda mi familia y amigos, quiero agradecerles por su apoyo incondicional, lecciones y momentos compartidos que me han ayudado a ser quien soy hoy en día.

Índice

Lista de figuras

Lista de cuadros

Resumen

El objetivo de este estudio fue desarrollar un sistema de captura de movimiento inercial asistido por herramientas de aprendizaje profundo como alternativa a métodos tradicionales. Se propuso en el uso exclusivo de datos originales generados por unidades de medición inercial y obtenidos por un sistema de captura de movimiento óptico para el entrenamiento de los modelos. Esta aplicación se implementó con el fin de mejorar la precisión y reducir la dependencia de sistemas externos para reconstruir la pose y trayectoria de agentes en el espacio. De esa manera, este estudio abarca los desafíos asociados con el efecto del error acumulado y la habilidad de los modelos para realizar predicciones dado el comportamiento estocástico de los datos obtenidos.

El proyecto parte del diseño y ensamblaje de dispositivos capaces de interactuar con los componentes esenciales para captura de movimiento, lo que permitió generar los conjuntos de datos que se utilizaron para el entrenamiento de los modelos de aprendizaje. Así, se exploró el potencial de distintas arquitecturas de redes convolucionales y algoritmos de filtros adaptables para evaluar su capacidad de extracción de características en conjuntos de datos secuenciales. Para concluir, los resultados sugieren que futuras investigaciones puedan profundizar en el estándar de entrenamiento de los modelos propuestos para mejorar la precisión y validar su habilidad para reconstruir la pose y trayectoria en tiempo real.

Abstract

The primary objective of this study was to develop an inertial motion capture system assisted by deep learning tools as an alternative to traditional methods. A focus was placed on using raw data from inertial measurement units and data obtained from optical motion capture systems as references for training the models. This approach was implemented with the goal of improving accuracy and reducing dependencies on external systems for reconstructing the pose and trajectory of agents in the available space. Thus, the study addresses the challenges associated with the effect of accumulated error and the ability of the models to make predictions, given the stochastic behavior of the system.

The project began with the design and assembly of devices capable of interacting with the essential motion capture components, which enables the generation of datasets later used to train the models. This study addresses the potential of different convolutional neural network architectures and adaptive filter algorithms to assess their ability to extract features from sequential datasets. The results observed from this implementation suggest that future research could further investigate the training standards of the proposed models to improve accuracy and validate their ability to reconstruct pose and trajectory in real-time.

CAPÍTULO 1

Introducción

El presente trabajo surge como un acercamiento al problema de captura del movimiento inercial en el Departamento de Ingeniería Electrónica, Mecatrónica y Biomédica de la Universidad del Valle de Guatemala. En particular, este trabajo aborda los desafíos asociados con la reconstrucción de la pose y trayectoria de agentes dotados de unidades de medición inercial, mediante técnicas emergentes de aprendizaje automático, que constituyen un área de investigación con un notable crecimiento en la última década [1]. Esto se debe al esfuerzo llevado a cabo en este campo en busca de sistemas que permitan la captura de movimiento inercial, garantizando resultados más precisos y reduciendo la dependencia de sistemas externos.

Principalmente, la problemática de esta línea de investigación consiste en la presencia de sesgo y error en las lecturas de los sensores que afectan significativamente el desempeño de los algoritmos para reconstruir la pose y trayectoria empleando las características asociadas al movimiento de los agentes. Es decir, se trata de un desafío que involucra reducir el efecto de divergencia en una tasa con incremento exponencial provocado por el ruido en las lecturas de los acelerómetros, giroscopios y magnetómetros, típicamente generados mediante unidades de medición inercial. Por ello, se considera el uso de herramientas de aprendizaje automático con el objetivo de extraer las características en el comportamiento de las señales y establecer sistemas robustos ante el error acumulado.

El objetivo de este trabajo es un acercamiento a la problemática mediante algoritmos de aprendizaje automático conocidos y la asistencia de un sistema de captura de movimiento óptico exclusivamente para el entrenamiento de los modelos. El trabajo realizado se estructura en cuatro etapas para recuperar y procesar los datos originales extraídos de las unidades de medición inercial. Así, considera el diseño y ensamblaje de un dispositivo para obtener y procesar los datos, un programa que establece la conexión entre los dispositivos y sistemas de captura, la propuesta de los modelos de aprendizaje y el estándar para el entrenamiento de dichos algoritmos.

Para alcanzar estos objetivos, el proyecto partió del diseño de los dispositivos capaces de interactuar con los elementos esenciales a través del servidor del ecosistema Robotat. Estos

dispositivos, equipados con unidades de medición inercial y marcadores ópticos, facilitaron la captura y generación de datos aptos para el entrenamiento de modelos de aprendizaje profundo. Dados los conjuntos de datos, se exploraron arquitecturas de aprendizaje profundo con un enfoque en variantes de redes convolucionales y el objetivo de mejorar la capacidad de predicción sobre datos secuenciales. Además, se analizaron diversas herramientas de aprendizaje automático, como filtros adaptables, para evaluar su efectividad en la aproximación de sistemas desconocidos.

Un aspecto crítico de este estudio fue considerar el potencial de los modelos para posteriormente replicar su comportamiento en aplicaciones de captura de movimiento inercial en tiempo real. Los resultados obtenidos sugieren que, aunque los modelos propuestos enfrentan dificultades para realizar predicciones precisas en cuanto a la pose y trayectoria de los agentes, existe potencial para evaluar un estándar más robusto en su implementación en futuras investigaciones.

CAPÍTULO 2

Antecedentes

De manera que se pueda comprender con mayor detalle el funcionamiento y los desafíos en la implementación de un sistema de captura de movimiento basado en unidades de medición inercial (*Inertial Measurement Unit* - IMU) en conjunto con el ecosistema al que se desea asociar el trabajo, se llevó a cabo una investigación acerca de las soluciones y herramientas empleadas en diversos estudios. En este caso, los estudios realizados han buscado implementar métodos que permitan a los usuarios obtener datos precisos sobre la dirección, orientación y posición a partir de los sensores para comprender información relevante sobre su trayectoria o pose en el espacio. En su mayor parte, los antecedentes al proyecto han presentado alternativas competitivas a partir de métodos y herramientas de aprendizaje automático (*machine learning*) para afrontar el error al procesar los datos generados por los sensores debido a condiciones predispuestas en la configuración y el ruido en la lectura de valores aproximados. A continuación, se describen más a detalle las contribuciones de los estudios mencionados en cuanto a las soluciones propuestas, herramientas, métodos empleados y resultados obtenidos durante la implementación.

Self-Contained Position Tracking of Human Movement Using Small Inertial/Magnetic Sensor Modules

Entre los métodos tradicionales empleados para la captura de movimiento con unidades de medición inercial, este estudio [2] presenta las ventajas de estimación de posición en sistemas de navegación para peatones. En este caso, el proyecto introduce la capacidad de los sensores en los dispositivos para calcular las distancias recorridas de acuerdo con la cantidad de pasos que se toman dado el comportamiento que se presenta al caminar. Más a detalle, el sistema permite hacer correcciones sobre las lecturas de velocidad bajo el supuesto que esta se encuentra en un valor de 0 cuando se completa el ciclo y el pie entra en contacto con una superficie. De esa manera, el proyecto concluye con la observación sobre cómo esta tecnología permite obtener trayectorias precisas sobre el movimiento de ciertos agentes en el espacio bidimensional y tridimensional.

El autor remarca que la aplicación descrita es una versión que permite la visualización de los datos luego de procesar la información generada por los sensores. No obstante, los objetivos del estudio buscan una implementación que se pueda integrar para el uso de captura de movimiento en tiempo real contenidos en un dispositivo independiente para aplicaciones en realidad virtual. Así, hace referencia a proyectos que permitan asociar distintos comportamientos y métodos para obtener trayectorias precisas sobre el movimiento natural que se presenta en las extremidades.

Implementación de inteligencia articial para captura de movimiento inercial

A través de varios estudios, el uso de unidades de medición inercial presenta una solución con potenciales para desarrollar tecnologías de navegación independientes, accesibles y eficientes en consumo de energía tanto como en recursos disponibles para dispositivos de uso diario. No obstante, la tecnología presenta varios retos en cuanto a la captura de movimiento cuando se trata de reducir errores producidos por el ruido en lecturas de los sensores disponibles como aceleración, rotación y dirección. Así, el uso de herramientas de *machine learning* en conjunto con métodos de fusión de sensores tradicionales ha permitido desarrollar varias soluciones al problema conocido como deslizamiento (*drift*) en posición y otros relacionados a mejorar las predicciones del comportamiento que se presenta. En este caso, se presentan soluciones destacadas utilizando las siguientes herramientas para obtener la trayectoria de los agentes:

- **RIDI:** implementación con doble integración de aceleración con máquina de vectores de soporte (*Support Vector Machine*- SVM) y regresión de vectores de soporte (*Support Vector regression* - SVR) [3].

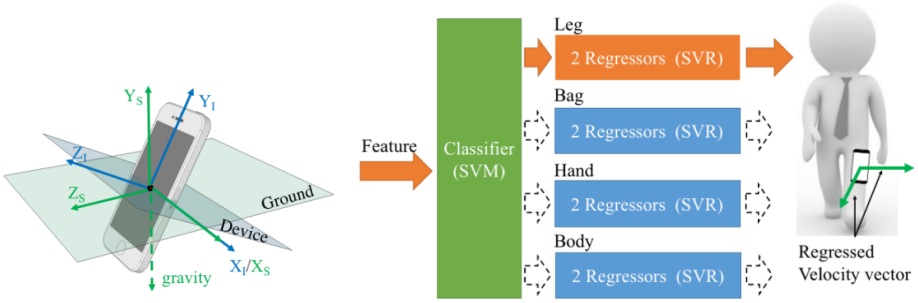

Figura 1: Procesamiento de datos en sistema RIDI para regresión de velocidad [3]

- **RoNIN:** implementación de arquitecturas como redes neuronales residuales (*Residual Neural Networks* - ResNet), redes de memoria a largo y corto plazo (*Long Short Term Memory Networks* - LSTM) y redes de convolución temporal (*Temporal Convolutional Networks* - TCN) para entrenamiento basado en datos [4].

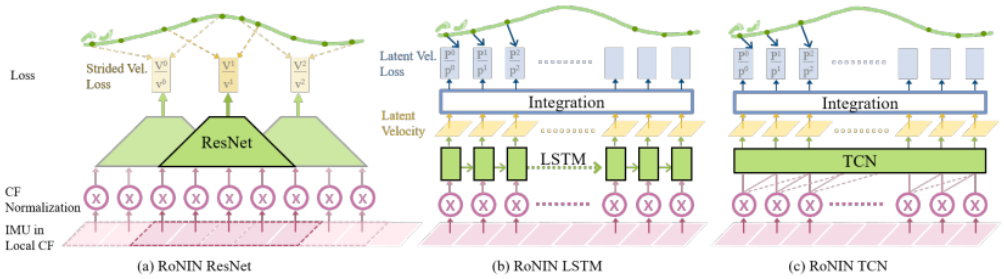

Figura 2: Modelos RoNIN para estimación de trayectorias asistido por aprendizaje profundo [4]

- **AI-IMU *Dead-Reckoning*:** implementación de aprendizaje de redes neuronales profundas (*Deep Neural Networks* - DNN) en conjunto con filtros de Kalman invariantes extendidos (*Invariant Extended Kalman Filter* - IEKF) para captura de trayectoria en vehículos con ruedas [5].

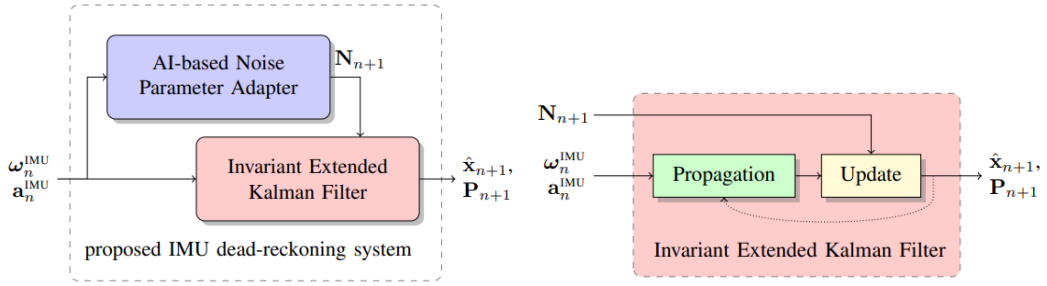

Figura 3: Sistema de navegación por estima asistido por aprendizaje profundo [5]

En los estudios que presentan los métodos anteriores, se comparan los resultados obtenidos entre sí, partiendo de RIDI como el primer método de navegación inercial basada en datos y principalmente sobre un plano en dos dimensiones en conjunto con odometría. En particular, estos trabajos hacen uso de distintas herramientas tal como se ilustra en las Figuras 1, 2 y 3. A continuación, se detallan algunos de las soluciones propuestas y las herramientas que se emplean en los mismos.

RIDI: *Robust IMU Double Integration*

El método de captura de movimiento con unidades de medición inercial [3] plantea una propuesta competitiva a partir de datos y el uso de *machine learning* para reducir los efectos de error acumulado en los sensores. Con más detalle, se propone un método robusto que implementa algoritmos con máquinas de vectores de soporte y regresión de vectores de soporte para abordar el problema de deslizamiento en la posición estimada, a partir de una doble integración de aceleración generada por el dispositivo. Así, el estudio busca facilitar la implementación del método para entrenamiento del algoritmo, por medio de regresión de la velocidad con supuestos que restringen la posición a un espacio en 2 dimensiones y eliminan la ambigüedad de ángulos de inclinación (ϕ, *roll*) y cabeceo (θ, *pitch*) para evaluación de trayectorias.

En este contexto, a diferencia de otros métodos para captura de movimiento que se ven limitados por supuestos sobre orientación y eventos conocidos sobre la velocidad, como navegación por estima (*dead-reckoning*) o conteo de pasos respectivamente, se busca una implementación que permita capturar el movimiento natural que se presenta al realizar tareas diarias, tal como se muestra en la Figura 4. De esa manera, se justifica la implementación de los sensores dada su accesibilidad en varios dispositivos y la eficiencia que presenta para obtener datos relevantes a su trayectoria, que podrían propulsar el desarrollo de tecnologías ideales para sistemas de navegación independientes.

Figura 4: Procesamiento de datos en sistema RIDI para regresión de velocidad [3]

Robust methods for estimating the orientation and position of IMU and MARG sensors

El estudio realizado estimar la orientación y posición asistido por una red neuronal profunda [6] busca demostrar la capacidad de las herramientas de aprendizaje profundo para mitigar el error acumulado debido al ruido en las señales de los sensores. Primeramente, la implementación propone realizar predicciones de velocidad mediante una red convolucional unidimensional profunda basada en los datos generados por unidades de medición inercial y arreglos de sensores magneto-inerciales (*Magneto-Inertial Sensor Arrays* - MARGs). Además, este acercamiento introduce un sistema de captura de movimiento óptico para el entrenamiento del modelo y la calibración con métodos tradicionales de fusión de sensores. Posteriormente, se describe como la integración de los resultados de velocidad obtenidos del modelo de aprendizaje profundo permite realizar una estimación de la trayectoria y orientación de los agentes de interés.

Tal como se muestra en las Figuras 5 y 6, la implementación del sistema entrenado por un sistema de captura de movimiento óptico y asistido por aprendizaje profundo supera las limitaciones de los modelos tradicionales para estimar la posición y orientación. A manera de concluir, el estudio afirma que la solución propuesta tiene un potencial para generar un sistema que permita realizar la captura de movimiento con mayor precisión y en tiempo real.

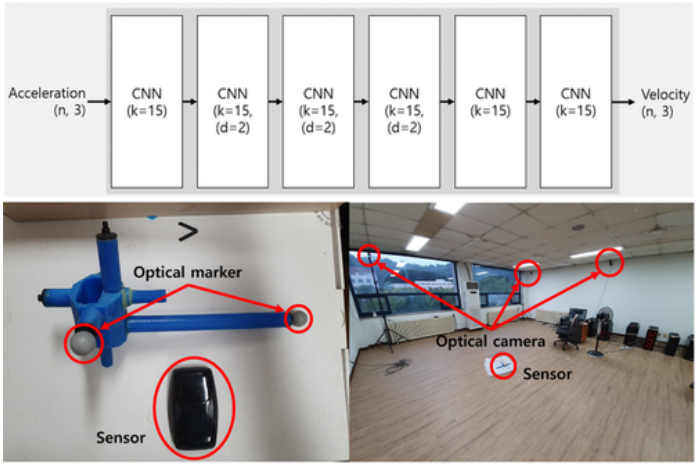

Figura 5: Sistema y arquitectura de red convolucional en estudio para predicción de velocidad asistida por captura de movimiento óptico [6]

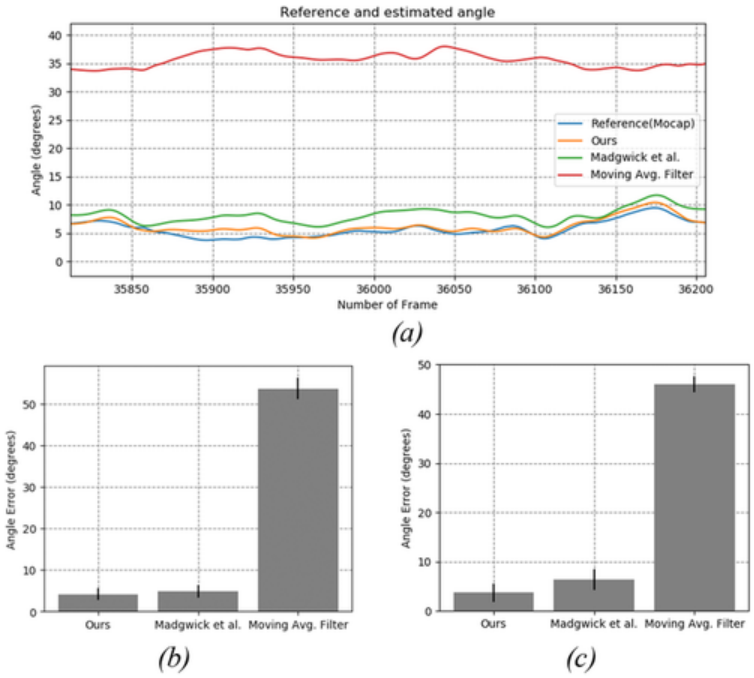

Figura 6: Resultados de estudio en redes convolucionales profundas para predicción de velocidad [6]

Implementación en ecosistema robótico

Dado las herramientas disponibles, ciertos estudios buscan mejorar las lecturas realizadas por las unidades de medición inercial al hacer una fusión de sensores con un método de captura óptico en el entrenamiento de los algoritmos con inteligencia artificial. De esa manera, trabajos previos demuestran que se puede facilitar la obtención de datos a partir de la conexión inalámbrica al servidor del ecosistema robótico dado la disposición de un sistema de captura de movimiento que se utilizará para el entrenamiento del algoritmo.

Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica

La tesis presentada por Perafán Montoya [7] describe las capacidades de captura de movimiento con el ecosistema de experimentación en el campo de robótica, Robotat, ensamblado en la Universidad del Valle de Guatemala. Planteó como objetivo la implementación de una comunicación inalámbrica mediante wifi para experimentación en captura de movimiento de múltiples agentes con cámaras OptiTrack. Además, dentro del alcance del proyecto, se introdujo la posibilidad de trabajar con unidades de medición inercial para capturar datos relevantes a los agentes presentados en el espacio disponible e integrar los resultados a partir de una antena inteligente ilustrada en la Figura 7.

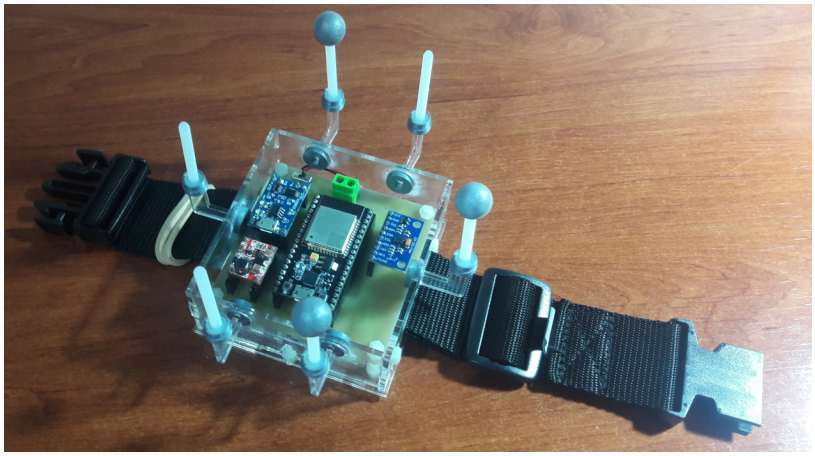

Figura 7: Antena inteligente para interacción con el ecosistema Robotat [7]

En este caso, con la intención de mejorar la precisión en orientación, el proyecto emplea un método tradicional mediante la predicción y corrección de los datos por fusión de sensores con un filtro de Kalman (*Kalman Filter* - KF). Así, el sistema es capaz de aprovechar las ventajas que se presentan en los métodos de captura de movimiento óptico e inercial en donde se consigue una referencia real del espacio y se aumenta la exactitud en orientación de los agentes debido a los datos generados respectivamente. Además, concluye que las librerías desarrolladas para el procesamiento de los datos son aptas para operar con los recursos disponibles en un microcontrolador ESP32 y facilitan la obtención de información de las cámaras. De esa manera, se hace una referencia a futuras implementaciones al ecosistema que permitan capturar movimientos más complejos como para el comportamiento natural de los humanos.

CAPÍTULO 3

Justificación

En la actualidad, los sistemas de captura de movimiento y navegación presentan soluciones importantes en la comprensión del comportamiento de varios agentes en el espacio disponible. Las aplicaciones de esta tecnología han sido de interés para optimización y desarrollo en campos de estudio de medios de transporte, sistemas de localización, realidad aumentada, medicina e incluso entretenimiento. De esa manera, se considera que ciertos métodos destacan de acuerdo con la aplicación y disponibilidad de herramientas específicas en donde, por motivos de este estudio, muchas hacen uso notable de las categorías de captura de movimiento ópticos e inercial. No obstante, los estudios han demostrado que existen varios retos que aún afectan la eficiencia y veracidad de los métodos empleados de manera que el desarrollo de nuevas tecnologías, como herramientas de inteligencia artificial, han ayudado a impulsar el entendimiento de los datos para algoritmos más robustos en los sistemas de captura.

En este caso, la dirección a la que apunta esta área de investigación se encuentra en el desarrollo de una tecnología eficiente en recursos, capaz de representar el movimiento en tiempo real de múltiples agentes en el espacio, sin necesidad de dispositivos externos como cámaras y sistemas de localización satelital. Por esa razón, el enfoque de este trabajo es la implementación de dispositivos haciendo uso de unidades de medición inercial, cuyos beneficios remarcan la eficiencia en recursos, accesibilidad y datos que permiten estimar valores asociados al movimiento de los agentes con alta velocidad. Así, se considera que encontrar solución al reto de estimación del movimiento con los sensores representa un valor agregado que se ha podido optimizar para aplicaciones de captura en tiempo real con herramientas de inteligencia artificial. Por ello, estudios anteriores han buscado la implementación de este método de captura con algoritmos inteligentes con fusión de sensores para experimentación en trayectorias de varios agentes como vehículos, peatones y movimiento de extremidades en el cuerpo humano.

La implementación propuesta es un dispositivo capaz de entregar información sobre la pose de los agentes y un algoritmo entrenado con datos de cámaras especializadas en captura de movimiento, para estimar la posición y la orientación que mantienen. Más a detalle,

este trabajo busca aprovechar las cualidades de esta tecnología para experimentación en el ecosistema robótico ubicado en la Universidad del Valle de Guatemala, en donde se encuentra instalado el sistema de captura con cámaras OptiTrack, para el entrenamiento del algoritmo. Así, en conjunto con las instalaciones y herramientas, haciendo uso de estos dispositivos permitirá a los usuarios comprender las capacidades de la tecnología; sus aplicaciones en varios campos de estudio y experimentación en distintos escenarios de robótica.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Implementar un sistema eficiente para estimar trayectorias y la pose completa de las manos, empleando un dispositivo dotado de sensores inerciales y un algoritmo de aprendizaje profundo entrenado con mediciones de un sistema de captura de movimiento óptico.

4.2. Objetivos específicos

- Diseñar y ensamblar dispositivos de captura de movimiento inercial manteniendo un factor de forma ergonómico que facilite el montaje de marcadores ópticos utilizados para interactuar con el sistema de captura con cámaras OptiTrack
- Configurar los dispositivos para interactuar en el Robotat y facilitar el entrenamiento del algoritmo mediante una red de comunicación wifi local embebida en el microcontrolador seleccionado
- Proponer un algoritmo con arquitectura de red neuronal profunda para minimizar el error generado por captura de movimiento específico en aplicaciones con unidades de medición inercial y evaluar el método de aprendizaje a partir de los datos obtenidos en el sistema de captura óptico instalado.
- Estandarizar el proceso de entrenamiento del algoritmo para captura de movimiento inercial mediante los datos generados por el sistema de captura OptiTrack como datos de campo reales en las trayectorias y poses de los marcadores.

CAPÍTULO 5

Alcance

Este trabajo se centró en la implementación de modelos de aprendizaje profundo para estimar la pose y la trayectoria a partir de los datos generados por dispositivos equipados con unidades de medición inercial. Se adoptó un enfoque exploratorio, dado que la problemática abordada continúa representando un desafío en esta línea de investigación. A su vez, se propuso el uso de modelos de aprendizaje para evaluar su potencial frente a las limitaciones de métodos tradicionales y la dependencia de sistemas externos. Por ello, se definió la necesidad de desarrollar los modelos y un flujo de trabajo para obtener conjuntos de datos fiables, considerando los recursos disponibles y la complejidad del problema de optimización que implica la estimación de pose y trayectoria en el contexto de captura de movimiento inercial. Además, se buscó hacer uso de los elementos instalados en el ecosistema robótico, para asistir en la captura de movimiento óptico en un espacio controlado.

La primera etapa consistió en el diseño de dispositivos para asistir en la captura de movimiento inercial, con el objetivo de generar los conjuntos de datos requeridos para el entrenamiento de los modelos de aprendizaje. Esto permitió disponer de un sistema adecuado para la obtención de los datos inerciales y los datos del sistema de captura Optitrack, dada la dependencia de la calidad de los conjuntos generados para asegurar el desempeño de los modelos. En este contexto, la comunicación entre dispositivos y el servidor del ecosistema Robotat se estableció a través del protocolo TCP/IP, garantizando la generación de los conjuntos de datos ordenados secuencialmente a costa de la velocidad de envío y recepción de los mismos. Aunque se diseñaron dos dispositivos de captura, se decidió limitar las sesiones de grabación con un solo dispositivo en las primeras fases del trabajo con el fin de simplificar la comunicación y generar los conjuntos esenciales para iniciar la experimentación con los modelos. Esta decisión permitió invertir el tiempo en establecer un programa multitarea más robusto en los dispositivos de captura para interactuar con los elementos del sistema, buscando generar conjuntos de datos más informados y facilitar las primeras etapas del diseño de arquitecturas en los modelos de aprendizaje profundo.

Así, la experimentación con modelos de aprendizaje partió de las redes neuronales profundas, estableciendo un enfoque en la extracción de características relevantes mediante capas

convolucionales y herramientas adicionales para procesar datos secuenciales, estudiadas durante el desarrollo del trabajo. Esta decisión permitió enfocar la investigación a aplicaciones que emplean técnicas basadas en datos y evaluar el potencial de arquitecturas con cierta noción del funcionamiento de métodos tradicionales, tal como sugieren la encuesta [1] y empleando un acercamiento similar a [6] descrito en los antecedentes. Además, durante el desarrollo del trabajo, se optó por explorar los filtros adaptables, que surgieron como una alternativa adicional dentro del contexto de aprendizaje automático, con el fin de evaluar su capacidad para identificar sistemas de manera iterativa y en tiempo real. Sin embargo, dado el carácter exploratorio del trabajo, no se priorizó el ajuste de parámetros durante la etapa de experimentación con los modelos y algoritmos propuestos. En su lugar, se priorizó encontrar tendencias claras en la respuesta de los mismos para demostrar su potencial en las condiciones para la estimación de pose y trayectoria.

CAPÍTULO 6

Marco teórico

De manera que se pueda generar un entendimiento del alcance y contexto del proyecto, a continuación se detallan los conceptos y herramientas relevantes para la captura de movimiento con unidades de medición inercial. Debido a la naturaleza de los sistemas descritos y el conocimiento de soluciones que emplean herramientas de inteligencia artificial, se exploran los conceptos que abarcan los métodos de captura de movimiento, aprendizaje profundo, filtros adaptables y fusión de sensores.

Además, se detallan las herramientas empleadas para la solución propuesta, considerando el funcionamiento de los sensores y la implementación del proyecto en un ambiente dotado de un sistema de captura de movimiento óptico. Más adelante, se desarrollan ciertos conceptos de importancia relacionados con la capacidad de las herramientas empleadas en el trabajo descrito.

6.1. Captura de movimiento

La captura de movimiento (*Motion Capture* - MoCap) es un campo en el que se emplean diversas técnicas para representar la interacción de determinados agentes dentro de un espacio definido en un entorno digital. Debido a las ventajas que ofrece esta tecnología, su aplicación para capturar el movimiento de múltiples agentes en un espacio definido ha sido objeto de interés en diversos estudios [8]. En este sentido, han surgido varias técnicas para mejorar su habilidad de reconstrucción de la posición y orientación de los agentes estudiados. En consecuencia, las aplicaciones abarcan una amplia variedad de campos, que incluyen sistemas de navegación, programas de rehabilitación, medicina, seguridad industrial, robótica y entretenimiento [9].

En el contexto de este trabajo, se consideran los métodos de captura de movimiento óptico e inercial que se caracterizan por facilitar la comprensión del movimiento natural de los agentes en un entorno digital mediante sensores con diversas limitantes. No obstante, el objetivo principal de su implementación radica en la medición de parámetros, tales como

posición, orientación y dirección para estimar las trayectorias o poses de los agentes en 3 dimensiones [10].

6.1.1. Captura de movimiento óptico

El método de captura de movimiento óptico hace uso de marcadores para determinar la ubicación de los agentes en el espacio visible a partir de cámaras. Las técnicas que abarca este método permiten capturar movimiento sobre el espacio definido con una precisión elevada, limitada en su capacidad por el periodo de muestreo y el espacio visible de las cámaras [11]. Más a detalle, esta tecnología emplea técnicas de captura en donde los marcadores emiten o reflejan luz infrarroja y derivan las características de su posición dados distintos puntos de referencia.

En el caso de la técnica de captura óptica activa, los marcadores se encargan de emitir una señal de luz infrarroja de manera que se pueda identificar inequívocamente entre otros objetos, incluso en ambientes expuestos a otras fuentes de luz. Dadas las características de esta herramienta, se considera que la aplicación de la tecnología otorga la habilidad para desempeñarse incluso en espacios abiertos al exterior.

En este contexto, la técnica de captura de movimiento pasiva se encarga de emplear marcadores con materiales que reflejan luz infrarroja emitida desde las cámaras instaladas en el área de trabajo. En este caso, la configuración se puede aprovechar en escenarios controlados según la capacidad de resolución para identificar los marcadores y el campo de visión de las cámaras.

Cámaras Optitrack

Optitrack es una compañía que se especializa en ofrecer herramientas, equipo y desarrollar soluciones para aplicaciones con sistemas de captura de movimiento ópticos [12]. Por motivos de su implementación en este trabajo, el sistema instalado en el ecosistema robótico permite recuperar los datos reales (*gorund truth*) de los agentes en el espacio definido. Más a detalle, se trata de un sistema dotado con las cámaras primeX4 de la marca, tal como se muestra en la Figura 8.

Figura 8: Cámaras Otritrack - primeX4 [13]

Dado el modelo de cámaras se consideran las siguientes especificaciones relevantes para su implementación en el proyecto [13] :

- **Resolución:** 2048×2048.
- **Tasa de fotogramas nativa:** 180 Hz.
- **Latencia:** 5.5 ms.
- **Rendimiento de precisión en 3D:** $\pm$ 0.10 mm.
- **Rendimiento con marcadores pasivos:** 30 m (100').

6.1.2. Captura de movimiento inercial

El método de captura de movimiento inercial se basa principalmente en el uso de sensores específicos para obtener información relevante sobre el comportamiento en orientación, posición y dirección de múltiples agentes. Se atribuye al método la habilidad de las unidades de medición inercial para medir valores como la aceleración, velocidad angular e intensidad de campos magnéticos de manera que se pueda estimar la pose y trayectoria a partir de los conceptos introducidos en el contexto de fusión de sensores [14].

No obstante, el desarrollo de sistemas de captura inercial ha demostrado que existen retos para la estimación de posición debido al ruido que se genera en la medición de los valores relevantes. Más a detalle, los retos indican un error que se acumula en la obtención de posición, dado un fenómeno al que se le conoce como deriva (*drift*) y sesgo (*bias*) en las mediciones relevantes al giroscopio de los sensores utilizados. Por ello, con los avances en herramientas de inteligencia artificial, se han propuesto soluciones que permiten un acercamiento a sistemas de captura eficientes en recursos y más precisión.

Unidad de medición inercial

Las unidades de medición inercial (IMU) son dispositivos que proporcionan información sobre los fenómenos físicos que experimentan los agentes en movimiento. Estos dispositivos son capaces de medir distintas magnitudes dentro de un marco de referencia inercial, en donde sus grados de libertad describen la cantidad de características extraídas [15]. En este contexto, las unidades son capaces de medir las magnitudes a lo largo de los ejes de su marco de referencia empleando los siguientes sensores:

- **Acelerómetro:** medición de aceleración.
- **Giroscopio:** medición de velocidad angular.
- **Magnetómetro:** medición de intensidad de campo magnético.

Además, se atribuyen las aplicaciones de los sensores dadas las características de un sistema embebido con tecnología microelectromecánica (*Micro-Electromechanical System* - MEMS) de manera que se suelen aprovechar ventajas relacionadas con la accesibilidad, eficiencia de recursos y bajo perfil para proyectos y dispositivos de uso diario [14].

En el presente caso, los dispositivos descritos son aptos para trabajar en ambientes de uso experimental dado la accesibilidad y facilidad de conexión con otros dispositivos o placas de prueba. Mediante protocolos de comunicación I2C o SPI, tal como se muestra en la Figura 9, estos dispositivos son capaces de enviar la información relevante al comportamiento medido por los sensores incluidos.

Módulo MPU-92/65

El módulo MPU-92/65 es un dispositivo equipado con una unidad de medición inercial MPU-9250 que permite 9 grados de libertad. A su vez, el módulo es capaz de operar con un diseño de bajo consumo de energía y una resolución de 16-Bits para los sensores de aceleración, giroscopio y magnetómetro incluidos. Así, se presentan las siguientes especificaciones [16]:

- **Voltaje de operación:** 3-5 VDC.
- **Protocolos de comunicación:** I2C (hasta 400 kHz) o SPI (hasta 1MHz).
- **Rango de acelerómetro:** $\pm$2g, $\pm$4g, $\pm$8g, $\pm$16g.
- **Rango de giroscopio:** $\pm$250, $\pm$500, $\pm$1000, $\pm$2000 deg/s (dps).
- **Rango de magnetómetro:** $\pm$4800μT.

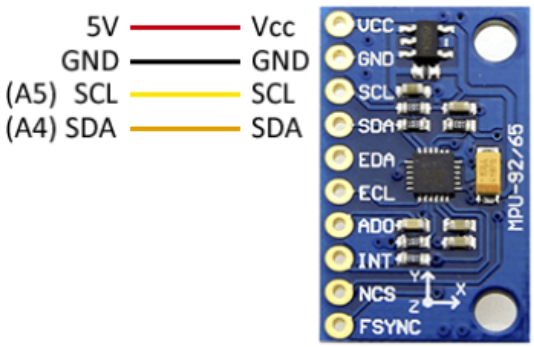

Figura 9: Conexión I2C a módulo MPU 92/65 con IMU de 9 grados de libertad [17]

6.2. Aprendizaje profundo

Estos algoritmos tratan de comprender modelos a partir de los datos generados para diversas aplicaciones, al minimizar el error entre los valores esperados y los resultantes. Estos métodos se distinguen por permitir al algoritmo entrenar y aprender a partir de los conjuntos de datos generados bajo un criterio de etiquetas supervisadas o no supervisadas que se asignan a los valores. De esa manera, las técnicas empleadas permiten comprender y estimar el comportamiento del modelo en función del tipo de información al que se expone para generar estimaciones más precisas [18].

6.2.1. Perceptrón

El comportamiento de estos algoritmos se basa en el funcionamiento de redes neuronales artificiales, construidas por unidades conocidas como perceptrones, ordenadas en varias capas que describen su arquitectura [19]. El funcionamiento de los perceptrones es fundamental, dado que se encargan de procesar los datos para la interpretación de los valores introducidos al modelo descrito. Para lograr los resultados esperados de las unidades, se modelan tal como se muestra en la Figura 10, considerando vectores de variables de entrada, un factor de importancia a las conexiones, una función de activación y la predicción representativa según la aplicación.

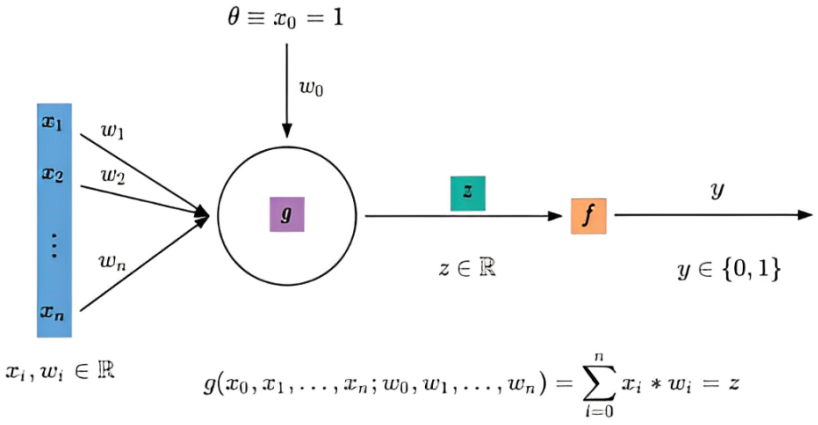

Figura 10: Estructura de un perceptrón [19]

6.2.2. Función de activación

Este elemento se encarga de actuar sobre la predicción de la unidad basado en la suma de importancia asignada y sesgo de las entradas para tomar una decisión sobre el modelo que se está evaluando [19][20]. En este caso, también se pueden considerar algunas las funciones de activación comúnmente empleadas, tal como se ilustran en la Figura 11.

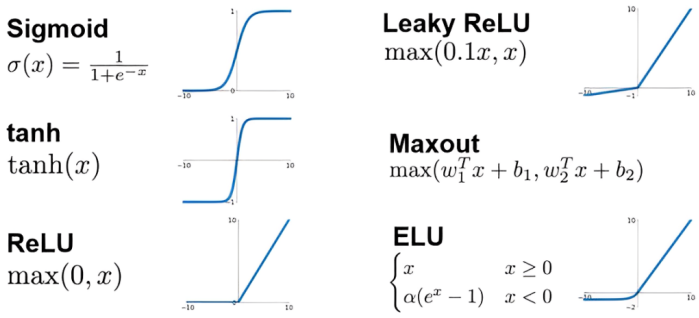

Figura 11: Funciones de activación típicamente empleadas [21]

La implementación de estas funciones comprende un rango de admisión de los valores resultantes y en ciertos casos introduce la habilidad de incluir no linealidades en el sistema descrito [20]. Aunque se reconoce la función de activación de unidad lineal rectificada (*Rectified Linear Unit* - ReLU) mostrada en la Figura 12, existen varias funciones de activación que permiten distintos beneficios para realizar las predicciones, según la aplicación esperada del modelo, como un problema de regresión o clasificación.

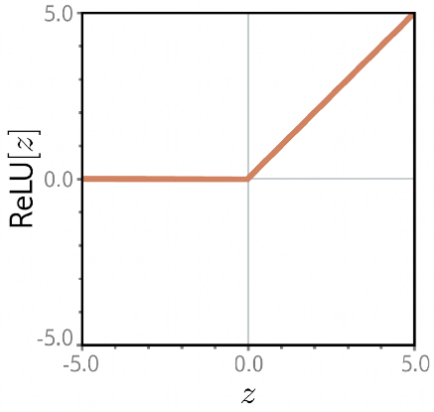

Figura 12: Función de activación de unidad lineal rectificada - ReLU [22]

6.2.3. Criterio de pérdida

Se mencionó que el objetivo de la técnica de aprendizaje profundo, al igual que otras en el campo de aprendizaje automático (*machine learning*), trata de encontrar la mejor predicción del modelo descrito mediante una técnica que minimiza el error contemplando los valores esperados. De esa manera, se comprende el concepto de un criterio de pérdida que describe el error entre los valores para encontrar el mejor "mapeo" para la aplicación deseada [22].

En este caso, los criterios de pérdida comunes se pueden considerar como los siguientes [23]:

- **Pérdida de entropía cruzada (*Cross Entropy Loss*)**:

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = -\sum_{c=1}^{C} w_c \log \left(\frac{\exp(x_{n,c})}{\sum_{i=1}^{C} \exp(x_{n,i})} \right) y_{n,c}. \tag{1}$$

- **Pérdida de entropía cruzada binaria (*Binary Cross Entropy Loss* - BCE)**:

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = -w_n \left[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n) \right]. \tag{2}$$

- **Error cuadrático medio (*Mean Squared Error* - MSE)**:

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = (x_n - y_n)^2. \tag{3}$$

- **Error absoluto medio (*Mean Absolute Error* - MAE)**:

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = |x_n - y_n|. \tag{4}$$

De manera que se pueda comprender en el contexto del entrenamiento de las redes neuronales, los valores de pérdida cumplen con la función de permitir una referencia del error en los valores resultantes y, a su vez, evidencia el aprendizaje de la red respecto al modelo. No obstante, la etapa encargada de ajustar los parámetros para minimizar el error se atribuye al optimizador [22].

6.2.4. Optimizadores y retropropagación

El optimizador es una herramienta que se utiliza en los modelos de aprendizaje profundo para realizar la etapa de entrenamiento de las redes. Tal como se describió, el objetivo es que esta etapa en los modelos de las redes pueda minimizar el error encontrado entre los resultados y los valores esperados De esa manera, en términos de la pérdida, el mejor caso resulta cuando el optimizador encuentra un mínimo global del modelo; en donde la perdida es mínima tal como se puede apreciar en la Figura 13 para los optimizadores de descenso de gradiente (*Gradient Descent*) y estimación de momento adaptativo (*Adaptive Moment Estimation* - Adam).

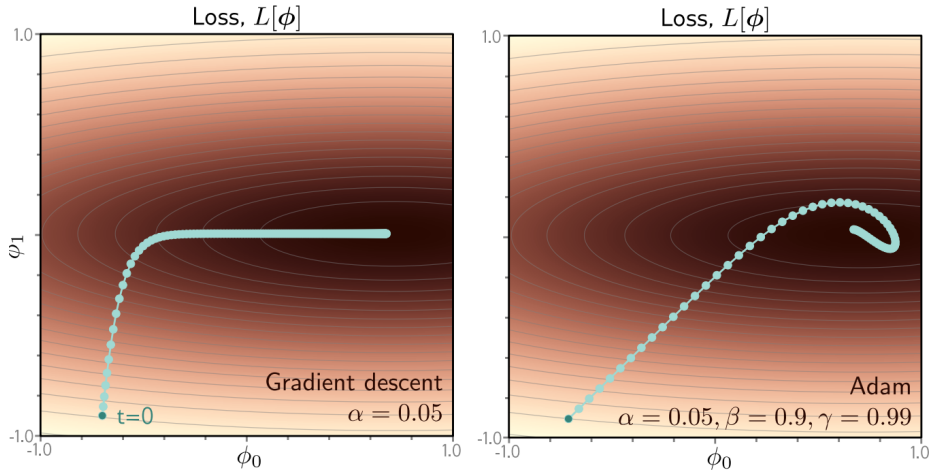

Figura 13: Variantes de optimizador de descenso de gradiente y Adam [22]

En este caso, tal como para los criterios de pérdida, se ha demostrado la aplicación de distintos optimizadores que cuentan con sus ventajas en cuanto a eficiencia de recursos, convergencia de los modelos y velocidad. Más a detalle, la selección de las variantes de modelos de optimización permite el ajuste del entrenamiento respecto a hiperparámetros, dentro de los que cabe mencionar la tasa de aprendizaje y momentum. Así, se pueden considerar las siguientes variaciones de optimización típicamente utilizadas para el entrenamiento de los modelos descritos [22]:

- **Descenso de gradiente (*Gradient Descent*)**: es un algoritmo de optimización que busca minimizar una función de pérdida empleando conjuntos completos de datos. El

algoritmo permite ajustar los parámetros en cada iteración, de acuerdo con la dirección del gradiente y la tasa de aprendizaje.

- **Descenso de gradiente estocástico (*Stochastic Gradient Descent* - SDG)**: es una variante del descenso de gradiente que utiliza un subconjunto aleatorio de los datos en cada iteración. Aunque introduce un comportamiento estocástico en la actualización de los gradientes, este enfoque busca acelerar la convergencia de los modelos y minimizar el riesgo de estancamiento en mínimos locales.
- **Estimación de momento adaptativo (*Adaptive Moment Estimation* - Adam)**: es un algoritmo de optimización que introduce el concepto de momento al algoritmo de descenso de gradiente estocástico, tal como se ilustra en la Figura 13. Este elemento busca mejorar la eficiencia y la estabilidad durante el entrenamiento.

Para comprender este concepto, es posible señalar que el entrenamiento de las redes neuronales se basa en múltiples iteraciones que conllevan la propagación hacia adelante (*forward propagation*) del resultado de cada capa y retropropagación (*backpropagation*) para ajustar los gradientes [20]. En conjunto con la configuración del criterio implementado, estas operaciones buscan minimizar el error y ajustar los parámetros internos de las redes neuronales para realizar predicciones acordes al sistema descrito por los datos expuestos [22].

6.2.5. Nodos convolucionales

Comprendiendo las herramientas que constituyen a una red neuronal, cabe mencionar que existen otras unidades aparte de los perceptrones dentro de las que se debe considerar la introducción de los nodos convolucionales y algunas variantes del mismo. Estas capas en una red neuronal, a diferencia de los perceptrones, utilizan convolución para procesar los datos que mantienen una relación en cuanto al espacio en que se encuentran en cada muestra [19]. Típicamente, estas capas son utilizadas para extracción de características en los sets de datos, por lo que suelen emplearse en el procesamiento de imágenes, sonido o datos en series de tiempo [22].

En el contexto de la construcción y ajuste de las redes neuronales, cabe mencionar que estas capas presentan hiperparámetros adicionales en cuanto a los espacios que evalúa por cada iteración en donde el tamaño de las muestras y la manipulación de una "ventana móvil" o *kernel* son esenciales para su funcionamiento. Considerando las capas neuronales como un filtro para los datos presentados, algunos parámetros como la forma, dilatación y el salto en espacios de datos (*stride*) del *kernel* o relleno (*padding*) de los sets de datos permiten ajustar los mismos para la extracción de características en la aplicación deseada [22]. Tal como se ilustra en la Figura 14, el resultado de la configuración de estos hiperparámetros permite establecer la capacidad de las capas convolucionales para extracción de características directamente sobre los conjuntos de datos ordenados a diferencia de distintas configuraciones en la conexión de capas con múltiples perceptrones.

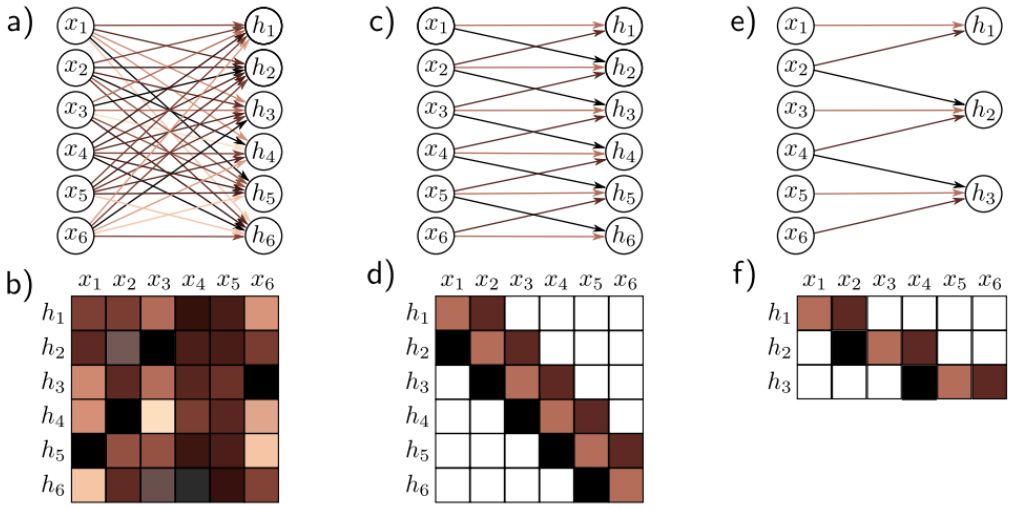

Figura 14: Dispersión en capas convolucionales según configuración de parámetros [22]

Adicionalmente, en el contexto de este trabajo, la habilidad de estas capas para procesar múltiples canales de datos permite un entendimiento de un sistema o filtro de múltiples entradas y múltiples salidas (*Multiple Input Multiple Output* - MIMO). Esta configuración permite evaluar valores para una misma señal en múltiples instancias de espacio o tiempo, tal como se puede apreciar en la Figura 15.

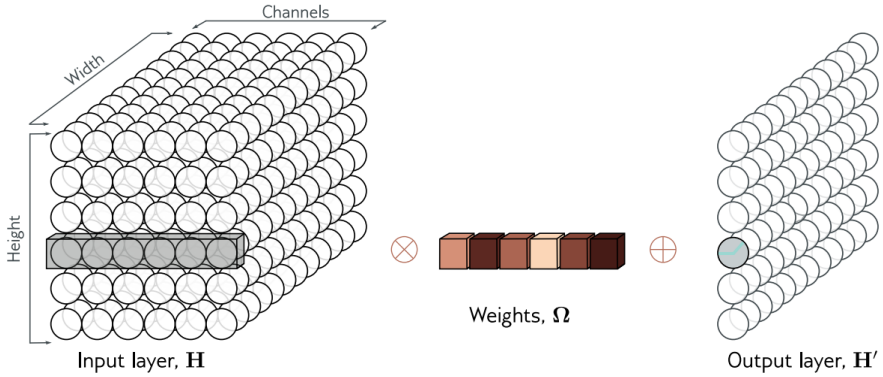

Figura 15: Diagrama de múltiples entradas y múltiples salidas en capa convolucional [22]

Nodos convolucionales de 1 dimensión

El caso de las capas convolucionales de 1 dimensión, se sugieren el uso de las operaciones de los nodos convolucionales para procesar un canal con señales y series de datos secuenciales [22]. De esa manera, se comprende que se trata de una capa que procesa los datos a partir de varias iteraciones que se realizan sobre muestras que representan una porción del set de dato en cierta instancia de tiempo.

6.3. Arquitecturas de red neuronal profunda

Comprendiendo el funcionamiento de las unidades empleadas, la construcción general de las arquitecturas, tal como se muestra en la Figura 16, está conformada por capas con múltiples unidades de procesamiento que cumplen con las siguientes funciones en una red neuronal:

- **Capa de entrada:** se encargan de recibir las entradas como un vector de valores x.
- **Capas ocultas:** se encargan de procesar los datos al asignar un peso a la operación en conjunto con una función de activación para generar el modelo a partir de sus unidades.
- **Capa de salida:** tal como la capa oculta, la capa de salida está compuesta por varias unidades y se encarga de entregar los valores de salida para la aplicación propuesta para el algoritmo.

La técnica de aprendizaje profundo parte de la noción del funcionamiento de redes neuronales superficiales (*Shallow Neural Networks*) y, a su vez, en la relación descrita por regresión lineal para generar predicciones del modelo descrito [22]. A diferencia de las redes neuronales superficiales, las redes neuronales profundas aprovechan las herramientas disponibles para agregar complejidad a las arquitecturas mediante la adición de capas ocultas, tal como se demuestra en la Figura 16.

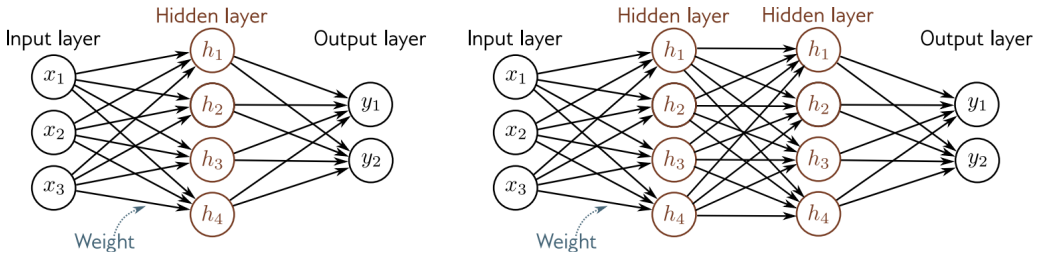

Figura 16: Arquitectura de redes neuronales superficiales y profundas [22]

De esa manera, la adopción de múltiples capas para la construcción de arquitecturas más complejas permitió el desarrollo del campo de aprendizaje profundo, considerando las ventajas de las herramientas que emplean para optimizar su aprendizaje. Algunas arquitecturas establecidas se pueden encontrar en el esfuerzo realizado para generar una librería de redes neuronales [24] de manera de que se presentan los beneficios de la implementación para cada una.

Por motivos de este trabajo, se presentan los casos para las arquitecturas de redes neuronales residuales, tanto como las redes neuronales de convolución temporal. En este caso, se trata de arquitecturas diseñadas para trabajar con la predicción de modelos sobre datos en una serie de tiempo continuo.

6.3.1. Redes neuronales residuales

Las redes neuronales residuales (ResNet) son un tipo de arquitectura de red neuronal profunda que utiliza conexiones residuales entre sus capas con el fin de generar relaciones entre los datos ingresados y su salida [25]. A diferencia de las redes neuronales tradicionales, las capas calculan un residual acumulado tras la suma de la entrada original y su salida, tal como se presenta en la Figura 17. Así, el concepto de los bloques de construcción para esta arquitectura recibe el nombre de bloques residuales, cuyo objetivo consiste en asistir en el entrenamiento al generar un efecto de "memoria" de acuerdo con los gradientes asignados [22].

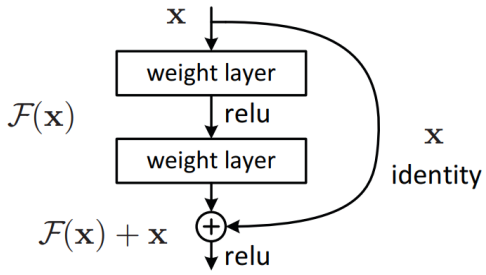

Figura 17: Estructura de bloque residual [25]

De esa manera, se define un bloque residual como:

$$y = \mathcal{F}(x, \{W_i\}) + x. \tag{5}$$

Así, se define un bloque residual en donde $\mathcal{F}(x, \{W_i\})$ representa la salida de la capa de red en función de la entrada y el gradiente, x es la entrada original y y es la salida del bloque temporal. De esa manera se puede definir que la ventaja principal de estos elementos radica en facilitar la propagación de aprendizaje a través de múltiples capas sin perder o distorsionar los gradientes.

Adicionalmente, se considera que las redes residuales emplean varias herramientas clave para mejorar su rendimiento. En esta instancia, adicional a los bloques residuales, estas arquitecturas de red suelen acompañarse de la normalización de lote (*batch norm*) y unidades de rectificación lineal (*ReLu*), cuyas funciones principales son manejar la activación en la toma de decisiones y generar relaciones complejas mediante la introducción de no linealidades [22].

En el contexto de este trabajo, las redes residuales comprenden una forma de trabajar con datos secuenciales que consideran series con cierta noción temporal. La ventaja de su adición radica en su capacidad para mantener gradientes estables y, a su vez, asistir en el entrenamiento del modelo.

6.3.2. Red neuronal convolucional temporal

Las redes convolucionales temporales (TCN) son una arquitectura de red neuronal diseñada particularmente para el procesamiento eficiente de secuencias de datos, especialmente en tareas de segmentación y detección de características con noción del tiempo [26]. Estas redes emplean capas convolucionales que procuran mantener causalidad y buscan asegurar dependencia únicamente sobre la información previa del sistema analizado [27]. Este comportamiento tiene el fin de garantizar que las predicciones realizadas se generen con cierta noción temporal presente en las secuencias de datos.

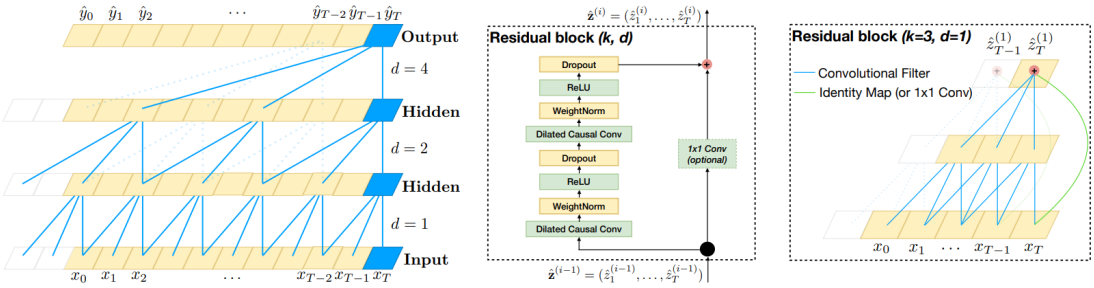

Figura 18: Herramientas empleadas en redes convolucionales temporales [27]

Entonces, la estructura básica de las redes convolucionales temporales supone series de capas convolucionales unidimensionales empleadas para procesar secuencias de datos de longitud fija. Por ello, a través de la dilatación entre las capas implementadas, los modelos adquieren una extensión en el campo receptivo que indica la capacidad para capturar dependencias temporales a largo plazo [27]. En este caso, se comprende que este diseño aprovecha la capacidad de estas conexiones de manera eficiente para el entrenamiento de los modelos y hace uso de herramientas adicionales como bloques residuales para mantener la estabilidad de los gradientes asignados, tal como se ilustra en la Figura 18.

6.3.3. Mecanismos de atención

Los mecanismos de atención son métodos diseñados para asistir en el entrenamiento de redes neuronales al generar conexiones a partes específicas del conjunto de datos expuesto. De manera general, esta herramienta permite ponderar la importancia de distintos segmentos para agilizar el aprendizaje de ciertas tareas particulares en conjuntos de datos extensos al reforzar patrones observados [28]. Comúnmente, estos mecanismos operan de manera dinámica en el contexto de los datos de entrada, lo que permite asignar relevancia de los datos mediante tres componentes principales:

- **Consulta (*query*):** un vector que representa el estado actual o el elemento siendo procesado.
- **Clave (*key*):** un vector que representa los elementos potencialmente relevantes a la consulta realizada.
- **Valor (*value*):** representa el valor de los vectores asociados con la clave que se analiza contra la consulta realizada.

Así, se suele determina la importancia de los segmentos procesados mediante la similitud entre los vectores de consulta y clave. En este contexto, algunos de los mecanismos más populares debido a su capacidad de generar modelos de aprendizaje robustos incluyen la atención por producto punto escalado (*Scaled Dot-Product Attention*) y atención multi-cabeza (*Multi-Head Attention*).

Mecanismo de atención por producto punto escalado

El método introducido en [28] referido como mecanismo de atención por producto punto escalado propone la ponderación de los segmentos tras la operación de producto punto entre los vectores de consulta y clave. Tal como se muestra en la Figura 19, el mecanismo considera una estructura con operaciones que incluye una función de activación *Softmax* que permite asignar el peso de la información procesada en el sistema generado.

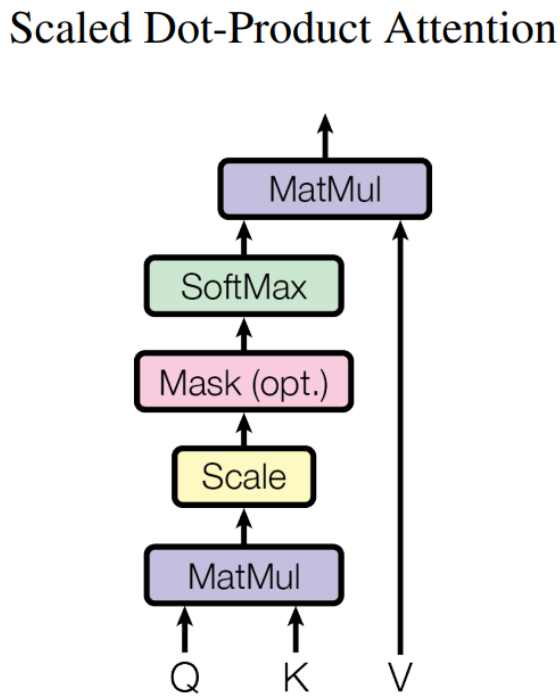

Figura 19: Estructura de mecanismo de atención por producto punto escalado [28]

Más a detalle, el método propuesto se describe mediante un distintivo factor de escalamiento $\sqrt{d_k}$ además de la operación entre los elementos de consulta Q, clave K y valores V como matrices con los datos relevantes.

$$(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{6}$$

En particular, se argumenta que el factor de escalamiento se requiere en el mecanismo para de mitigar el efecto provocado por dimensiones elevadas de la matriz de claves d_k [28].

En este caso, se trata de mitigar valores elevados del producto en la operación entre matrices que ubica a la función de activación en regiones con gradientes reducidos.

Mecanismo de atención multi-cabeza

El mecanismo de atención multi-cabeza descrito en [28], se considera como una extensión del método de atención por producto punto escalado que busca aumentar la capacidad de generar dependencias en los conjuntos de datos más profundas. Tal como se ilustra en la Figura 20, la estructura considera el aumento de capacidad para analizar patrones a partir de h capas de atención paralelas en el modelo.

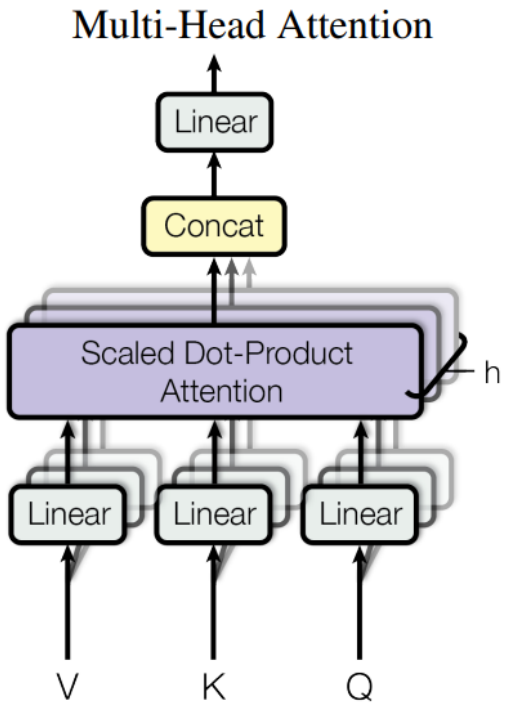

Figura 20: Estructura de mecanismo de atención multi-cabeza [28]

De esa manera, el resultado del mecanismo se obtiene mediante la concatenación de cada capa y una proyección lineal de cada elemento [28]. además, se argumenta que al reducir las dimensiones en el vector de salida de cada capa, el método es capaz de generar una solución más eficiente para generar múltiples conexiones manteniendo el uso de recursos similar al método de atención por producto punto escalado.

6.4. Filtros adaptables

Los filtros adaptables son ampliamente utilizados en procesamiento de datos con el objetivo de minimizar el error entre la señal ingresada y la señal deseada mediante la actualización de sus coeficientes. Estos filtros son capaces de ajustar sus parámetros en diversas iteraciones a lo largo de los conjuntos de datos, de manera que se pueda generar una convergencia más precisa. Los algoritmos empleados incluyen el cálculo de la predicción, el cálculo del error y la actualización de los coeficientes [29].

De acuerdo con [30], los filtros adaptables están descritos por la ecuacion (7) en donde y representa la predicción, $\mathbf{x}$ el vector de entrada al filtro y $\mathbf{w}_i$ el vector de pesos actualizados [31][32].

$$y(n) = \sum_{i=0}^{N-1} \mathbf{w}_i(n) x(n-i). \tag{7}$$

Por otro lado, se determina que la actualización del vector de pesos se da a partir de la ecuación (8) en donde $\mathbf{w}(n)$ es el vector de pesos actualizados, $\mathbf{w}(n\text{-}1)$ es el vector de pesos previos y $\Delta\mathbf{w}(n)$ representa la diferencia calculada de los pesos en esa instancia de tiempo n [29].

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \Delta\mathbf{w}(n). \tag{8}$$

De esa manera, se debe resaltar que el objetivo de estos filtros es buscar la convergencia al mínimo error calculado sobre los datos esperados del modelo, en múltiples iteraciones hasta alcanzar un valor constante de "estado estacionario" [29].

6.4.1. Filtro LMS

Los filtros LMS son un caso de filtros adaptables comúnmente utilizados para el procesamiento de datos. Estos filtros utilizan los métodos descritos anteriormente para el cálculo de predicciones y actualización de pesos considerando un modelo lineal. No obstante, los filtros se caracterizan por calcular el error con mínimos cuadrados medios (*Least-Mean-Square* - LMS) o el criterio de error cuadrático medio instantáneo de los resultados para actualizar sus parámetros acorde a las ecuaciones (9) y (10) [29][30]:

En primer lugar, el error calculado de los resultados se presenta en donde e es el error calculado, d representa la señal deseada, $\mathbf{w}(n\text{-}1)$ es el vector de pesos previos y $\mathbf{x}$ es el vector de entrada al sistema.

$$e(n) = d(n) - \mathbf{w}(n-1)^{\mathrm{T}}\mathbf{x}(n). \tag{9}$$

De la misma manera, para expandir su definición, la actualización de los pesos adquiere la siguiente forma en donde se introducen los términos de tasa de aprendizaje η (*learning*

rate), el error calculado e y el vector de entrada $\mathbf{x}$ en lugar de $\Delta\mathbf{w}(n)$ descrito previamente.

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \eta e(n)\mathbf{x}(n). \tag{10}$$

Es importante notar que el sistema descrito en estos escenarios considera una sola entrada y una sola salida (*Single Input Single Output* - SISO) [29].

6.4.2. Filtro LMS adjunto

La propuesta en el trabajo de [31], describe las ventajas de una variante en el algoritmo de un filtro LMS implementado que busca generalizar los elementos de las ecuaciones definidas previamente para recibir múltiples canales de entrada y múltiples salidas. La propuesta del filtro LMS adjunto (*Adjoint LMS Filter*) permite evitar la complejidad de otros algoritmos para múltiples canales y mejorar la convergencia a las señales deseadas descritas [31].

En este caso, las ecuaciones se generalizan de manera que la actualización de pesos adquiere los términos de $M2$, que se refiere al orden del filtro de respuesta finita al impulso (*Finite Impulse Response Filter* - FIR), y $\tilde{e}$ que representa el resultado de errores calculados para las señales de salida descritas[31].

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\tilde{e}(n-M2)\mathbf{x}(n-M2). \tag{11}$$

De la misma manera, el error actualizado de $\tilde{e}$ está dado según la función $\hat{C}(q^{+1}, k)$ que describe la actualización del error y e como el error actual resultante de la predicción para cada señal deseada.

$$\tilde{e}(n) = \hat{C}(q^{+1}, n)e(n). \tag{12}$$

Esta propuesta de filtro también argumenta que, a diferencia de otros filtros de múltiples canales, este posee la habilidad de presentar convergencia aún en escenarios de modelos no lineales [31]. Así, en el contexto de este trabajo se espera la convergencia para la trayectoria de los agentes en el espacio, independientemente del comportamiento que se presenta del error, considerando que aún se cuenta con las señales deseadas.

6.4.3. Filtro de *kernel* adaptable LMS

El metodo de *kernel* adaptable para los filtros LMS, particularmente lineales, presenta la adopción de distintos modelos y funciones para insertar razgos de no linealidad al sistema, típicamente mediante la reproducción de una función de base radial (*Radial-Basis Function* - RBF) o *kernel* gaussiano descrito en (13) [33]. Esta técnica se emplea para poder realizar una aproximación de los modelos más precisa en cuanto a la convergencia de los pesos, dado el espacio agregado a la dimensión de los datos ingresados en cada iteración durante el entrenamiento [29].

$$\phi(\mathbf{x}, \mathbf{c}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}\|2}{2\sigma^2}\right). \tag{13}$$

En este caso se suele comprender, en conjunto con el concepto de descenso de gradiente previsto para el entrenamiento de las redes neuronales, que este es un modelo que no presenta mínimos locales dado su capacidad para realizar aproximaciones "universales" [29]. Se debe notar que esto sucede para el caso descrito por un *kernel* gaussiano, pero existen variaciones con diversos tipos de *kernel* como el polinomial que presentan distintos beneficios.

De esa manera, las funciones del algoritmo para el filtro LMS conocidas para actualización de pesos y cálculo de error adquieren los términos ϕ y ω considerando la aplicación de función de base radial implementada al vector de entrada. En este caso, ϕ representa el vector de características en la entrada y ω es la estimación de los pesos asociados al filtro [29].

$$\omega(n) = \omega(n-1) + \eta e(n)\phi(n), \tag{14}$$

$$e(n) = d(n) - \omega(n-1)^T \phi(n). \tag{15}$$

6.5. Fusión de sensores

La fusión de sensores es una herramienta que busca integrar las lecturas de varios dispositivos de manera simultánea para generar una única señal con mejor precisión. En la práctica, esta herramienta permite mejorar las mediciones mediante técnicas con filtros adaptables tales como los filtros de Kalman y filtros extendidos de Kalman que se detallarán más adelante.

En este caso, se consideran categorías que describen el comportamiento de los algoritmos empleados para la herramienta de la siguiente manera[34]:

- **Sensores redundantes:** los algoritmos se encargan de recibir la misma información de varios sensores de manera que se discriminan lecturas erróneas del comportamiento.
- **Sensores complementarios:** los algoritmos se encargan de interpretar la información de los sensores que realizan mediciones sobre cantidades independientes para describir el comportamiento en dado contexto.
- **Sensores coordinados:** los algoritmos se encargan de comprender el contexto a partir de la información independiente de los sensores en una secuencia dada.

A menudo, la aplicación de esta tecnología recurre a distintos métodos de acuerdo a la información que presentan los sensores para describir fenómenos físicos y la representación de los datos en un ambiente establecido [35][36][37]. Por motivos de este trabajo, se establece que la herramienta se emplea de manera tradicional para interpretar las lecturas

de aceleración, velocidad angular e intensidad magnética en simultáneo, con el objetivo de reconstruir la pose y trayectoria de diversos agentes en un espacio tridimensional. Así, el método empleado se caracteriza por reducir la ambigüedad de las lecturas generadas por el ruido que experimentan los sensores [38].

6.5.1. Filtros de Kalman

El filtro de Kalman se caracteriza por ser un método para la fusión de sensores empleando un algoritmo recursivo para la predicción del estado asociado a sistemas lineales dinámicos. En este caso, el método original considera la estimación y observación del vector de estados a partir de los valores actuales tanto como la medición de ruido en la señal y la predicción previa del estado en el sistema para reducir el error en los datos generados [39]. Así, el método emplea dos ecuaciones principales para procesar los datos a-priori y a-posteriori para cada iteración de las mediciones reales sobre la señal a la que se expone. Es decir, el filtro emplea las ecuaciones de predicción de estados y observación respectivamente para corregir el comportamiento de las predicciones mediante una ganancia de Kalman variable en el tiempo [40] .

Dado el contexto de los filtros de Kalman, se consideran las siguientes ecuaciones:

Ecuación de predicción de estados

$$X_k = F_{k+1,k} X_k + W_k, \tag{16}$$

en donde X representa el vector de estados en el tiempo k que describen el comportamiento del sistema expuesto al filtro. Así mismo, F es una matriz de transición que asume el valor actual tanto como la predicción de los valores en el vector de estados y W es el ruido que se produce bajo el supuesto que mantiene un comportamiento gaussiano con una matriz de covarianza Q asociada.

Ecuación de medición

$$y_k = H_k X_k + V_k, \tag{17}$$

en donde y son los valores del estado observados en el tiempo k y H es la matriz de observación que se encarga de combinar las mediciones de los sensores mediante transformaciones lineales. De la misma manera, V es el ruido que se presenta bajo el supuesto que mantiene un comportamiento gaussiano con una matriz de covarianza R asociada.

Más adelante, se consideran variaciones del filtro de Kalman que permiten la predicción de los vectores de estado dada la interpretación de sistemas no lineales.

6.5.2. Filtro extendido de Kalman

La introducción del filtro extendido de Kalman se presenta al considerar la capacidad del algoritmo original orientado a comprender el comportamiento del estado en sistemas lineales. Así, se trata de un algoritmo que permite adaptarse a sistemas no lineales a partir de dos etapas de linealización para los valores que se presentan en la estimación del estado. De esa manera, se considera el procedimiento previo a la aplicación de las ecuaciones originales del método para describir el sistema asumiendo la no linealidad en las matrices y posible varianza en el tiempo [39].

En este caso, las etapas de linealización para el filtro extendido de Kalman se presentan de la siguiente manera:

Linealización de matriz de transición $f(k,x)$ y Linealización de matriz de observación $h(k,x_k)$

$$F_{k+1,k} = \left.\frac{\partial f(k,x)}{\partial x}\right|_{x=\hat{x}_k}, \tag{18}$$

$$H_k = \left.\frac{\partial h(k,x_k)}{\partial x}\right|_{x=\hat{x}_k^-}. \tag{19}$$

Aproximación de Taylor de primer orden para las matrices

$$F(k,x_i) \approx F(k,\hat{x}_k) + F_{k+1,k}(x_k,\hat{x}_k), \tag{20}$$

$$H(k,x_i) \approx H(k,\bar{x}_k^-) + H_{k+1,k}(x_i,\bar{x}_k^-). \tag{21}$$

Al considerar las etapas mencionadas, la aproximación del sistema para el algoritmo resulta en las siguientes ecuaciones para predicción de estado y medición.

$$x_{k+1} \approx F_{k+1,k}x_k + w_k + d_k, \tag{22}$$

$$\bar{y_k} \approx H_k x_k + v_k, \tag{23}$$

en donde los valores introducidos a la ecuación representan el vector de observación yk y ruido dk.

$$\bar{y}_k = y_k - h(x_k,\hat{x}_k^-) - H_k\hat{x}_k^-, \tag{24}$$

$$d_k = f(x_k,\hat{x}_k^-) - Fk+1,k\hat{x}_k. \tag{25}$$

6.6. Protocolo TCP/IP

El protocolo de control de transmisión (*Transmission Control Protocol* - TCP) en conjunto con el protocolo de internet (*Internet Protocol* - IP) se emplean en un método para comunicación inalámbrica de información en la red. Así, se comprende que el protocolo de control de transmisión se encarga de ordenar, enviar y recibir los paquetes de información hasta cierto destino descrito por la dirección IP del dispositivo conectado a la misma red [41].

En este caso, el protocolo permite una comunicación en ambas direcciones de los dispositivos conectados de manera que se puede recibir y enviar información simultáneamente en una secuencia, tal como se muestra en la Figura 21. Dado el conocimiento de su funcionamiento, se consideran los términos de servidor y cliente respectivamente en la conexión mediante un puerto conocido como *socket* entre los dispositivos para describir la recepción y emisión de información [41]. Así, la implementación del protocolo en este trabajo resulta en la solución para adquirir la información del microcontrolador ESP32-TinyS3 funcionando como servidor para enviar datos haciendo uso del protocolo descrito.

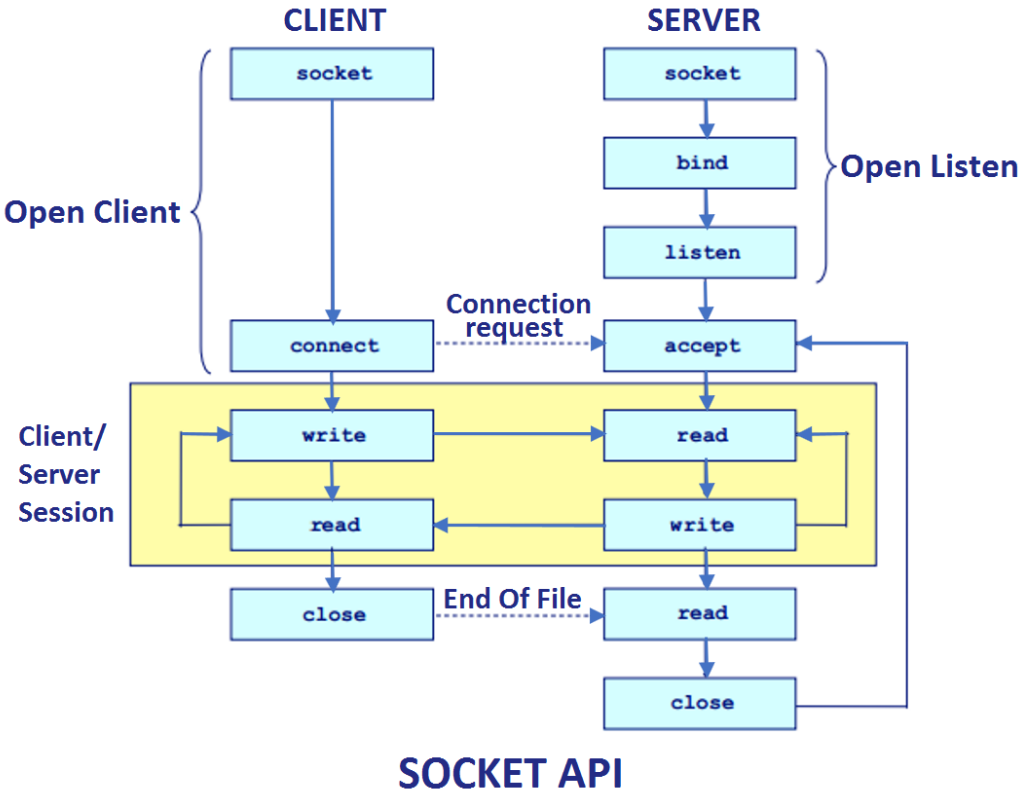

Figura 21: Diagrama de comunicación mediante protocolo TCP [42]

6.7. ESP32 — TinyS3

El TinyS3 es una placa de desarrollo diseñada por Unexpected Maker equipada con un microcontrolador de la serie ESP32-S3 por Espressif y un diseño que facilita explorar sus capacidades para interactuar con conexión a la red en varios proyectos. Además, tal como se muestra en la Figura 22, se considera la capacidad de la placa para conectarse a la red, una conexión con regulador de voltaje a 3.3 V y una conexión vía USB-C para interactuar con la programación del microcontrolador.

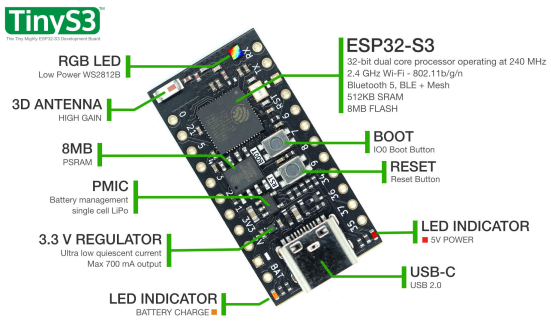

Figura 22: Especificaciones de placa de desarrollo ESP32 - TinyS3 [43]

Dado el modelo de la placa, se presentan las siguientes especificaciones relevantes al proyecto [43]:

- **Microcontrolador:** ESP32-S3FN8
- **Tamaño:** 35 × 18 mm
- **Latencia:** Hasta 240 Mhz
- **Antena / WiFi:** 2.4Ghz b/g/n
- **Conexión:** USB-C con protección ante retroalimentación.
- **GPIO:** 17
- **Batería:** equipado con pads JST compatibles con conector PH para baterías LiPo.

CAPÍTULO 7

Diseño, manufactura y programación de los dispositivos para captura de movimiento inercial

Este capítulo abarca el proceso completo de la creación de los dispositivos utilizados para obtener datos esenciales para un sistema de captura de movimiento inercial, para el cual se consideraron lecturas de una unidad de medición inercial y el sistema de captura Optitrack. En este caso, se resaltó la importancia de esta primera etapa, dada la necesidad de un sistema que permitiera emplear las herramientas disponibles y cumplir con los requerimientos para entrenar los modelos de aprendizaje.

En otras palabras, este capítulo buscó establecer una visión completa del flujo de trabajo para la obtención de conjuntos de datos a través de los dispositivos diseñados para la captura de movimiento inercial asistida por modelos de aprendizaje. Por ello, a continuación se detalla la integración de las tareas que incluyeron el diseño y manufactura de los dispositivos hasta la implementación de los programas encargados con la obtención de datos mediante el servidor del ecosistema Robotat.

7.1. Diseño y ensamblaje de los dispositivos

Un objetivo esencial del proyecto buscó proponer un dispositivo capaz de asistir en la obtención de datos desde el Robotat y los sensores de medición inercial. De esa manera, el contenido en este capítulo considera ciertas tareas principales que ayudaron a definir la estructura del dispositivo y la configuración de los componentes electrónicos en un espacio compacto.

7.1.1. Metodología

Esta sección describe los pasos seguidos para la fabricación de los dispositivos utilizados en la obtención de datos. En particular, se definieron los siguientes requisitos principales:

- Minimizar el espacio ocupado por los componentes para garantizar flexibilidad.
- Asegurar la capacidad de almacenar los elementos esenciales para su asistencia en la captura de movimiento inercial.
- Permitir la instalación de marcadores ópticos personalizados para interactuar con el sistema de captura óptico.

En consecuencia, los dispositivos fueron diseñados para ser empleados como accesorios en las extremidades, como las manos, permitiendo así una mayor flexibilidad durante las sesiones de recolección de datos y favoreciendo movimientos naturales.

Prototipos de los dispositivos

Para comprender el concepto del diseño de los dispositivos, se realizaron los modelos de prototipos que permitieron explorar el espacio disponible. De esa manera, el primer prototipo de diseño se centró en la posición de los componentes que permitieron realizar los primeros experimentos para la obtención de datos de las unidades de medición inercial.

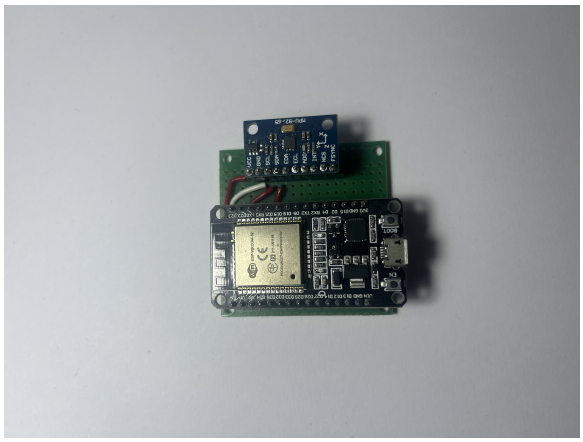

Figura 23: Placa de prototipo para obtención de datos de unidad de medición inercial

Tal como se muestra en la Figura 23, el dispositivo cuenta con los siguientes componentes sobre una placa de soldadura para prototipado:

- **Microcontrolador:** placa de desarrollo ESP32.
- **Módulo de unidad de medición inercial:** MPU-92/65.

Más adelante, se consideró un boceto y modelo preliminar para la estructura de los dispositivos con los componentes del prototipo, de manera que se pudieran encontrar configuraciones para la implementación de los marcadores ópticos. Dado el diseño del prototipo realizado con las herramientas en la plataforma de diseño Onshape[1], mostrado en la Figura 24, se realizó el modelo para la versión final de la estructura para los dispositivos.

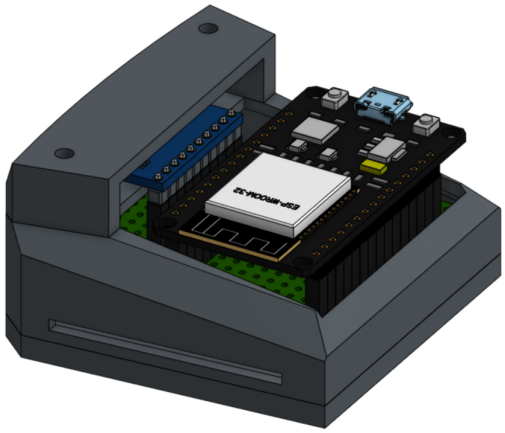

Figura 24: Diseño de estructura para primer prototipo

Diseño y estructura de los dispositivos

Para la estructura de los dispositivos se consideró la configuración de posiciones seleccionadas para los componentes en el diseño final, tanto como el método de sujeción a las extremidades. Dadas las observaciones de los diseños de prototipo, para mantener un factor de forma, se tomó en cuenta la reducción en dimensiones de los componentes. Así, en esta etapa del diseño, se introdujeron componentes como el microcontrolador ESP32 — TinyS3 y una batería de iones de litio (**Li-Ion** 3.7V - 1000mAh) para facilitar movilidad. Cabe resaltar que parte importante de la elección de estos componentes se realizó debido a la compatibilidad y dimensiones reducidas. En este caso, el microcontrolador incluyó un regulador de voltaje para baterías de litio hasta 4.7V para facilitar la conexión.

Así, la idea principal del diseño para los dispositivos se seleccionó de manera que se puedan contener los componentes esenciales para el funcionamiento y permitir la adición de marcadores para cuerpos rígidos del sistema de captura OptiTrack, ilustrados en la Figura 25.

[1] **Onshape:** Es un software para diseño asistido por computadora (*Computer Aided Design* - CAD) que permite el control de versiones en la nube.

Figura 25: Marcador de cuerpo rígido

Así, para concluir con esta etapa, se consideró el método de sujeción para lograr utilizar los dispositivos como un accesorio en las manos. Dado el diseño de la estructura final en la Figura 30, se ensamblaron los dispositivos con el fin de facilitar al usuario portar el sistema con mayor flexibilidad.

Placas de circuito impreso

Al tener la estructura de los dispositivos, se realizó el diseño de las placas de circuito impreso de manera que se pudieran conectar los componentes esenciales y ensamblar en el diseño. En este caso, tal como para la placa de prototipo, se utilizó el circuito para realizar la conexión I2C entre el controlador **ESP32 — TinyS3** y el módulo **MPU-92/65** que se muestra en la Figura 26.

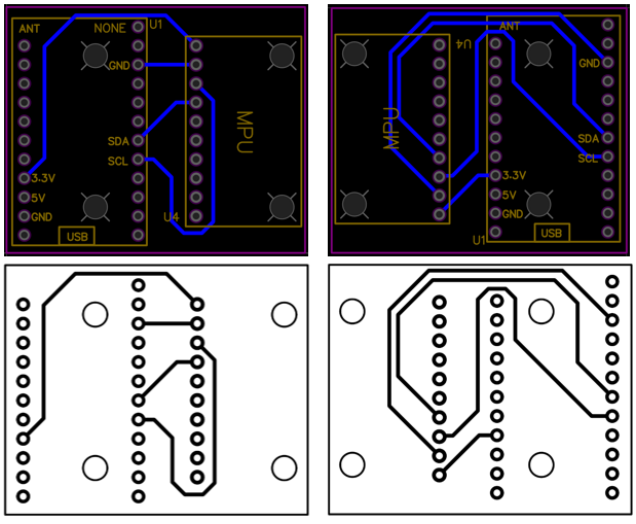

Figura 26: Resultado de conexiones en placas de circuito impreso

Dada la disposición de los materiales se realizó un diseño mediante las herramientas de

EasyEDA [44] y se aplicó el procedimiento de transferencia de tóner y perforación para fabricar la placa. Posteriormente, se realizaron las soldaduras de conectores de pines (*Headers*) sobre los espacios de cobre para completar el circuito impreso ilustrado en la Figura 27.

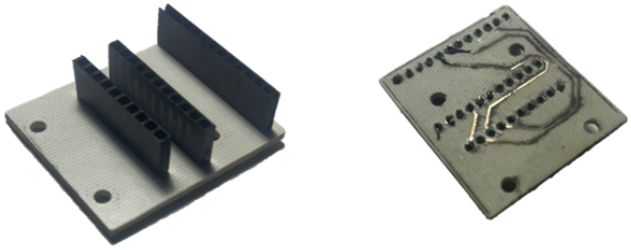

Figura 27: Resultado de placa de circuito impreso derecho

7.1.2. Resultados

Los resultados de este objetivo demostraron las características finales del diseño seleccionado para los dispositivos de asistencia en captura de movimiento. Además, se realizó el análisis del factor de forma obtenido para utilizar el dispositivo mediante el método de sujeción para portarlos como una herramienta para recolección de datos del sistema.

Como resultado principal del objetivo se realizó la primera versión de los dos dispositivos ilustrados en la Figura 28, con el fin de permitir marcadores para captura de movimiento inercial en las manos izquierda y derecha. Cabe notar que en la imagen no se encuentran los marcadores ópticos y en su lugar se añadieron elementos impresos ficticios con las mismas dimensiones relevantes para ocupar el espacio correspondiente.

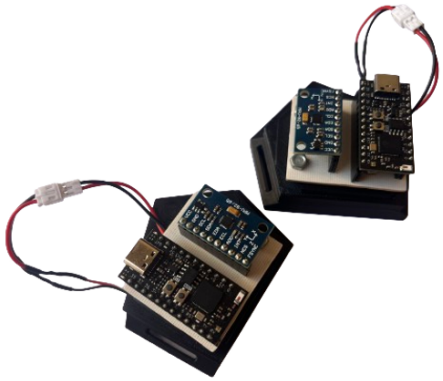

Figura 28: Dispositivos de captura de movimiento inercial

De la misma manera, se consideran los siguientes modelos con las dimensiones del dispositivo en la Figura 29.

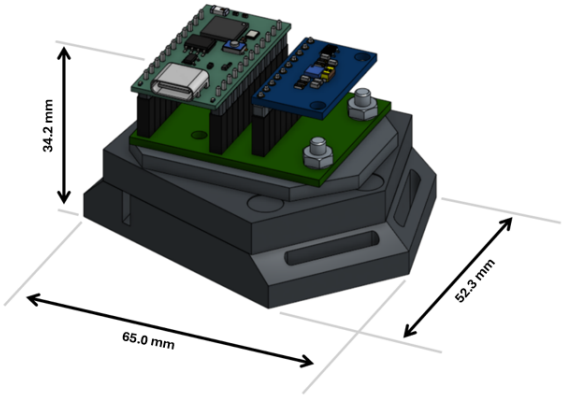

Figura 29: Dibujo con dimensiones de dispositivos en herramienta CAD

El resultado final de los dispositivos se muestra en la Figura 30, en donde se pueden apreciar los marcadores instalados para interactuar con el sistema de captura de movimiento óptico y el método de sujeción.

Figura 30: Dispositivos de captura de movimiento inercial

7.2. Comunicación inalámbrica y obtención de datos

Dado el acceso a las funciones del microcontrolador en el dispositivo y el ecosistema Robotat, la segunda etapa del proyecto descrita en este capítulo explica el funcionamiento del dispositivo para obtención de datos mediante una comunicación inalámbrica. En este caso se consideran los programas que permiten el envío y recepción de datos en distintas tareas, así como la arquitectura cliente-servidor de la comunicación establecida entre dispositivos.

Cabe mencionar la importancia de esta etapa, dado que los datos generados desde los dispositivos son vitales para el funcionamiento de las etapas de entrenamiento y diseño de la red neuronal y herramientas de aprendizaje automático. Por ello, se buscó establecer

controles simples para interactuar con los dispositivos y la extracción de datos ordenada.

7.2.1. Metodología

En cuanto a la metodología empleada para cumplir el objetivo, el sistema diseñado contó con la conexión inalámbrica entre 3 dispositivos: el dispositivo de captura de movimiento inercial, el ecosistema Robotat y una computadora o dispositivo que ejecuta el *script* de Python correspondiente. De esa manera, en esta sección se describen los pasos para adquirir los datos de manera ordenada, la implementación de librerías y la comunicación entre los dispositivos.

Implementación del protocolo de comunicación TCP/IP

Para iniciar la implementación de esta etapa del proyecto, se consideró establecer el protocolo de comunicación inalámbrica TCP/IP para permitir extraer los datos esenciales para el entrenamiento de la red neuronal y herramientas de aprendizaje automático. En este caso, el protocolo se implementó de tal manera que la arquitectura de conexión constó de 3 dispositivos en una configuración de clientes y servidores que se puede apreciar en la Figura 31.

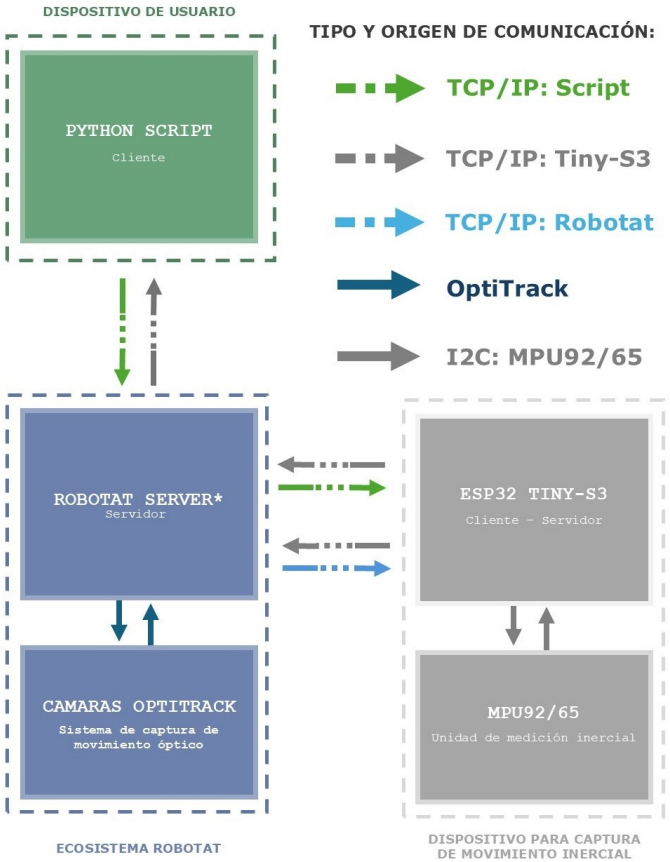

Figura 31: Diagrama de conexiones entre dispositivos

Cabe mencionar que el criterio que se utilizó para implementar el protocolo de comunicación TCP/IP, además de permitir el acceso a los datos del servidor del ecosistema Robotat, también permitió garantizar el envío y recepción de datos de manera ordenada. Así, a partir de la arquitectura de conexión, se pudo definir la configuración de los dispositivos de la siguiente manera:

- **ESP32 Tiny-S3:** servidor-cliente (intermediario).
- **Dispositivo con *Script* de Python:** cliente.
- **Ecosistema Robotat:** servidor.

Estos términos permitieron describir el flujo de datos que se estableció al momento de iniciar el dispositivo para captura de movimiento inercial. De la misma manera, es importante comprender que los datos se envían en un formato JSON[2] que permite asignar etiquetas a los datos enviados para mantener un registro de las lecturas realizadas y garantizar la recepción correcta en el siguiente dispositivo.

Dado el contexto de las conexiones realizadas entre los dispositivos, se debe hacer referencia al sistema implementado que permite al dispositivo intermediario asignar etiquetas sobre el origen de los datos recibidos dentro del mensaje. Este sistema se implementó con el objetivo de permitir a los dispositivos descritos a continuación decodificar y ordenar los datos obtenidos.

Diseño de programa para dispositivos de captura de movimiento

Durante esta etapa, se consideró aprovechar las características del microcontrolador ESP32 Tiny-S3 mediante una arquitectura de programación tal como se muestra en la Figura32 y 33. Para cumplir con el objetivo, se distribuyeron los recursos del controlador de manera que fuera dotado de la capacidad para trabajar ciertas tareas (*tasks*) en sus 2 diferentes núcleos(*cores*), para recibir comandos y enviar los datos correspondientes.

Para lograr este comportamiento en el microcontrolador, cabe mencionar que el programa se realizó mediante el entorno de desarrollo integrado (IDE) de Arduino en C# empleando al esquema de gestión de tareas en tiempo real con *FreeRTOS*. Así, se logró garantizar que el controlador simule un comportamiento lo más cercano posible a la ejecución de las tareas en “paralelo".

De esa manera, el programa se realizó considerando dos configuraciones de modo para generar los primeros conjuntos de datos con las arquitecturas ilustradas en la Figura 32 y 33

[2]**Formato JSON:** (*JavaScript Object Notation* - JSON) Es una estructura de datos ligera que busca ordenar la información para facilitar la legibilidad. Las aplicaciones de este formato son ampliamente utilizadas en comunicación entre aplicaciones y sistemas.

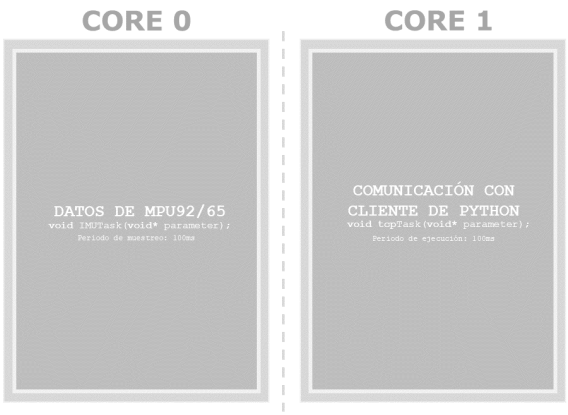

Figura 32: Diagrama de arquitectura para programa en ESP32 Tiny-S3 durante pruebas en MPU-92/65

En el diagrama de la Figura32 se presenta el comportamiento del controlador al recibir el comando para empezar una prueba de obtención de datos exclusiva de los datos del módulo MPU-92/65 mediante la librería *MPU9250_asukiaaa* [45]. En este caso, el controlador se encarga de solicitar, filtrar y enviar muestras de aceleración, velocidad angular y orientación magnética en conjunto con el tiempo en que se tomaron los datos y la etiqueta de origen con una frecuencia de 10Hz mediante el protocolo de comunicación.

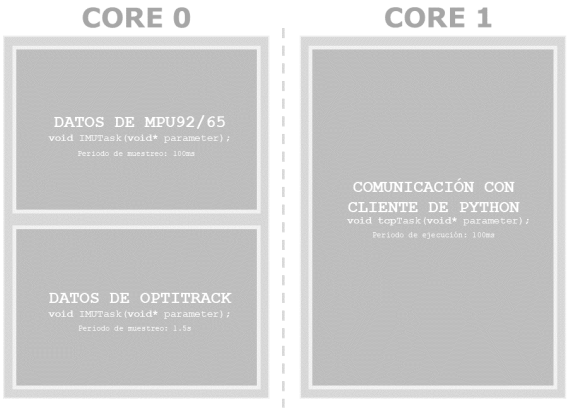

Figura 33: Diagrama de arquitectura para programa en ESP32 Tiny-S3 durante generación de sets de entrenamiento

Por otro lado, el diagrama en la Figura 33 ilustra la distribución de tareas en el controlador al recibir el comando para generar datos de entrenamiento para la red neuronal. En este caso, el controlador se encarga de distribuir la carga de obtención de datos de la unidad de medición inercial y las operaciones para recibir datos del sistema de captura de movimiento óptico. Posteriormente, en las tareas respectivas se asignan las etiquetas de dispositivo origen correspondientes antes de ser enviadas.

Dado el contexto de la arquitectura del programa, se destacó la capacidad del dispositivo

para operar bajo una configuración servidor-cliente, como se mencionó previamente en esta sección. En particular, el programa del dispositivo se realizó con el fin de responder a los comandos enviados por el usuario mediante el *script* en Python, que permitieron establecer el protocolo de inicio y las tareas requeridas para cada caso.

Solicitud de datos del sistema de captura de movimiento óptico

La tarea de obtención de datos que corresponde a la interacción con sistema de captura de movimiento óptico se realizó a través de un proceso de solicitud al servidor del Robotat y decodificación del formato del mensaje. Dada la naturaleza de los datos que se reciben del sistema en cuaterniones, también se implementaron las operaciones para ajustar la presentación a ángulos de Euler mediante la secuencia de rotación ZYX que posteriormente fueron evaluadas en la red neuronal.

En este caso, la solicitud para obtener la pose de un marcador óptico específico al servidor se realiza mediante el siguiente comando, en donde *agent_ID* se refiere al marcador de cuerpo rígido seleccionado:

```
DynamicJsonDocument jsonDoc(2048);
jsonDoc["dst"] = 1; // DST_ROBOTAT
jsonDoc["cmd"] = 1; // CMD_GET_POSE
jsonDoc["pld"] = agent_ID;

String jsonString;
serializeJson(jsonDoc, jsonString);
tcp_obj.print(jsonString);
```

Figura 34: Código de solicitud en formato JSON enviada a través de protocolo TCP/IP

Cabe mencionar que existe una limitante en cuanto al tiempo de respuesta del servidor, por lo que se estableció realizar el proceso de solicitud de la ubicación y pose de los agentes en un periodo establecido por el usuario. Esta característica establece que la respuesta del servidor se recibe a medida que es capaz de enviar los paquetes por el protocolo I2C. De esa manera, fue posible contemplar que la cantidad de muestras para las unidades de medición inercial contra las muestras del sistema de captura de movimiento óptico serían mayores, por lo que posteriormente se hizo una interpolación de los datos para asignar las etiquetas con su respectivo sello de tiempo.

Interacción de usuario con dispositivo para captura de movimiento inercial

Para lograr la interacción con los dispositivos, el programa en Python se realizó con el fin de permitir la interacción de usuario con el dispositivo para captura de movimiento inercial; la recepción y almacenamiento de datos esenciales para el entrenamiento de la red neuronal.

De manera que se pueda describir el proceso para ordenar la información en el programa, es importante hacer mención del sistema de formato implementado para asignar las etiquetas de dispositivo de origen al momento de recibir datos. Así, para el caso de una sesión de obtención de datos para entrenamiento de la red neuronal, se implementó una función que permitió al dispositivo clasificar los mensajes a los que se les hará referencia en las siguientes etapas como datos de entrada(*inputs*) que provienen de la unidad de medición inercial y etiquetas(*labels*) del sistema de captura OptiTrack.

Posteriormente se implementaron las funciones para visualizar y almacenar los datos en un archivo `.csv` de manera ordenada, que se discutirán en los resultados.

7.2.2. Resultados

Está sección abarca los resultados de los datos generados con el dispositivo para captura de movimiento inercial de manera que se pueda comprender la importancia de la calidad de los datos y discutir cómo pueden afectar los resultados en los capítulos que tratan sobre la implementación de los algoritmos para reconstrucción de trayectoria y pose de los agentes.

Conjuntos de datos

De manera general, durante la generación de los sets de datos fue posible notar con la configuración actual del ecosistema Robotat que la resolución de muestras para las etiquetas se vio afectada proporcionalmente por la capacidad en tiempo de respuesta. En este caso, la resolución de los datos generados por los dispositivos se encuentra alrededor de 12 datos de entrada en la unidad de medición inercial a 1 etiqueta del sistema de captura de movimiento óptico, tal como se muestra en el Cuadro 1 para 8 sets de datos con movimientos controlados.

Set de datos	**MPU-92/65 (*input*)**	**OPTITrack (*labels*)**	**Factor de escala**
Círculos y ondas	9770×10	840×10	12:1
Círculos	8700×10	750×10	12:1
Cubo	25600×10	2230×10	11:1
Líneas rectas	13440×10	1170×10	11:1
Intervalo de líneas rectas	11590×10	1000×10	12:1
Pulsos	10810×10	930×10	12:1
Intervalo de pulsos	11010×10	950×10	12:1
Cuadrado	10470×10	890×10	12:1

Cuadro 1: Dimensiones para conjuntos de datos controlados

Cabe mencionar en este momento que en los próximos capítulos se explora la manipulación de los datos disponibles en los conjuntos de entrenamiento que pueden afectar la cantidad de muestras. En este caso, esto se realiza de manera que se pueda evaluar como afectarán diferentes presentaciones de las lecturas en la estimación de trayectorias y pose de los agentes. También se debe resaltar que futuras menciones de los conjuntos de datos consideran la información en las figuras presentes en Anexos como referencia de las lecturas generadas por las unidades de medición inercial.

Además, se debe mencionar que se realizaron conjuntos de datos adicionales adjuntos en la documentación del trabajo presente en los Anexos. En este caso, se trata de los conjuntos de datos realizados durante el desarrollo del dispositivo descrito en el capítulo anterior con la característica de presentar trayectorias más extensas y comportamientos naturales en el movimiento de las manos.

Tal como se muestra en la Figura 35 y 36, los datos se generaron de manera ordenada y el programa permitió visualizar el comportamiento que presentan las diferentes magnitudes medidas tanto en el sensor como en el sistema de captura de movimiento óptico.

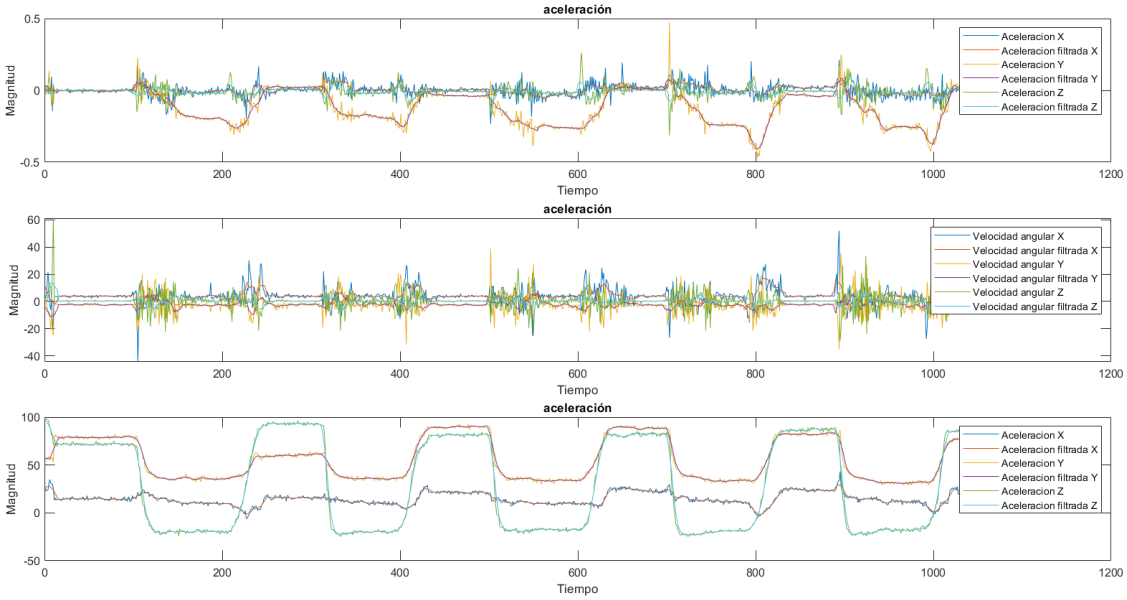

Figura 35: Representación visual de señal generada por datos de módulo MPU92/65 para set de datos de intervalo de pulsos

Dado el comportamiento de las señales en la Figura 35, fue posible notar que con las configuraciones actuales se introdujo ruido en todas las mediciones de la unidad de medición inercial. En este caso, fue posible estimar los siguientes promedios de desviación respecto a los sets de datos correspondientes al Cuadro 2.

Set de datos	A.X	A.Y	A.Z	V.X	V.Y	V.Z	M.X	M.Y	M.Z
Círculos y ondas	0.09	0.26	0.12	13.04	12.42	16.34	24.64	26.94	23.56
Círculos	0.08	0.12	0.03	4.54	6.36	10.57	27.46	24.37	17.99
Cubo	0.06	0.10	0.04	5.14	5.30	8.48	24.14	29.25	16.50
Líneas rectas	0.06	0.10	0.04	6.14	6.16	20.58	25.02	24.03	18.96
Intervalo de líneas rectas	0.07	0.12	0.04	8.19	7.58	21.56	26.38	32.71	19.09
Pulsos	0.04	0.10	0.04	5.59	4.78	4.08	7.25	15.76	51.69
Intervalo de pulsos	0.04	0.12	0.04	6.48	6.17	5.79	6.61	21.93	48.69
Cuadrado	0.09	0.21	0.07	11.74	13.15	16.28	25.07	23.57	16.78

Cuadro 2: Desviación estándar de los valores de la unidad de medición inercial: aceleración (A.), velocidad angular (V.) y orientación magnética (M.)

Para los escenarios descritos es importante recalcar que, adicional al procesamiento de los datos en bruto, también se consideró el caso en que los datos fuesen previamente filtrados mediante un promediador móvil de manera que se pudiera suavizar la señal de las magnitudes generadas por el sensor. De la misma manera, se consideró la implementación de estas características para evaluar el efecto de las señales en los algoritmos discutidos en los capítulos 8 y 9.

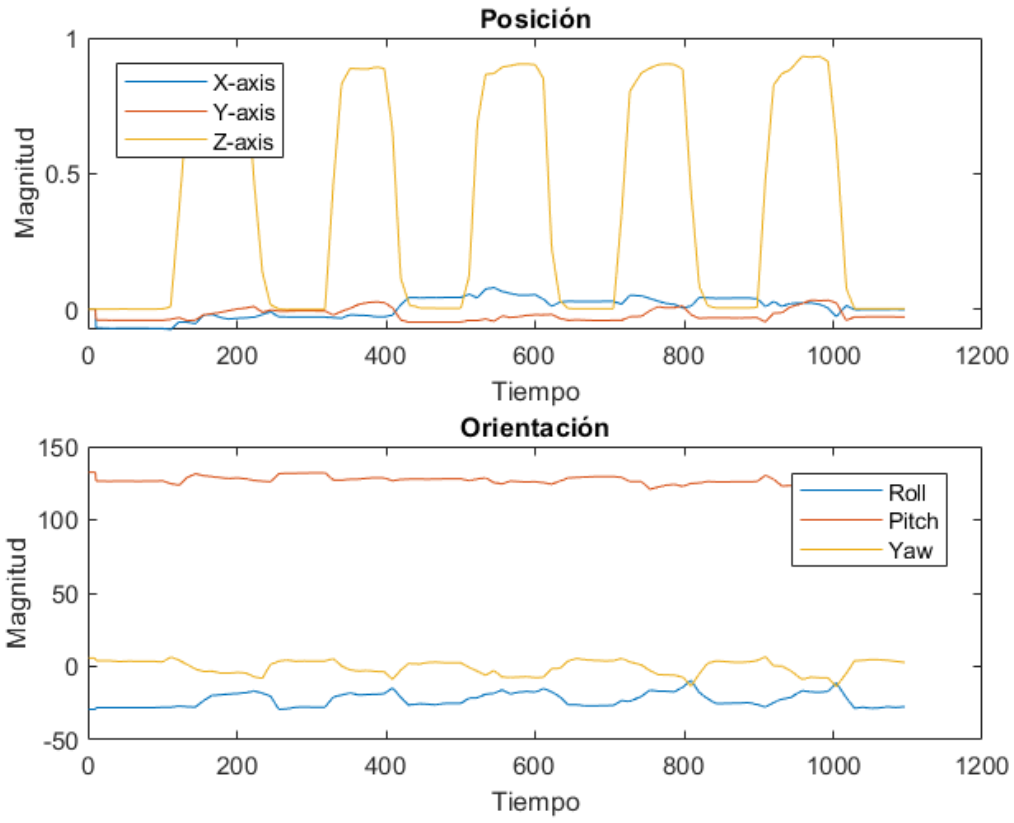

Figura 36: Representación visual de señal generada por datos de módulo MPU92/65 para set de datos de intervalo de pulsos

Por otro lado, la señal generada por el sistema de captura de movimiento óptico permitió describir la secuencia de coordenadas y orientación del marcador a través del set de datos. Dado el tiempo de respuesta que se consideró para estas etiquetas, fue posible apreciar en la Figura 36 cómo se obtuvo una resolución en la señal menor en comparación a la señal generada por los valores en la unidad de medición inercial.

7.3. Observaciones sobre los dispositivos

En estas etapas del trabajo, el diseño y desarrollo de los dispositivos para captura de movimiento inercial permitió establecer en flujo de trabajo para asistir en la obtención de datos. Este proceso involucró la creación y programación de los dispositivos que permitieron la extracción de datos de unidades de medición inercial y el sistema de captura Optitrack a una tasa sujeta a la velocidad propuesta por el servidor del ecosistema Robotat para recibir y enviar paquetes de información de las cámaras. Sin embargo, el esfuerzo concluyó con un método robusto para la captura de los datos escenciales empleados posteriormente para el entrenamiento de los modelos. Así, se resalta que la obtención de datos se realizó empleando solamente un dispositivo a la vez, con el propósito de simplificar la información recopilada e invertir el tiempo en la exploración de los modelos de aprendizaje discutidos a continuación.

En este contexto, el objetivo principal de esta etapa se cumplió al integrar los resultados discutidos en el capítulo con el fin de generar conjuntos de datos ordenados. En esta instancia, se definió que el formato de los conjuntos de datos eran ideales para iniciar la exploración de modelos de aprendizaje en arquitecturas de redes neuronales profundas, caracterizadas por generar las relaciones esenciales entre los datos para resolver problemas complejos de optimización. Este enfoque se remarcó más adelante, dado que se asumió ese caso para la problemática que enfrenta la línea de investigación en cuanto a la estimación de pose y trayectoria a partir de los datos de unidades de medición inercial.

CAPÍTULO 8

Arquitecturas de redes neuronales profundas

Las arquitecturas de redes neuronales descritas en este capítulo se realizaron con el objetivo de implementar herramientas de aprendizaje profundo para asistir en la estimación de pose y trayectoria. En particular, la implementación de los modelos se realizó con el propósito de generar un sistema de captura de movimiento capaz de funcionar con lecturas generadas por unidades de medición inercial, a partir de su entrenamiento con las lecturas generadas por el sistema de captura de movimiento óptico.

En este contexto, la selección de las arquitecturas se basó en el uso de capas covolucionales, cuya justificación en este trabajo se presenta mediante su capacidad para extraer características relevantes en conjuntos de datos ordenados. Asimismo, se exploraron variantes de estas arquitecturas, según su capacidad para establecer conexiones que contemplen la noción temporal de los datos en serie. Así, dado que se trató con los datos generados por las unidades inerciales, se esperó que las herramientas en los modelos pudieran aprender relaciones entre los 9 canales correspondientes a aceleración, velocidad angular e intensidad de campo magnético sobre los tres ejes, con el fin de reconstruir los movimientos registrados en los conjuntos de datos descritos en el capítulo 8.

De ese modo, como estándar para los modelos descritos en este capítulo, un aspecto crítico en el diseño describe la manipulación de los conjuntos de datos para generar los tensores de entrada. Dado el enfoque sobre la noción temporal de los conjuntos para reconstruir las trayectorias, el término de una "ventana de tiempo" móvil se atribuye a la serie de datos contenida en cada paquete de información ingresada a la red. Esto con el fin de facilitar la comprensión de la evolución en las secuencias de datos previos hasta el valor actual, tal como se ilustra en la Figura 37 y la Figura 38.

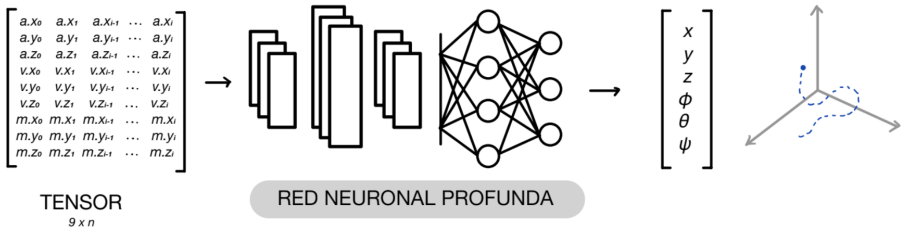

Figura 37: Diagrama de funcionamiento de arquitecturas de red

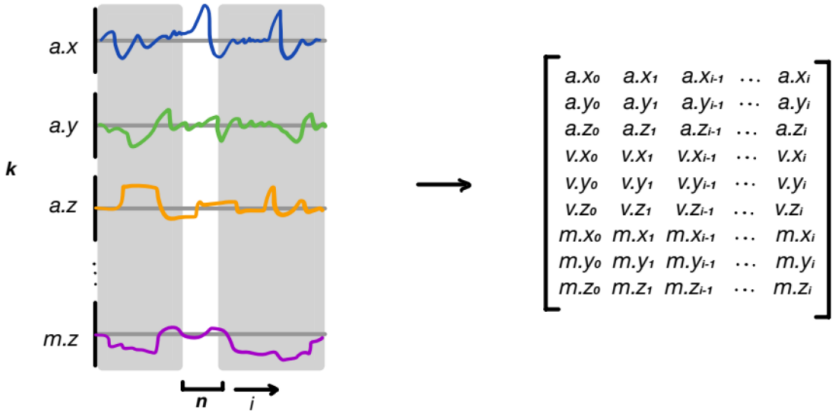

Figura 38: Construcción de tensores de entrada a arquitecturas de red

De esta manera, se comprende que los tensores se definieron como elementos de tres dimensiones que contienen información de una secuencia de datos ordenada para cada canal de la unidad inercial. Posteriormente, este formato permitió hacer la manipulación de los tensores internamente en los modelos propuestos de acuerdo a las operaciones realizadas y requerimientos de las capas que los componen.

Así, a continuación, cada modelo propuesto se evaluó en función de su capacidad para aprender patrones dentro de las secuencias de datos mediante la asignación de los parámetros ajustables característicos. Esto, a su vez, se realizó con el fin de evidenciar el desempeño de los modelos y su capacidad para realizar predicciones sobre el sistema deseado.

Con el fin de permitir una referencia para la evaluación de las características de cada red, se realizó el análisis de los modelos entrenados con los mismos conjuntos de datos. Principalmente, se consideró el caso de un conjunto de datos con movimientos rectos 76 para generar las métricas y predicciones del modelo. Así, debido a que no se logró un obtener un modelo que cumpliera con la predicción precisa de las trayectorias y poses, se realizó el análisis basado en los parámetros que generan los mejores resultados para los conjuntos seleccionados en lugar de establecer un estándar para ajustarlos garantizando resultados satisfactorios.

8.1. Red convolucional unidimensional

El primer acercamiento en el diseño de las arquitecturas de redes neuronales para la estimación de pose y trayectoria de los agentes se basó en la implementación de capas convolucionales unidimensionales y capas densas con el fin de procesar la información en dos etapas principales. Esta estructura, a su vez, se empleó con el fin de permitir al modelo realizar las etapas de predicción y corrección para procesar los conjuntos de datos en serie. Tal como se ilustra en la Figura 39, la red se organizó en módulos de tal manera que se permite realizar las conexiones entre los datos generados por las unidades de medición inercial.

Cabe resaltar que esta arquitectura se inspiró en las funciones principales de los filtros de Kalman, ampliamente utilizados en métodos tradicionales para asistir en captura de movimiento inercial y otras aplicaciones, debido su capacidad para aproximar sistemas desconocidos. En este caso, se deseó integrar la capacidad de estos filtros para realizar predicciones, así como la habilidad de manejar conjuntos de datos mayormente estocásticos mediante los módulos de predicción y corrección cuyas capas se seleccionaron para reconstruir las señales deseadas a partir de los datos de las unidades inerciales.

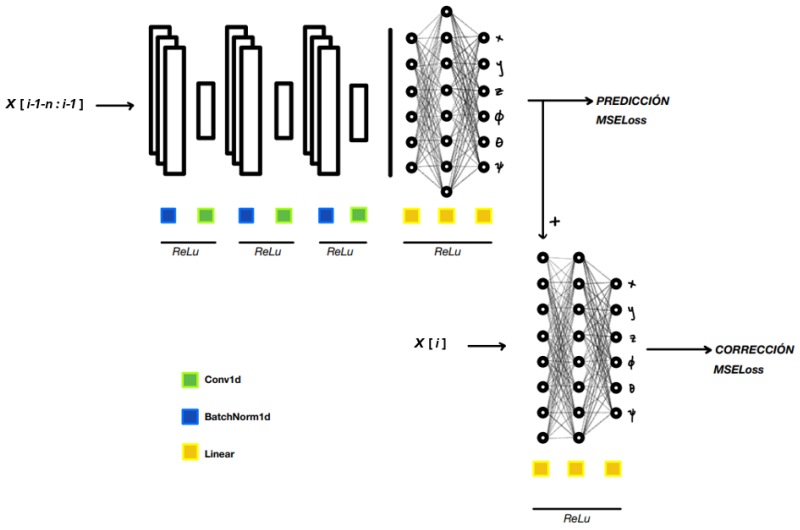

Figura 39: Diagrama de arquitectura de red convolucional unidimensional

Así, para distinguir entre los valores actuales y valores previos dentro de la ventana de tiempo, se empleó un método para procesar los datos según los requisitos de los módulos de corrección y predicción, tal como se muestra en la Figura 40. Esta estructura, a su vez, permitió establecer un método en que ambos pudieran operar de manera independiente, facilitando la incorporación de herramientas adicionales y mejorando la eficiencia del modelo para trabajar con los tensores en la entrada.

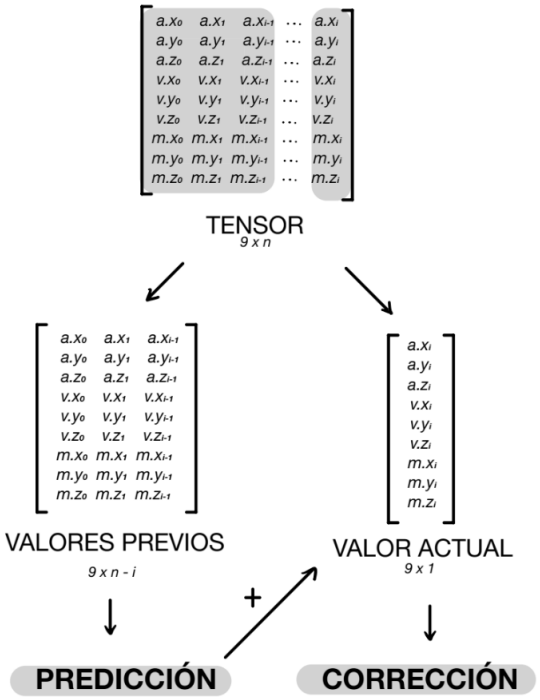

Figura 40: Diagrama de tensores ingresados a arquitectura de red convolucional unidimensional

Por ello, en primer lugar, se optó por ingresar los valores previos dentro de la ventana al módulo de predicción con el fin de obtener las características de las series de datos mediante las capas convolucionales unidimensionales. Así, este módulo obtuvo el objetivo principal de procesar la información previa para generar una predicción sobre los elementos que describen las coordenadas en los ejes correspondientes y la orientación. A su vez, este módulo se realizó de acuerdo un criterio de error cuadrático medio (MSE) dado el valor deseado en el tiempo anterior que se estableció como etiqueta de referencia para el entrenamiento del modelo.

Por otro lado, mediante la concatenación ilustrada en la Figura 40 y el entendimiento de la forma de la salida del modelo, se definió que el valor actual se debía ingresar al módulo de corrección con el fin de ajustar el comportamiento junto con la predicción realizada en el módulo previo. Este comportamiento se asemeja al funcionamiento de un bloque temporal dado el salto de las primeras capas convolucionales de predicción. De esa manera, el modelo se realizó con el fin de generar una respuesta, en forma de un vector $\mathbf{x}$ ajustado a la salida del modelo, empleando un criterio de error cuadrático medio contra el valor actual deseado en un formato idéntico a la salida del módulo de predicción:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}$$

,

en donde $\mathbf{x}$ resulta en un vector con las dimensiones 6×1; x, y, z son las coordenadas espaciales, y ϕ, θ, ψ son los ángulos roll, pitch y yaw en el tiempo actual respectivamente.

8.1.1. Resultados

Dado el contexto del funcionamiento del modelo empleando una arquitectura basada en los filtros de Kalman, los resultados de la implementación de la red sobre el conjunto de datos disponible demostraron su capacidad de aproximar su respuesta con un promedio de error absoluto medio de 0.68m y 19.99° en la predicción final. Cabe mencionar que estos resultados se obtuvieron tras ajustar los parámetros de la red para los conjuntos de datos como punto de referencia. Sin embargo, el mismo comportamiento no se asume para otros conjuntos de datos debido a la complejidad y características del movimiento que se presentan.

A pesar de los resultados en término de exactitud, se observó que no hubo una convergencia satisfactoria para la estimación de la trayectoria de los agentes, tal como se muestra en la Figura 41.

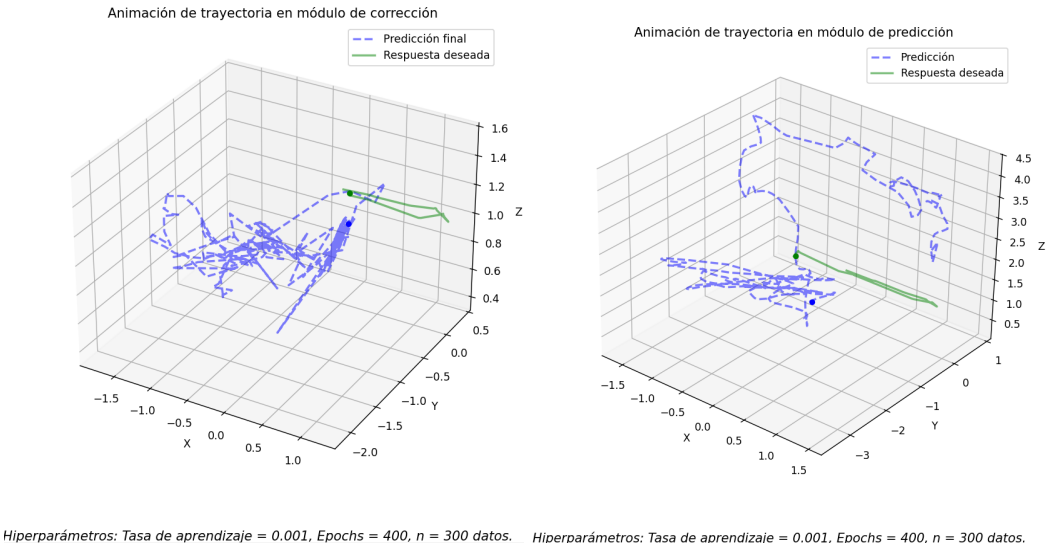

Figura 41: Estimación de trayectoria en módulo de corrección de red convolucional unidimensional

Más a detalle, en las etapas de predicción y corrección se observó el potencial del modelo para capturar la tendencia general de la señal deseada, aunque con un promedio de error significativo considerando que se obtuvo aproximadamente un metro de error respecto a la posición real. Así, en varios escenarios, tal como se evidencia en la Figura 42 y la Figura 43, estas etapas fueron capaces de realizar cierta extracción de características y generar relaciones entre los datos de las unidades inerciales para reconstruir la trayectoria y pose de los agentes. En particular, se observó que el modelo fue capaz de realizar la extracción de orientación efectivamente aún en presencia de alteraciones en las señales deseadas.

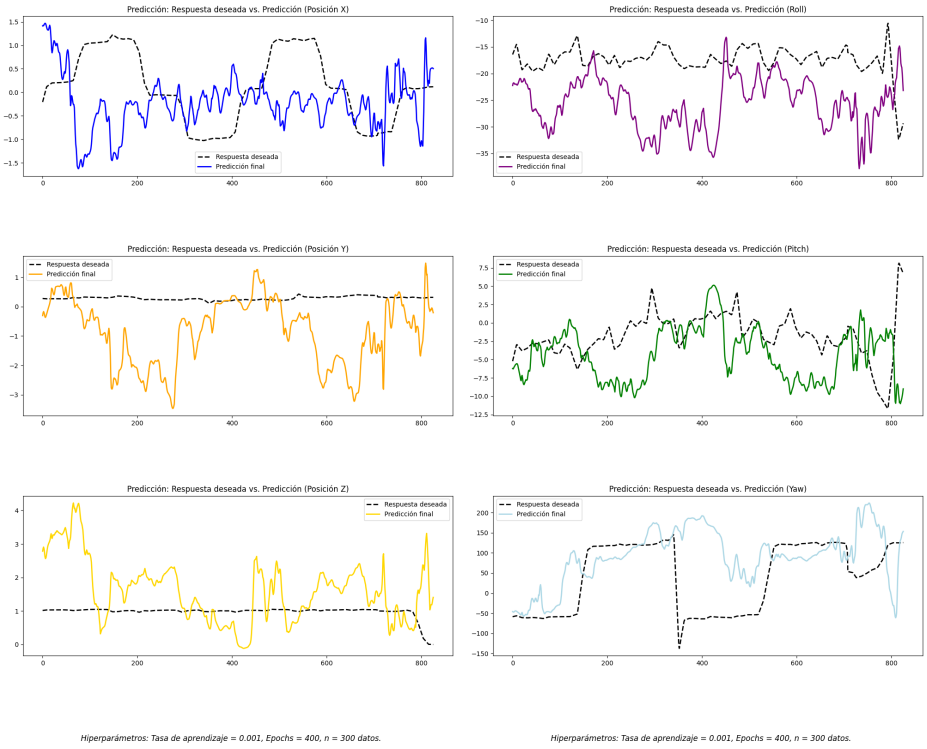

Figura 42: Estimación de pose en módulo de predicción de red convolucional unidimensional

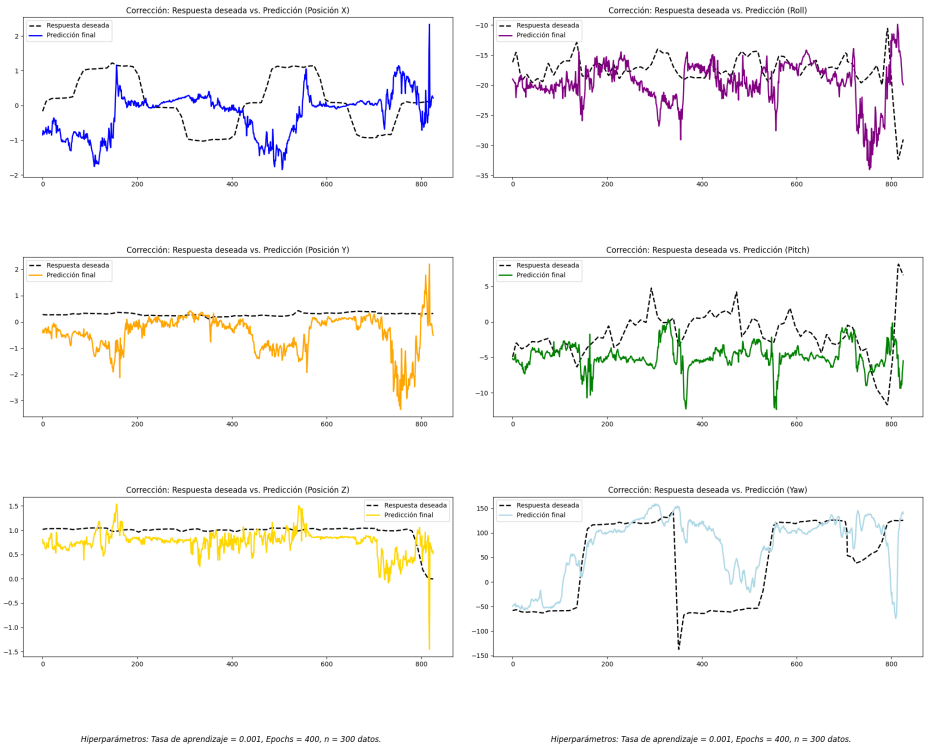

Figura 43: Estimación de pose en módulo de corrección de red convolucional unidimensional

Además de los resultados obtenidos durante la elaboración del modelo, se realizó un análisis más profundo de las características durante el entrenamiento de las redes, en donde se observaron las curvas de aprendizaje y precisión, tal como se presenta en la Figura 44. Esto se realizó con el objetivo de analizar la capacidad del modelo respecto a los parámetros configurados por lo que se identificó que el "pico" representativo al inicio se presenta de manera general a través de los conjuntos de datos y se consideró como el error generado en el tiempo requerido en épocas para ajustar los gradientes de la red.

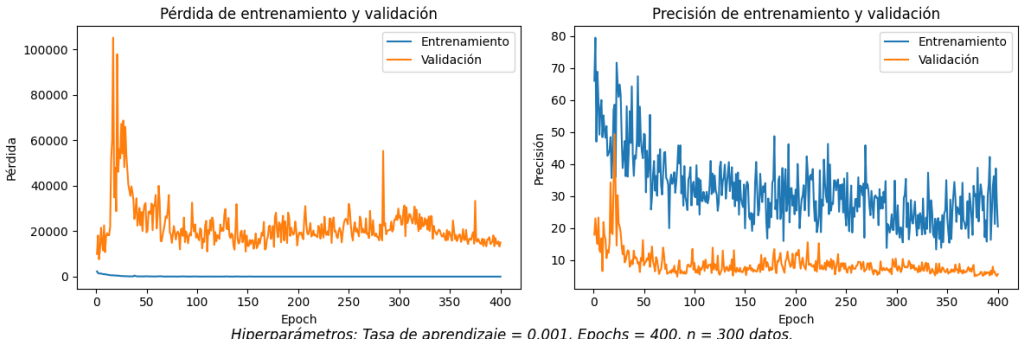

Figura 44: Curvas de aprendizaje y precisión de red convolucional unidimensional

Así, se discutió sobre la posibilidad de agregar herramientas para la integración de la noción temporal de la serie de datos. En este escenario, se consideró que la implementación de tales herramientas permitió facilitar el entrenamiento de la red al agregar dependencias en distintas porciones de los conjuntos de datos relevantes para establecer la relación entre las lecturas generadas por las unidades de medición inercial.

8.2. Red residual con mecanismo de atención

Partiendo de la estructura discutida en la sección anterior, la red residual con mecanismo de atención se realizó con el objetivo de mejorar el rendimiento de estimación de pose y trayectoria de los agentes. Así, la arquitectura del modelo se diseñó con el fin de evaluar la adición de herramientas, en el contexto de aprendizaje profundo, para buscar un comportamiento más robusto ante los conjuntos de datos secuenciales mediante la ponderación de relevancia de porciones específicas y el comportamiento aprendido sobre el movimiento.

De esa manera, tal como se presenta en la Figura 45, la elaboración de la arquitectura de red se conformó por una estructura similar a la propuesta en la sección anterior, basada en las funciones de los filtros Kalman, para realizar predicción y corrección sobre los conjuntos de datos con la adición de bloques residuales. Estos bloques, a su vez, se implementaron con el fin de establecer conexiones secuenciales más complejas, sumando las salidas de capas intermedias de convolución unidimensional mediante un "salto" de la entrada directo al residual acumulado, tal como se ilustra en la Figura 46.

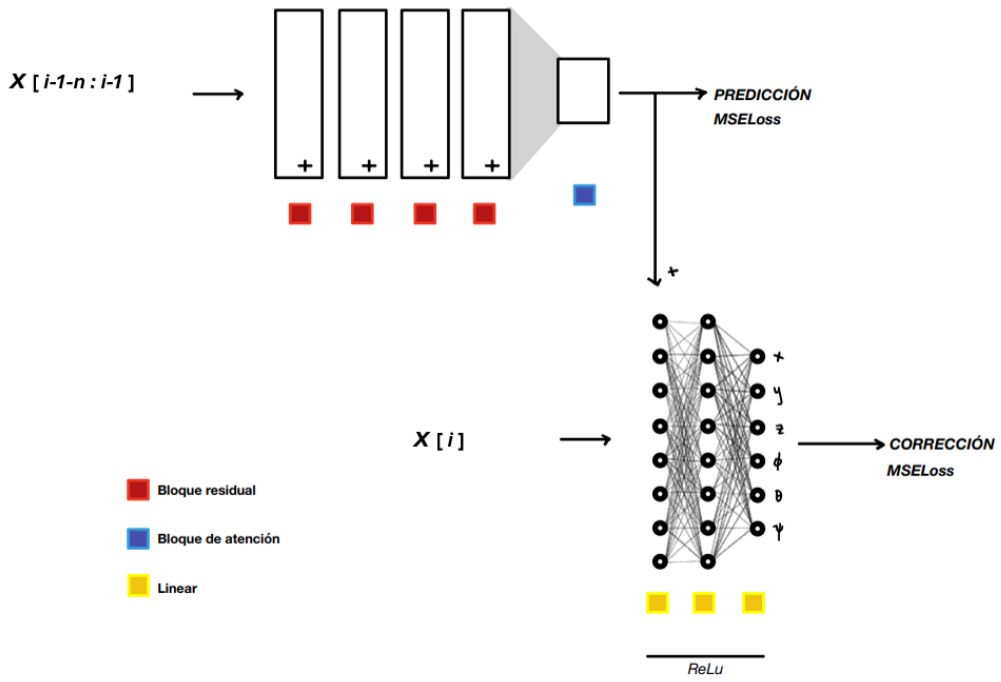

Figura 45: Diagrama de arquitectura de red con mecanismo de atención

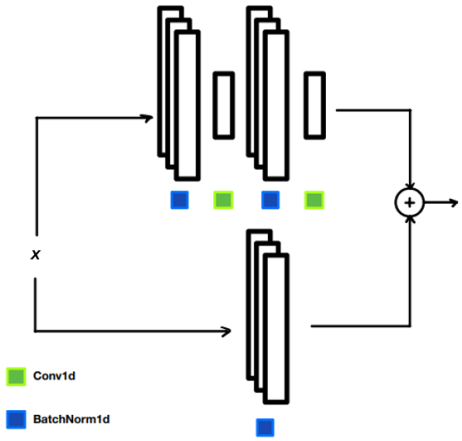

Figura 46: Diagrama de funcionamiento para bloque residual

Además, se incorporó un mecanismo de atención basado en el producto punto escalado, al que se le atribuyó como propósito ponderar ciertas características extraídas de los conjuntos de datos. Este mecanismo, a su vez, se implementó para incrementar la capacidad de generar una noción temporal de características importantes aprendidas en distintas porciones para los canales disponibles.

8.2.1. Resultados

Dados mejores parámetros evaluados durante el entrenamiento de la red neuronal, con el fin de generar un punto de referencia, los resultados de la red residual propuesta no obtuvieron una mejora significativa de precisión en la posición a diferencia de la orientación. En este escenario, se obtuvieron respuestas con un promedio de error absoluto medio de 0.60m y 9.2° en la predicción final del módulo de corrección.

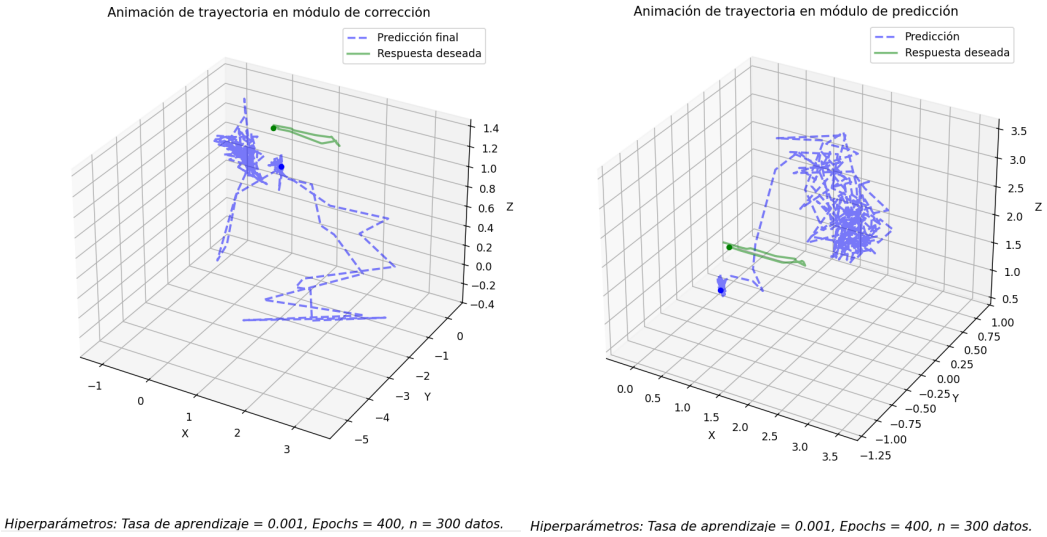

Figura 47: Estimación de trayectoria en módulo de corrección y predicción de red residual con mecanismo de atención

Al igual que el modelo anterior, se observó el comportamiento de la trayectoria y se concluyó que no es posible afirmar que el modelo es capaz de converger de manera precisa a la trayectoria de los agentes, tal como se ilustra en la Figura 47. Así, se analizó el detalle de las respuestas en donde se identificó que, dados los mismos parámetros utilizados para la red convolucional unidimensional, constantemente se generaron respuestas con menor capacidad de convergencia a las señales deseadas, tal como se muestra en la Figura 49 y la Figura 48. Particularmente, este comportamiento se hizo notable en el caso de la orientación dada las características de la predicción de orientación.

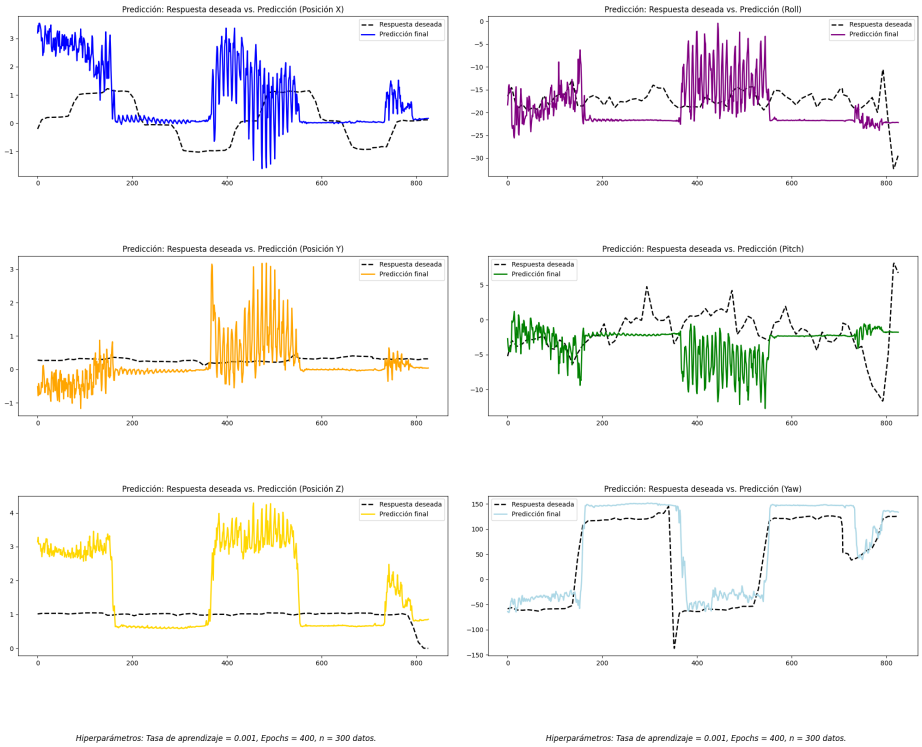

Figura 48: Estimación de pose en módulo de predicción de red residual con mecanismo de atención

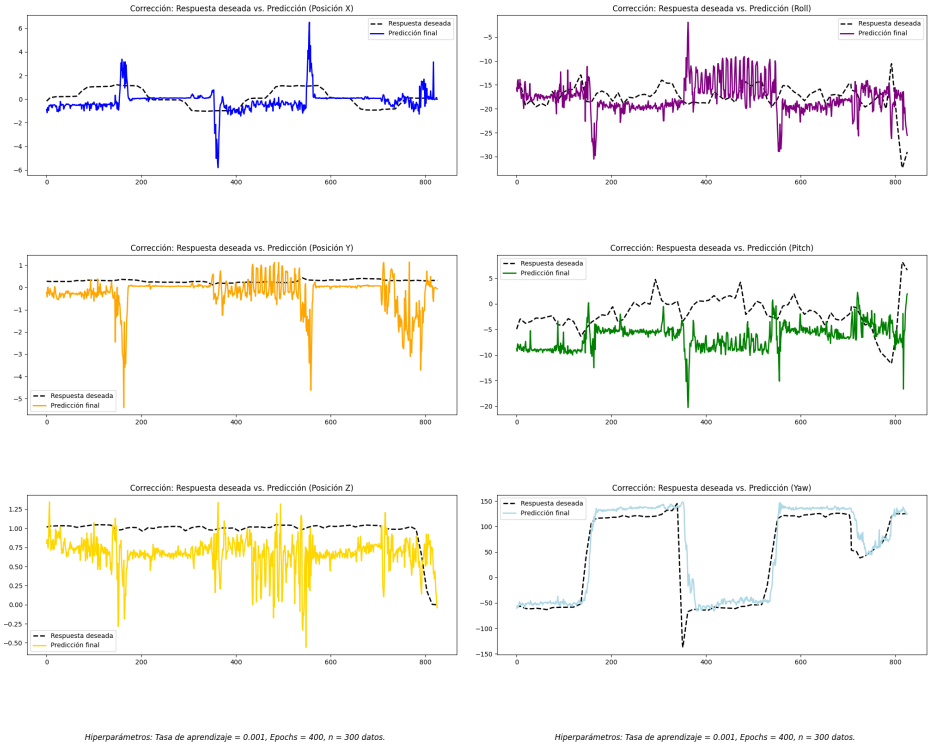

Figura 49: Estimación de pose en módulo de corrección de red residual con mecanismo de atención

Adicionalmente, el análisis de las curvas de aprendizaje y precisión demostraron una tendencia general descendente con notable disminución de pérdida y estabilización del error alcanzado las 100 épocas tal como se ilustra en la Figura 50.

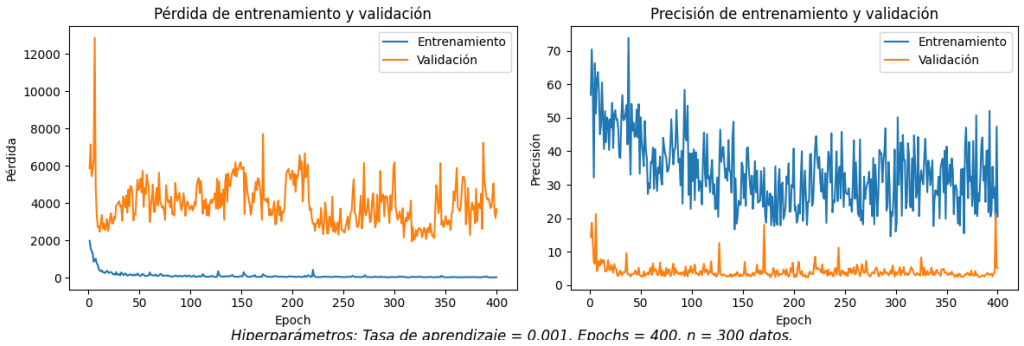

Figura 50: Curvas de aprendizaje y precisión de red residual con mecanismo de atención

A raíz de estos resultados, se discutió sobre la posibilidad de ajustar los hiperparámetros y aprovechar las herramientas del modelo para realizar un entrenamiento más extenso. Por ello, tras la obtención de estos resultados, se observó que existe una tendencia en la aproximación a los datos a las señales deseadas de orientación. Sin embargo, dado el comportamiento estocástico de los resultados del modelo, se definió que sería necesario un estudio a fondo de los hiperparámetros para comprobar la capacidad de predicción de estos elementos en el problema de captura de movimiento inercial planteado.

8.3. Red convolucional temporal

La arquitectura de red convolucional temporal propuesta se basó en la capacidad de este tipo de redes para procesar secuencias de datos, aprovechando este comportamiento para aprender patrones en el dominio de tiempo y realizar conexiones entre las características de los conjuntos. Este enfoque se realizó mediante el vínculo con la idea de los bloques residuales discutida previamente, con la adición de dilatación entre las capas convolucionales con el fin de capturar dependencias a lo largo de las secuencias de datos.

En particular, la arquitectura de este modelo se realizó bajo el entendimiento de bloques temporales, con el fin de procesar los datos en un estilo similar al modelo con capas convolucionales unidimensionales discutidos en la primera sección del capítulo en donde se dosifica el valor inicial en capas posteriores de la red. Así, el objetivo de la red propuesta fue evaluar su capacidad para capturar las dependencias y generar una sola respuesta sobre la estimación de pose y trayectoria.

Más a detalle, la implementación de la red propuesta para generar la conexión entre características temporales del conjunto de datos se realizó de acuerdo con una construcción basada en cuatro bloques temporales iniciales y una capa densa a la salida. De la misma manera, los bloques temporales se realizaron con el fin de permitir el "salto" de la entrada a la suma de un residual acumulado para establecer las relaciones temporales entre las series de datos.

Cabe mencionar que, para el caso del modelo propuesto, se consideró el uso de la entrada para generar la predicción solamente sobre el valor actual. Esto quiere decir que se empleó la serie de datos completos por cada tensor de entrada para generar una respuesta en lugar de incluir módulo de predicción y corrección.

8.3.1. Resultados

Dado el contexto de la implementación de red convolucional temporal, en la prueba realizada sobre el mismo conjunto de datos con movimientos rectos, se obtuvo un promedio de error absoluto medio de 0.61m y 26.63° en posición y orientación respectivamente. En este caso, tal como para los casos anteriores, se pudo identificar que no hubo una mejora significativa respecto a los modelos anteriores.

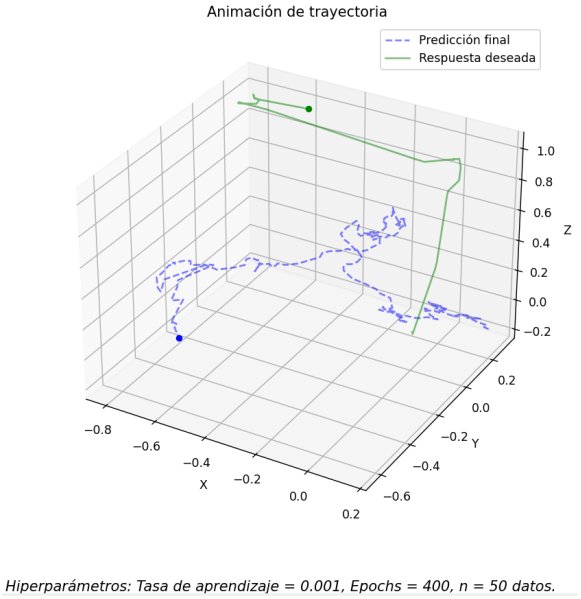

Figura 51: Estimación de pose con red convolucional temporal

No obstante, tal como se ilustra en la Figura 51, se pudo observar que el movimiento en la trayectoria obtenida presentó mayor "suavidad" respecto a los escenarios anteriores. Más a detalle, este comportamiento se evidenció al considerar cómo la extracción de características presentó una tendencia con menor ruido y resolución hacia las respuestas deseadas del sistema, tal como se muestra en la Figura 52. Cabe mencionar que se observó el mismo comportamiento con mayor claridad al desplegar el modelo con conjuntos de datos más extensos, tal como se ilustra en la Figura 87 para los mejores casos de predicción de los modelos entrenados.

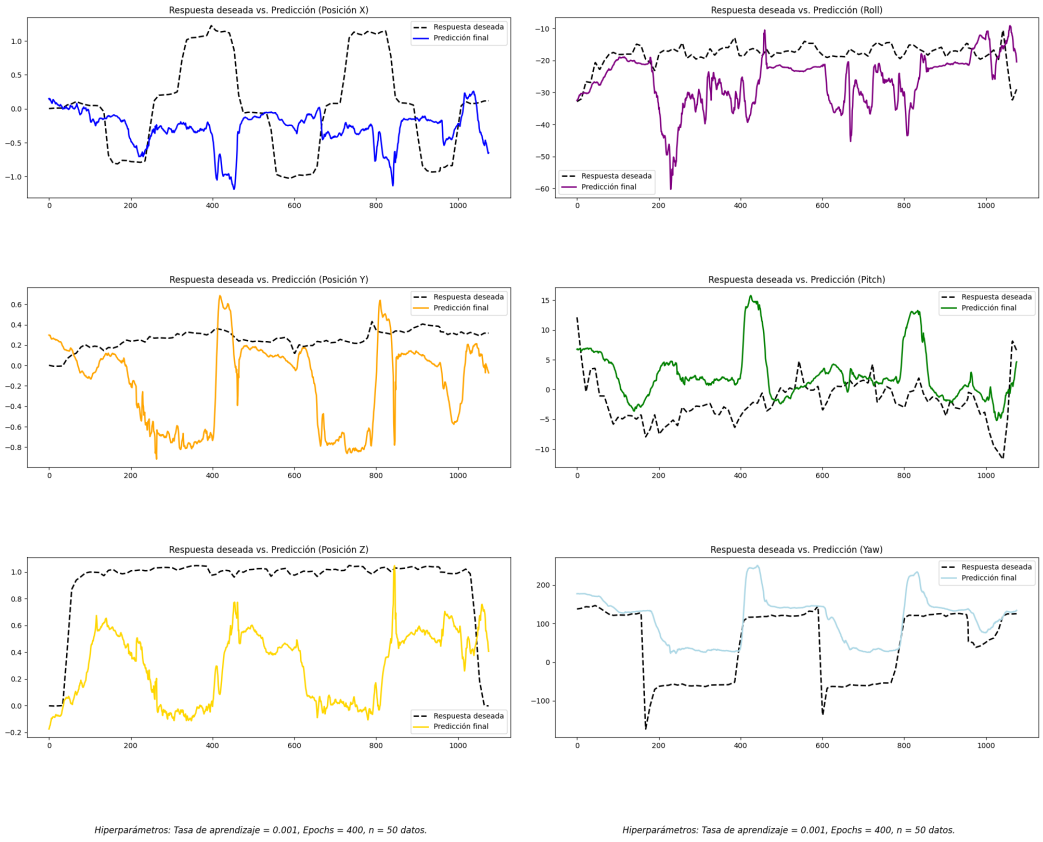

Figura 52: Estimación de pose con red convolucional temporal

Por otro lado, se discutió sobre las curvas de aprendizaje y precisión del modelo presentes en la Figura 53, con la característica de presentar una tendencia en descenso durante la mayor porción de entrenamiento y un estado con mayor estabilidad después de 100 épocas. Este comportamiento plantea la idea de continuar explorando la capacidad de aprendizaje de los modelos de redes temporales convolucionales dada la habilidad de generar las conexiones a través de las herramientas durante el entrenamiento.

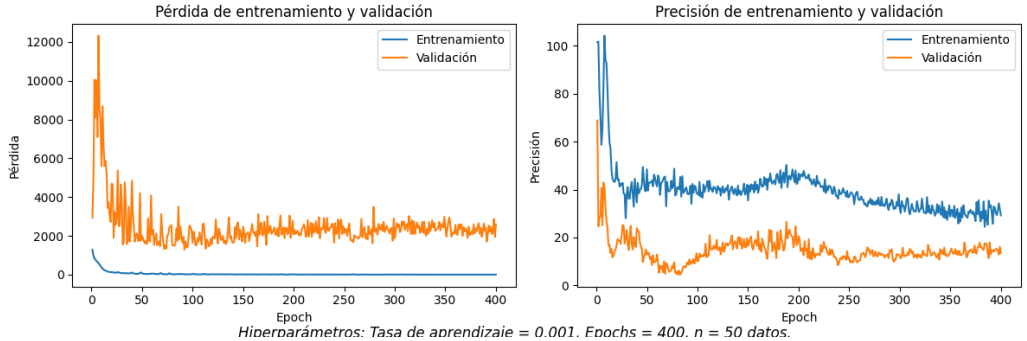

Figura 53: Curvas de aprendizaje y precisión de red convolucional temporal

8.4. Red convolucional temporal con mecanismo de atención

Partiendo de los resultados del modelo de red convolucional temporal discutido en la sección anterior, el objetivo de la arquitectura propuesta en este caso presentó una extensión al agregar un mecanismo de atención simple. Con la adición de los mecanismos de atención conectados a los bloques temporales del modelo, la arquitectura mostró capacidad para ajustarse a la respuesta deseada tras una última capa densa con las dimensiones requeridas en los canales del sistema.

En este contexto, el mecanismo de atención se agregó con el objetivo de permitir a la red generar relaciones temporales con la noción de una ponderación a características aprendidas previamente, tal como se realizó para la implementación de red residual. Así, considerando la simplicidad agregada de los bloques temporales a la red, se contempló la posibilidad de reducir la cantidad de recursos y tiempo de entrenamiento de la red al mantener cierto desempeño en su capacidad para extraer las características de los conjuntos de datos para reconstruir la trayectoria de los agentes.

Cabe mencionar que, al igual que con el modelo de red temporal convolucional en la sección anterior, este modelo también asume una sola respuesta de predicción basándose en la información completa incluida en los tensores de entrada.

8.4.1. Resultados

Primeramente, los resultados de implementación de este modelo presentaron un comportamiento característico agregado a la función para evaluar distintas profundidades en la red. En este escenario, el mejor caso evaluado cuya respuesta se muestra en la Figura 54, consideró un modelo con 10 capas convolucionales con 128 nodos conectados empleados para la extracción de características.

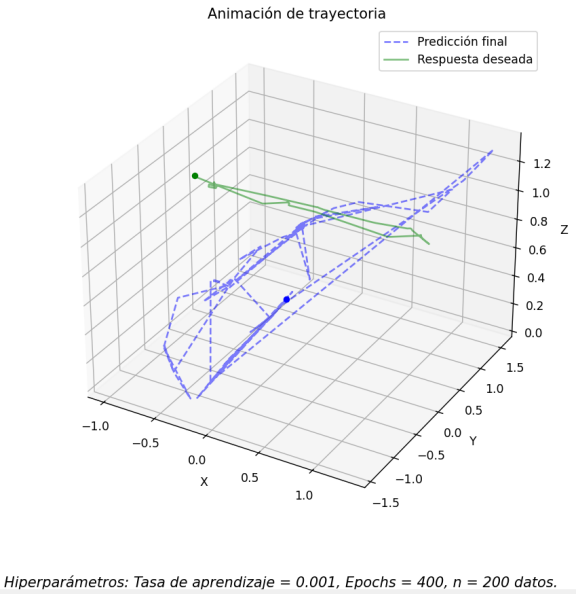

Figura 54: Estimación de pose con red convolucional temporal con mecanismo de atención

Dado la respuesta de la red en este escenario, se obtuvo un promedio de 0.49m y 8.85° de error absoluto medio como referencia respecto al desempeño de los modelos anteriores. Esto, a su vez, sugirió una respuesta aparentemente más acertada en comparación con el modelo de red convolucional temporal sin la adición de un bloque de atención simple sobre el conjunto de datos de movimientos rectos. De esa manera, se identificó la capacidad del modelo para extraer las características de los canales y reconstruir las señales deseadas, tal como se ilustra en la Figura 55. Sin embargo, se sostiene que este comportamiento no permite concluir que el modelo es apto para realizar las predicciones en tiempo real del sistema planteado. Al contrario, se observó que el modelo mantiene un mayor rendimiento en conjuntos de datos más compactos a diferencia del modelo de red convolucional temporal sin mecanismo de atención tal como se ilustra en la Figura 89.

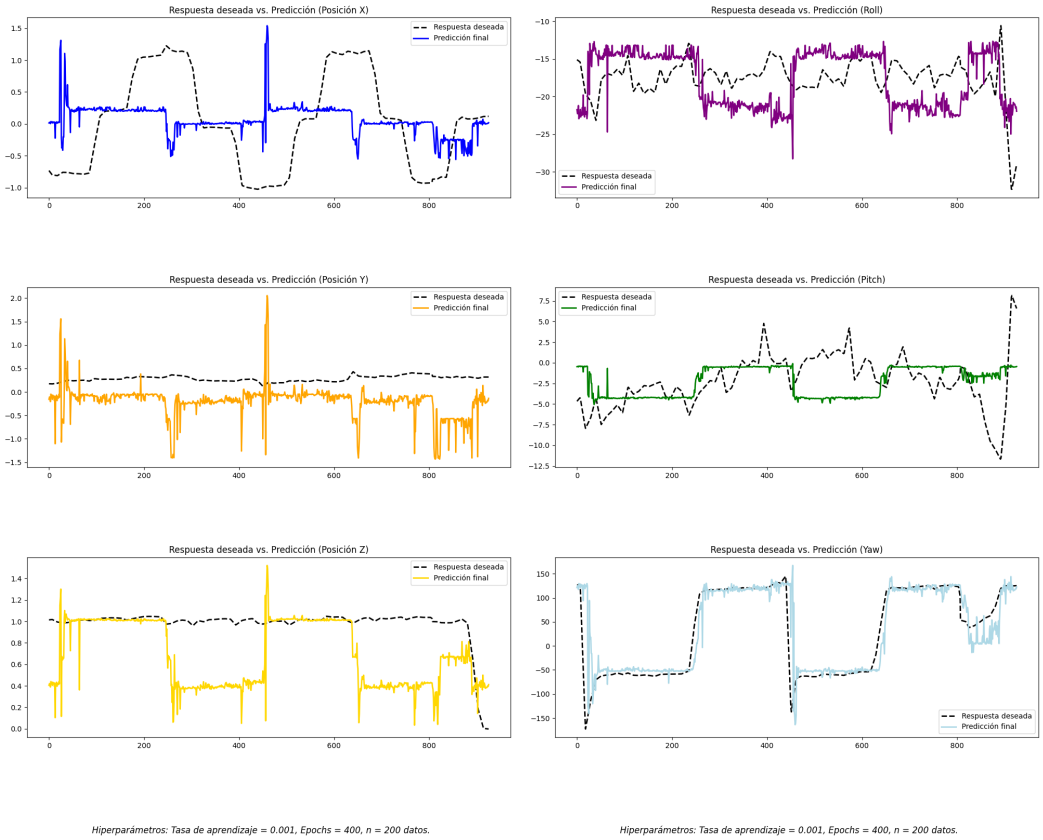

Figura 55: Estimación de pose con red convolucional temporal con mecanismo de atención

A manera de concluir, se observaron las tendencias en las curvas de aprendizaje y precisión en la Figura 56, en donde se identificó nuevamente un comportamiento de pérdida que fluctúa a lo largo del entrenamiento. Sin embargo, dado el caso de la precisión del modelo, se consideró que este fue capaz de estabilizarse luego de 120 épocas.

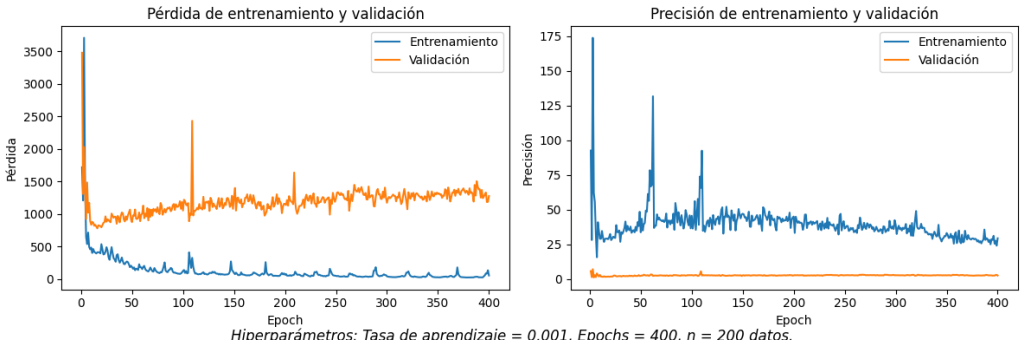

Figura 56: Curvas de aprendizaje y precisión de red convolucional temporal con mecanismo de atención

CAPÍTULO 9

Filtros adaptables

En este capítulo se presenta el análisis y la implementación de filtros adaptables empleados para estimar la pose y trayectoria de los agentes a partir de características de su movimiento. El enfoque se presenta en los datos generados por las herramientas descritas en el Capítulo 8, en donde se describen las lecturas de las unidades de medición inercial como canales de entrada y las lecturas del sistema de captura de movimiento óptico como señales deseadas. El objetivo de esta etapa fue identificar una relación de coeficientes óptimos, utilizando particularmente filtros de mínimos cuadrados medios (LMS) y variantes del mismo. Posteriormente, se realizó el análisis sobre adición de herramientas para evaluar su capacidad para extraer características no lineales dentro de los conjuntos de datos.

La propuesta de los filtros adaptables se realizó con el fin de explorar las ventajas que conllevan los algoritmos para ajustar los coeficientes de forma iterativa y operar en tiempo real. Este capítulo se enfoca en la implementación de los algoritmos LMS y *kernel* LMS para analizar sistemas no lineales y verificar características de su comportamiento. Las secciones a continuación detallan el desarrollo de los algoritmos y el proceso de análisis para la reconstrucción del movimiento en donde se buscó encontrar un comportamiento característico tanto en los coeficientes como en elementos de importancia de manera que se puedan realizar predicciones o estimar el sistema estocástico.

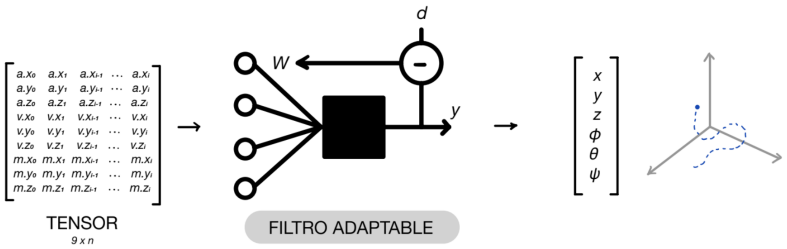

Figura 57: Diagrama de funcionamiento de filtros adaptables

Así, cabe mencionar que el contenido del capítulo surgió principalmente como alternativa, dentro del contexto de aprendizaje automático, para explorar las ventajas que proponen particularmente los algoritmos de filtros de *kernel* adaptables. Esto, a partir del entendimiento de su habilidad de identificar las características en los sistemas mediante un proceso iterativo de ajuste de sus elementos y la proyección del valor agregado a los datos según el tipo de *kernel* seleccionado [29]. Así, en el contexto de este trabajo, se refiere al comportamiento de estimación del sistema como la obtención de las relaciones entre los elementos y características de los canales de entrada que, en el caso ideal, demostrarían un método para la captura de movimiento empleando mediciones exclusivamente generadas por las unidades de medición inercial.

9.1. Filtro LMS

El filtro LMS se utilizó inicialmente para comprender cómo implementar algoritmos capaces de procesar múltiples canales de entrada y salida. Por lo tanto, la implementación del algoritmo se realizó mediante la generalización de las dimensiones en los elementos para un algoritmo LMS estándar que asume un modelo lineal. Esto se realizó con el fin de evaluar la habilidad del algoritmo para ajustar un tensor de coeficientes capaz de establecer las relaciones entre los canales de entrada. A su vez, se contempló como un método capaz de adaptarse y asegurar convergencia dada la tarea de aproximarse a la pose y trayectoria de los agentes.

Así, el objetivo fue estudiar como la exposición de un sistema no lineal y un filtro principalmente lineal afecta la convergencia de los coeficientes a medida que se itera sobre los conjuntos de datos. Tal como se presenta en la Figura 58, la idea de implementación del algoritmo asume una estructura en la que se dosifican las señales, se ajustan los coeficientes y se produce un resultado estimado de las señales deseadas respecto al error. Por ello, esta sección, tiene como objetivo explorar los resultados obtenidos al aplicar el filtro LMS en casos simulados sobre los conjuntos de datos descritos en el capítulo 8 para explorar su capacidad de convergencia. Cabe notar que desde un inicio se esperó verificar que, en efecto, los coeficientes o pesos agregados no presentarían convergencia dado la naturaleza no lineal estocástica del conjunto de datos que se aborda en la sección del filtro *kernel* LMS.

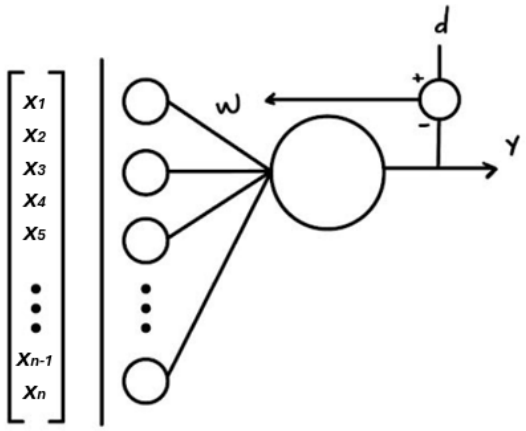

Figura 58: Diagrama de funcionamiento unidad de filtros LMS

9.1.1. Metodología

Dado el contexto de los filtros LMS, el análisis se realizó particularmente utilizando herramientas en el entorno de MATLAB para estructurar los resultados. Reiterando, el algoritmo se consideró con un enfoque en el proceso iterativo para ajustar los coeficientes W con el fin de minimizar el error entre las señales de las unidades de medición inercial y las señales de posición y orientación obtenidas del sistema de captura óptico para el movimiento de los agentes.

En este caso, el proceso se describe mediante los siguientes parámetros:

- n: cantidad de coeficientes
- i: paso de tiempo
- k: canal de entrada de unidad de medición inercial.
- j: canal de salida deseado.

Dado el contexto del algoritmo, primeramente se definió el vector de entrada ($\mathbf{x}_{\text{vec}}$) con los datos ordenados de la serie, cuyas dimensiones están descritas por la cantidad de coeficientes. Así, se puede considerar la cantidad de coeficientes como una “ventana de tiempo"móvil, permitiendo una interpretación similar al método de manipulación de los tensores cubierto en los modelos de aprendizaje profundo, como se muestra en la Figura 38. Esta manipulación de los datos permitió contemplar un comportamiento causal en donde el valor en el tiempo actual corresponde al último valor de la ventana $x(k, i-n+1)$ para $i = n : l$ en donde l es la cantidad de muestras en el conjunto de datos y se puede expresar de la siguiente manera:

$$\mathbf{x}_{\text{vec}} = \begin{bmatrix} x(k,i) \\ x(k,i-1) \\ \vdots \\ x(k,i-n+1) \end{bmatrix}.$$

Mediante este comportamiento, en cada iteración del algoritmo se realizan las actualizaciones de los coeficientes W y se generan las estimaciones Y. Cabe notar que esto se realiza con la asistencia del error calculado E y las señales deseadas d como el método característico del filtro.

Así, el algoritmo asumió las siguientes operaciones para cada iteración:

$$Y(j,i) = W(:,k,j)^{\top}\mathbf{x}_{\text{vec}},$$

$$E(j,i) = d(j,i) - Y(j,i),$$

$$W(:,k,j) = W(:,k,j) + 2\mu E(j,i)\mathbf{x}_{\text{vec}}.$$

Posteriormente, el análisis constó del ajuste de los parámetros μ (tasa de aprendizaje) y n (cantidad de coeficientes), lo que permitió observar cómo estos afectaron el desempeño para estimación de pose y trayectoria. Así, se obtuvieron ciertas configuraciones del algoritmo que resultaron en la convergencia de los movimientos realizados sin mayor error, tal como se discute en la sección de resultados de este capítulo. No obstante, cabe remarcar que este algoritmo aún contó con la asistencia de la señal de salida real para ajustar los coeficientes, de manera que no fue posible definir los resultados como predicción basada únicamente en los valores recuperados por unidades de medición inercial. De la misma manera, se debe resaltar nuevamente que el objetivo de este experimento fue evidenciar que los coeficientes no presentarían convergencia dado el supuesto de linealidad en el modelo implementado y no linealidad del sistema expuesto.

9.1.2. Resultados

Los resultados del filtro LMS demostraron que, bajo ciertas condiciones, el algoritmo es capaz de converger con precisión hacia las señales de salida deseadas. En pruebas realizadas sobre el conjunto de datos con forma de cubo representado en la Figura 74, se observó que la cantidad de coeficientes influye directamente en la velocidad de convergencia y la precisión de la estimación, como se ilustra en la Figura 59. Dados los parámetros, fue posible discutir sobre las características en el comportamiento de la convergencia mediante factores como tiempo de respuesta, precisión y consistencia.

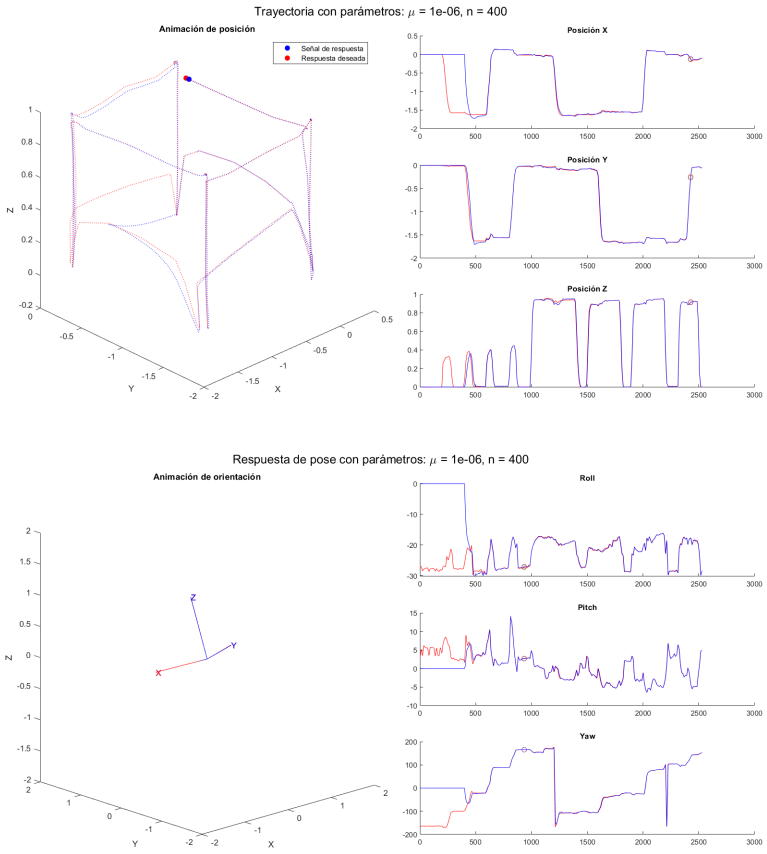

Figura 59: Resultados de estimación de pose y trayectoria de conjunto de datos con forma de cubo empleando filtro LMS

Se observó que los parámetros mantuvieron una relación entre sí que afectó directamente al desempeño del algoritmo en ciertas configuraciones y para distintos conjuntos de datos según los factores mencionados. En esta instancia, a medida que se incrementó la cantidad de coeficientes, se consideró necesario reducir la tasa de aprendizaje asignada para lograr resultados satisfactorios y estabilidad del modelo. En consecuencia, a medida que se aumentó la cantidad de coeficientes, se produjo un retraso en tiempo de respuesta inicial y se adquirió mayor consistencia en la convergencia de las señales, tal como se muestra en las Figuras 60 y 61 para la respuesta. Adicionalmente, se consideran los resultados en la Figura 62 como referencia en cuanto a la diferencia en cantidad de error para los casos de mayor y menor precisión.

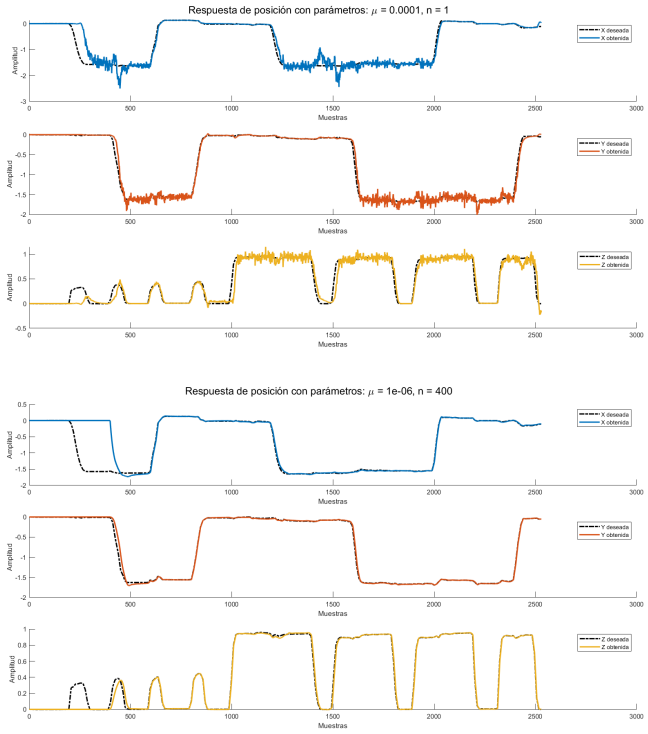

Figura 60: Respuesta a posición empleando filtro LMS con múltiples cantidades de coeficientes

Figura 61: Respuesta a orientación empleando filtro LMS con múltiples cantidades de coeficientes

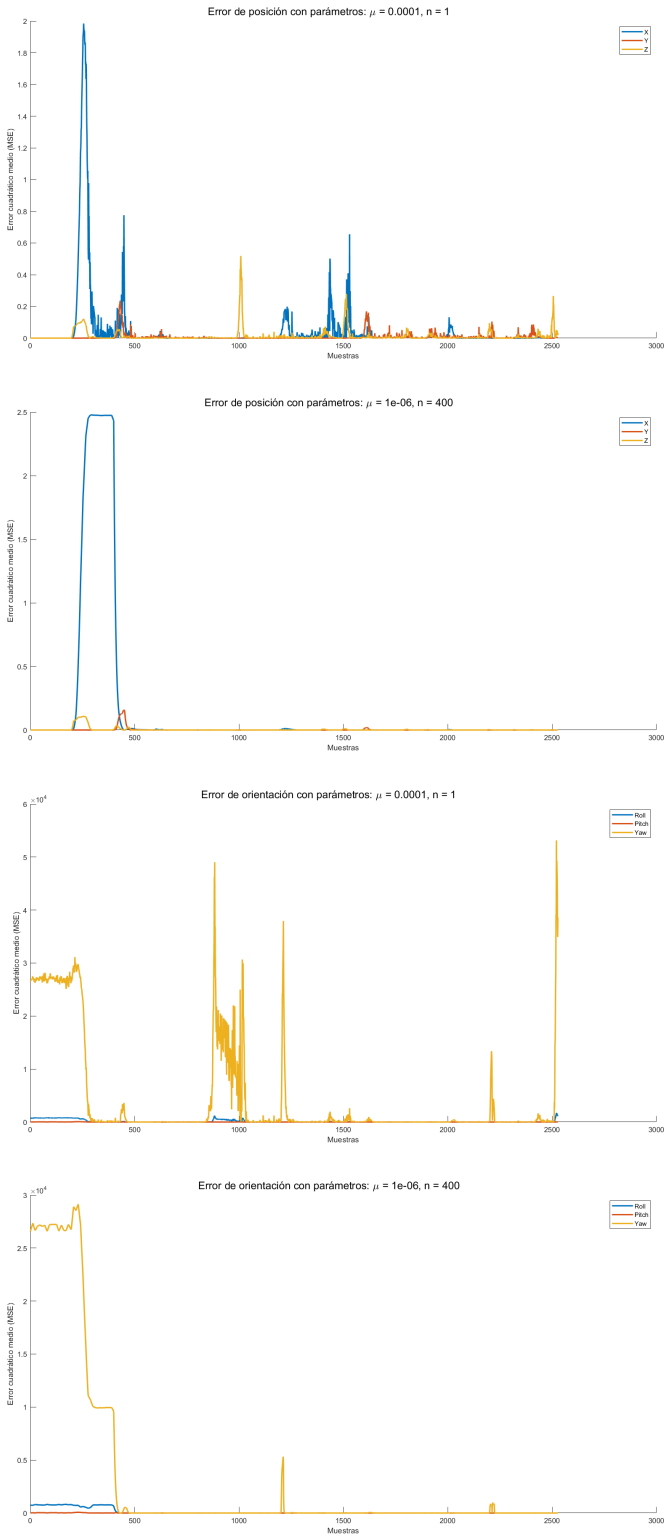

Figura 62: Error en respuestas con mayor y menor convergencia a posición y orientación empleando filtro LMS

De la misma manera, dado el objetivo principal de la implementación de los filtros LMS y el supuesto inicial sobre el algoritmo, se comprobó que no existió convergencia de los coeficientes a un valor óptimo constante o relación que pudiera permitir realizar predicción del modelo propuesto, tal como se muestra en la Figura 63. Así, cabe remarcar que este comportamiento se asume de la misma manera en el caso de todos los conjuntos de datos realizados en el Capítulo 8.

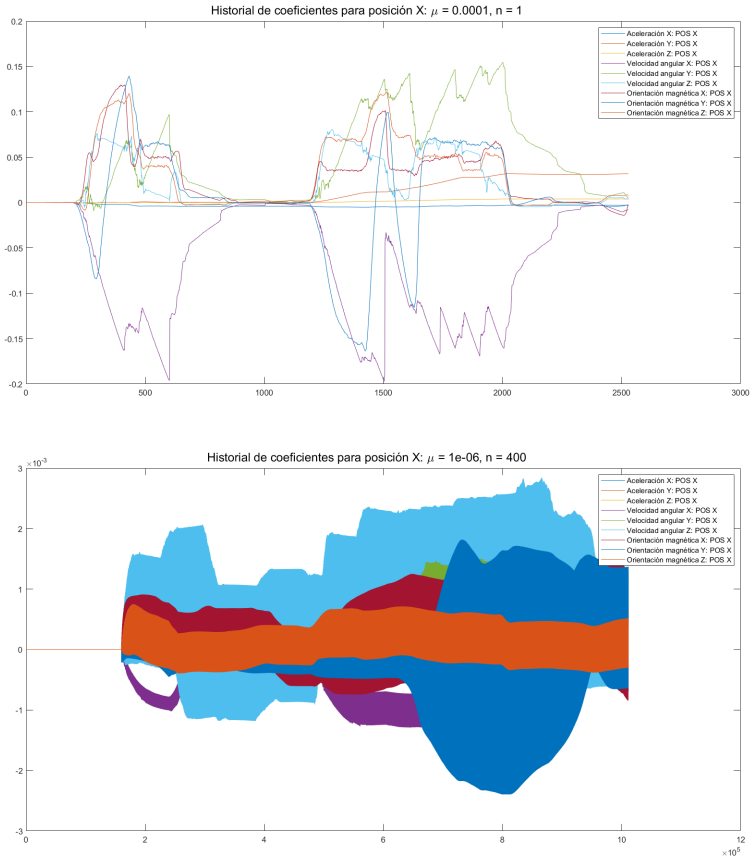

Figura 63: Divergencia en el tiempo de coeficientes en respuestas a coordenadas en x empleando filtro LMS

En este caso, en la Figura 63, se muestra la respuesta en los coeficientes para la reconstrucción de las coordenadas sobre el eje X y se supone el mismo comportamiento sobre el resto las variables consideradas para la posición y orientación.

Cabe mencionar que, a diferencia del caso real, un caso simulado en donde las señales fueron controladas demuestra que existe una convergencia siempre que el sistema se estabilice o se encuentre constante tras cierto tiempo. Tal como se muestra en la Figura 64 y la Figura 65, en donde se realizaron pruebas sobre sistemas de armónicos amortiguados y sinusoides, la estabilidad en la señal deseada permite una perspectiva de comparación en cuanto a la convergencia de los coeficientes particulares de cada caso.

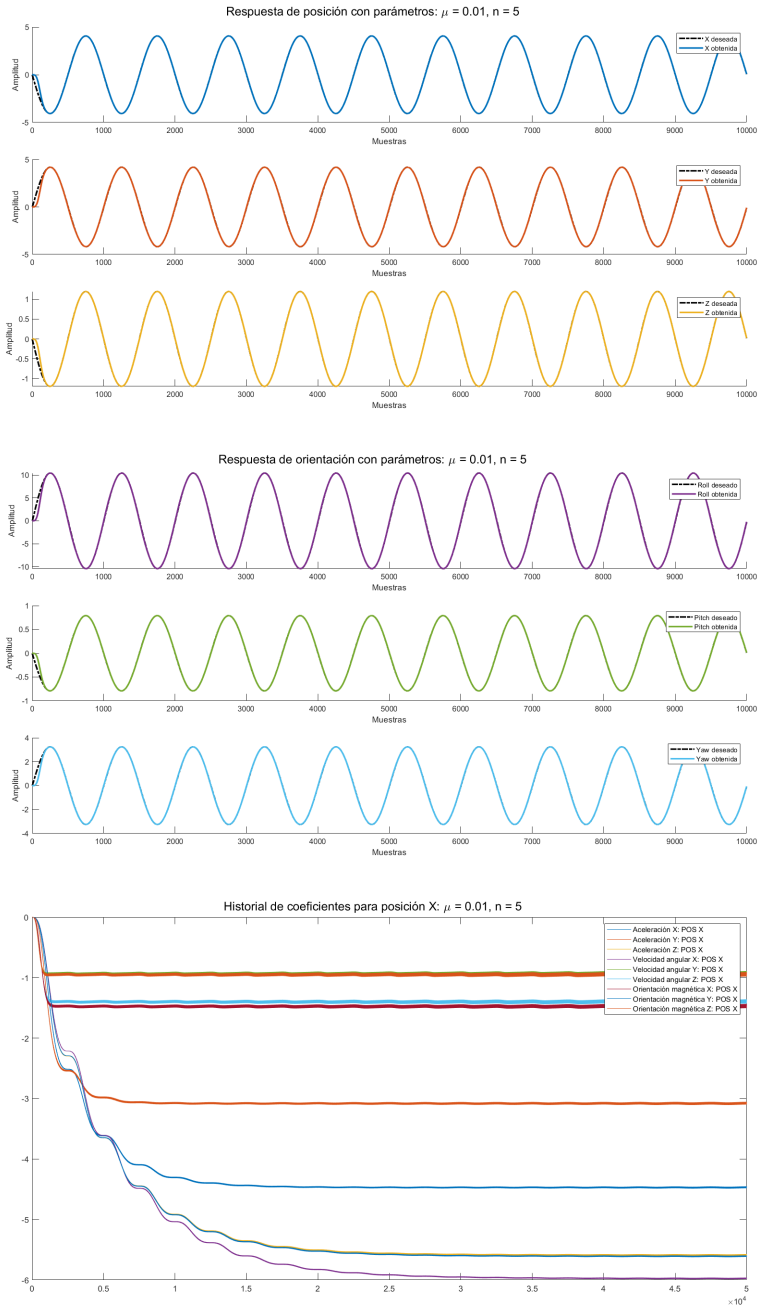

Figura 64: Resultados de simulación con señales sinusoidales en filtro LMS

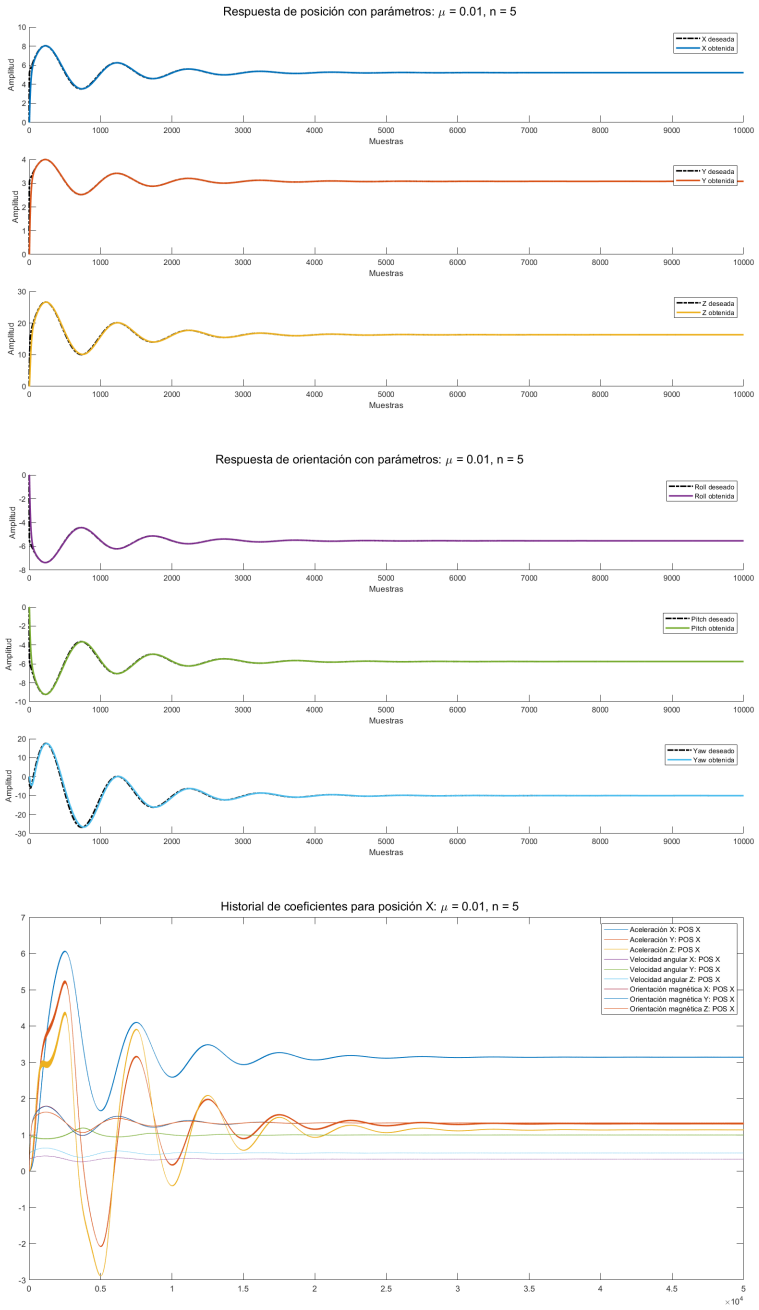

Figura 65: Resultados de simulación con señales de armónicos amortiguados en filtro LMS

A manera de concluir, estos resultados sugirieron que los escenarios de datos ruidosos promovieron la obtención de un comportamiento divergente. Este conocimiento es relevante ante el contexto del comportamiento típico obtenido en las lecturas generadas por unidades de medición inercial que afectarían principalmente la capacidad de convergencia de los coeficientes asignados a los canales de entrada del filtro.

9.2. Filtro de *kernel* LMS

El filtro de *kernel* LMS se introdujo para explorar la expansión de características no lineales y estudiar su capacidad para generar modelos de predicción en tiempo real. Este enfoque, se realizó con el fin de "mapear" los datos de entrada a un espacio de mayor dimensión en donde se esperó encontrar mayor detalle de relaciones complejas entre las señales de entrada características de un modelo no lineal.

Por lo tanto, en este estudio, se utilizó principalmente el *kernel* gaussiano dado su capacidad de transformar los datos en los canales de entrada a un espacio de características de mayor dimensión de forma universal [29]. Esto se consideró de manera que se pudiera tratar con las características del filtro LMS estándar que asume linealidad para encontrar patrones en los datos de un modelo no lineal. Así, la estructura del algoritmo propuesto para el análisis, asumió el comportamiento principal del filtro LMS en la sección anterior de este capítulo con la adición de la expansión en dimensiones de características en el vector de entrada.

Cabe mencionar que el análisis en esta sección cubre únicamente el esfuerzo realizado por estudiar la extracción de relaciones generadas por un modelo de filtro adaptable para realizar predicción. En esta instancia, no fue posible asumir que la convergencia de los coeficientes a un valor de estado estacionario permitiría la habilidad de generar predicciones del sistema dada la proyección particular de los canales de entrada generados por la unidad de medición inercial al espacio de mayor dimensión. Sin embargo, se hace referencia a los coeficientes con el objetivo de permitir una comparativa con el historial adquirido en los filtros LMS estándar.

9.2.1. Metodología

En la experimentación con filtro *kernel* LMS, se utilizaron las herramientas necesarias para analizar el desempeño del algoritmo con un enfoque en la capacidad para realizar predicciones de la posición y orientación usando los conjuntos de datos disponibles. Se consideró como primer paso incorporar el método de *kernel* al algoritmo del filtro LMS discutido en la sección anterior, de manera que se pudiera generar una referencia del efecto contra el caso propuesto como parte de la metodología. Esto con el objetivo de estudiar cómo la introducción del elemento afecta la convergencia hacia la señal deseada y discutir posibles técnicas para recuperar información relevante para descifrar el sistema.

De esa manera, además de la estructura básica del filtro LMS, se añadió particularmente el *kernel* gaussiano (función de base radial) (13) para procesar los canales de entrada generados por las unidades de medición inercial considerando los valores previos en el vector de entrada. En este caso, la introducción de esta función permitió incorporar un nuevo parámetro ajustable: el tamaño del *kernel* denotado por σ. A partir de esta adición, se experimentó con el ajuste de este parámetro para evaluar el desempeño del algoritmo.

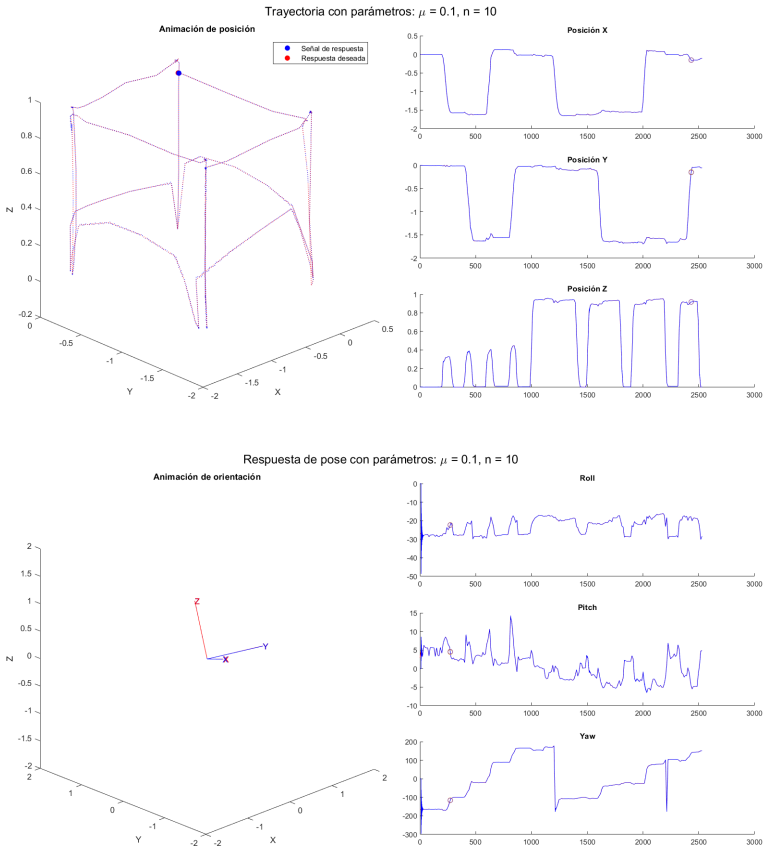

Figura 66: Resultados de estimación de pose y trayectoria de conjunto de datos con forma de cubo empleando filtro KLMS

Tal como se puede observar en la Figura 66, se pudo notar que los valores de tasa de aprendizaje y la cantidad de coeficientes necesarios fueron menores en comparación a con los resultados en el algoritmo LMS estándar. Esto, a su vez, sugiere un mejor rendimiento para los escenarios en los conjuntos de datos como en la Figura 67 y la Figura 68.

$$\phi(X_1, X_2) = \exp\left(-\frac{\|X_1 - X_2\|^2}{2\sigma^2}\right).$$

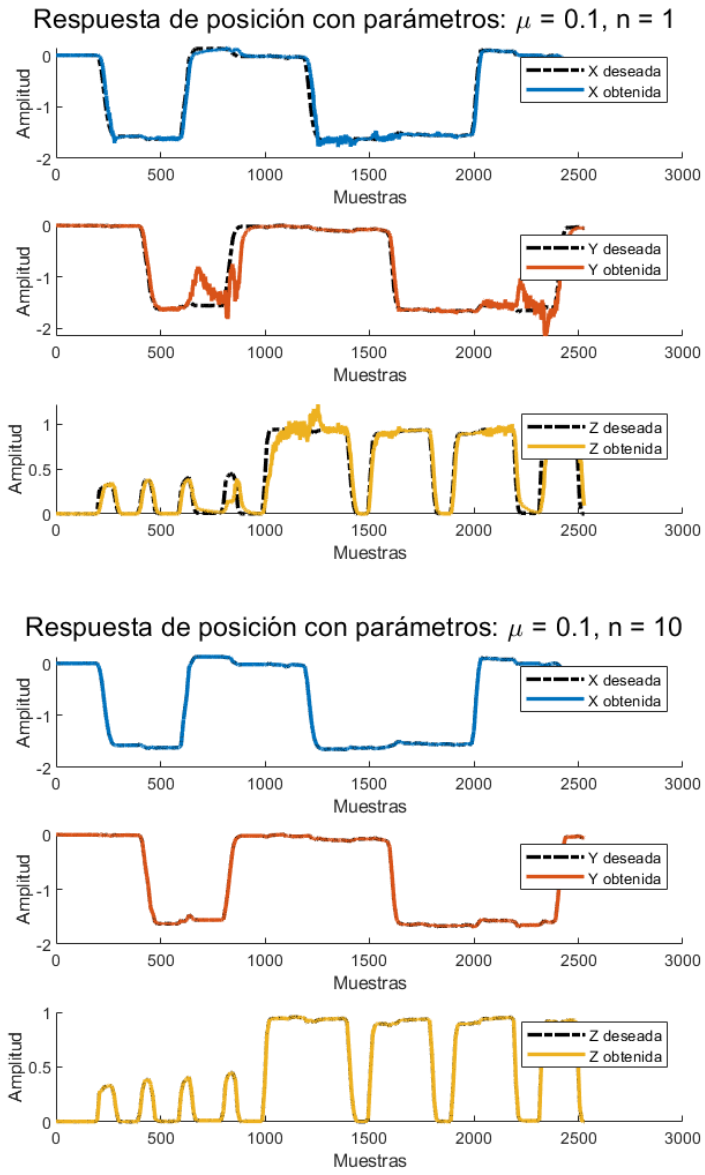

Figura 67: Respuesta a posición empleando filtro KLMS con múltiples cantidades de coeficientes

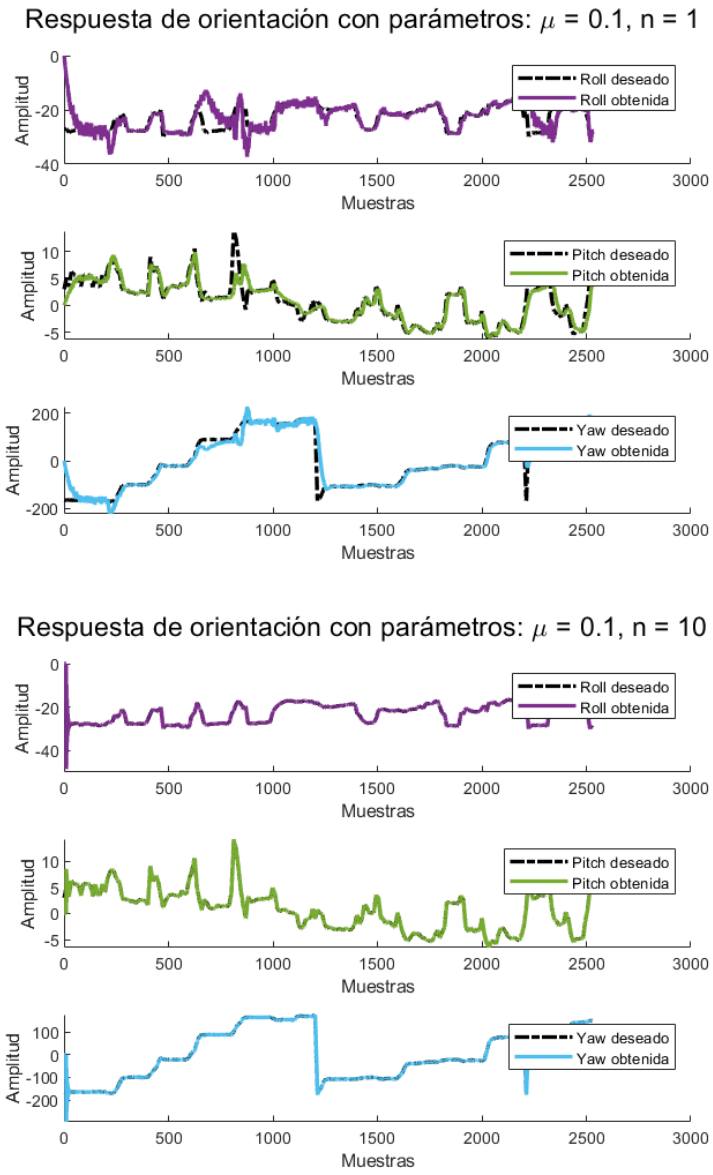

Figura 68: Respuesta a orientación empleando filtro KLMS con múltiples cantidades de coeficientes

Posteriormente, se realizó una observación del comportamiento de los coeficientes asignados a lo largo del conjunto de datos para generar una comparación respecto al filtro anterior. A su vez, se consideró el comportamiento bajo la noción de que la proyección se realizó mediante el valor actual y los valores previos del vector de entrada para generar la expansión de características a un espacio de infinitas dimensiones según el *kernel* gaussiano. Dado el contexto, se pudieron observar los patrones generados a lo largo de las señales en el mismo conjunto de datos evaluado para filtro LMS tal como se pueden apreciar en la Figura 69, a diferencia de la Figura 59.

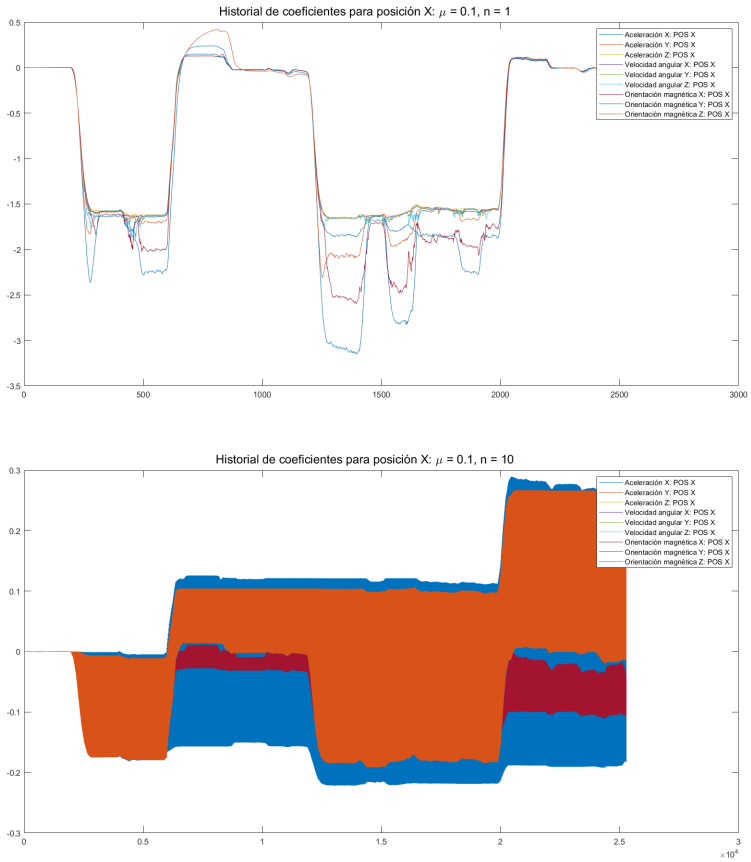

Figura 69: Comportamiento de coeficientes obtenidos con filtro KLMS

Posteriormente, para los resultados de esta sección se discutieron los comportamientos observados particularmente con el objetivo de generar una base para evaluar el potencial de filtros adaptables para la estimación del sistema óptimo para captura de movimiento inercial.

9.2.2. Resultados

En esta sección, se discute la adición del tipo de *kernel* empleado para estimar los movimientos a partir de la respuesta del algoritmo. Dado la comparativa realizada con los resultados del filtro LMS estándar, se aprovecharon las ventajas de aplicar el método de *kernel* con el fin de mejorar la precisión y tiempo de respuesta. Así, el algoritmo de filtro *kernel* LMS permitió un uso más eficiente de los recursos empleados al reducir las dimensiones de coeficientes y vectores de entrada requeridos para obtener una convergencia con mínimo error a las señales del movimiento.

De esa manera, el punto de partida con el *kernel* gaussiano en esta implementación, demostró tener una aproximación con resultados deseados respecto al desempeño en varios conjuntos de datos, como se presenta en la Figura 70 y la Figura 71.

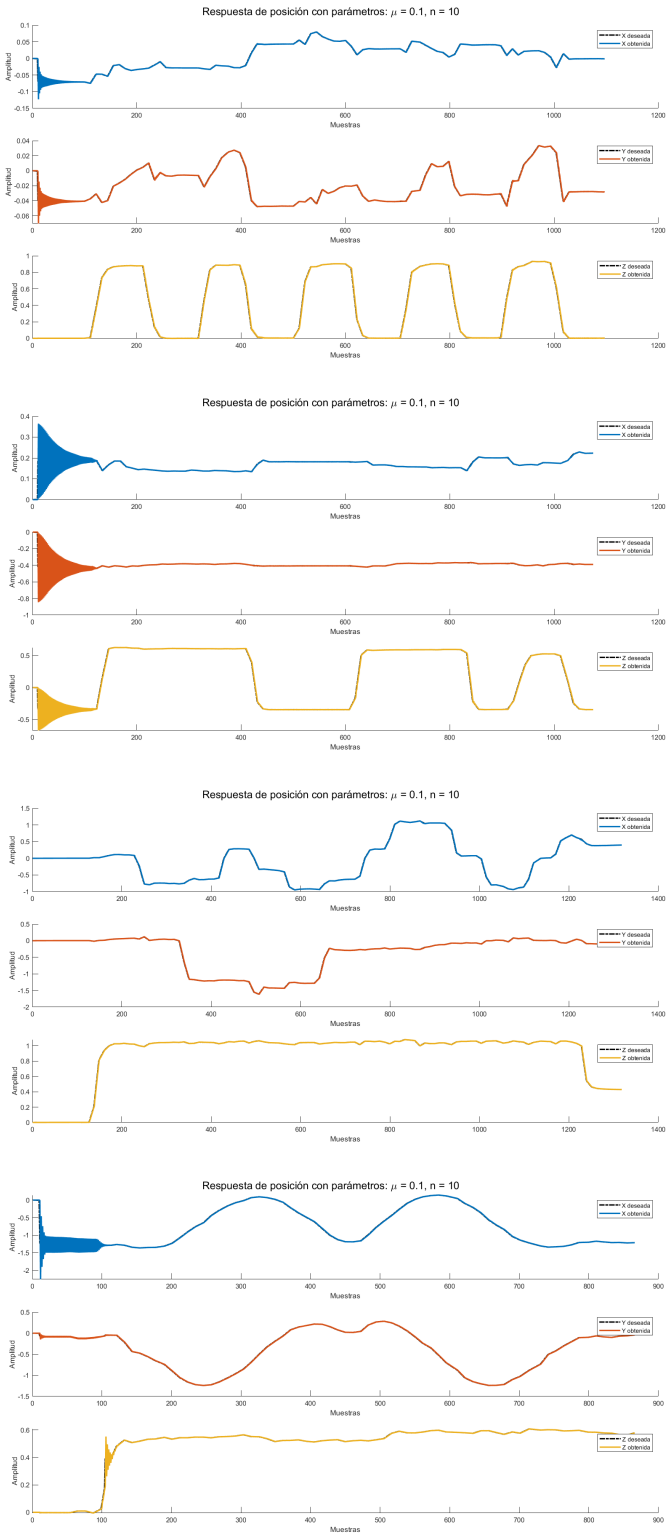

Figura 70: Estimación de posición deseada en múltiples conjuntos de datos empleando filtro KLMS

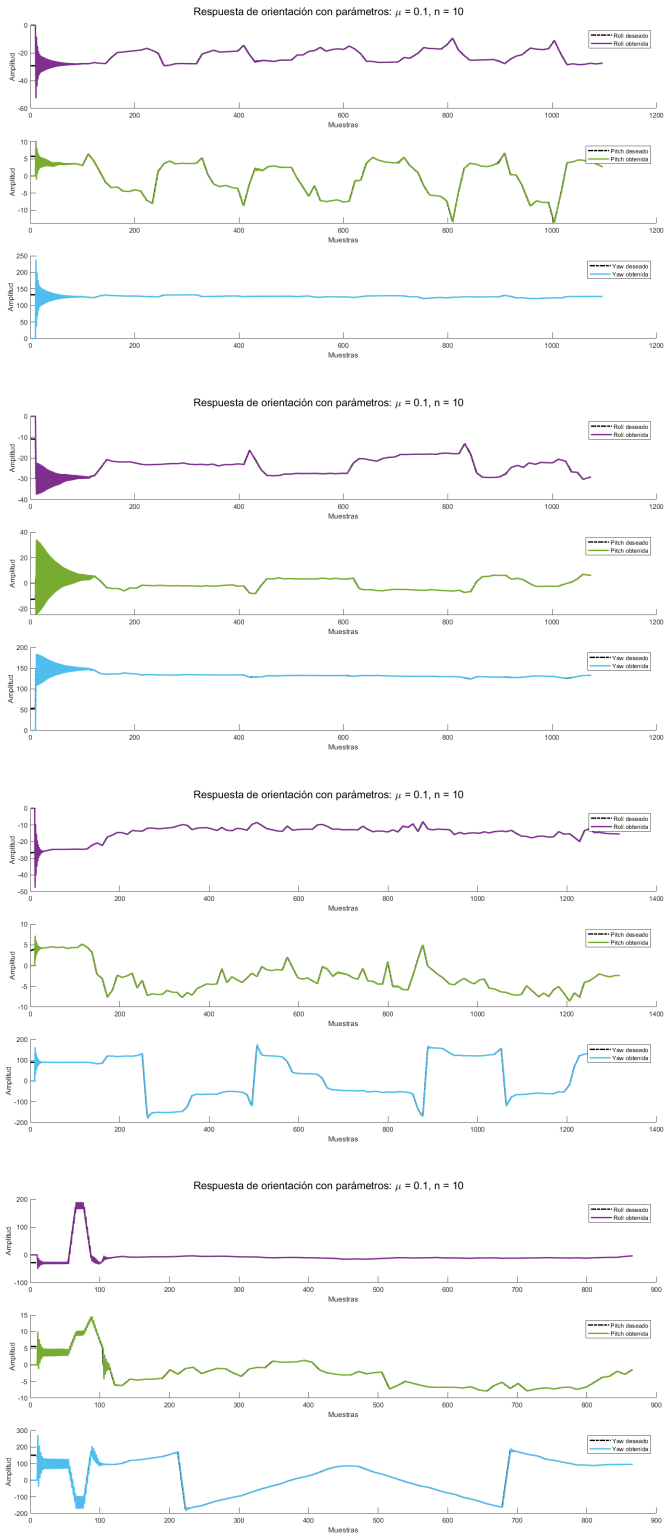

Figura 71: Estimación de orientación deseada en múltiples conjuntos de datos empleando filtro KLMS

A partir de estas observaciones, se discutió sobre la capacidad para concluir que el uso de

distintos tipos de *kernels* con dimensiones conocidas, como el *kernel* polinomial, se podrían emplear para generar una aproximación al sistema generado a partir de los conjuntos de datos [29]. Por ello, se llegó a la conclusión de qué tipos de *kernel* dinámicos comprenden un potencial para generar resultados deseados y estimar las relaciones óptimas requeridas con el fin de realizar predicciones, al considerar su capacidad para ajustarse al comportamiento de los datos en cada iteración. Así, propuestas como los sistemas de identificación dispersa de sistemas dinámicos no lineales (*Sparse Identification of Nonlinear Dynamical Systems* - SINDy [46]) o métodos de optimización por desambiguación lineal y no lineal (*Linear and Nonlinear Disambiguation Optimization* - LANDO [47]), presentan potencial con un enfoque a la estimación de los sistemas tal como se espera lograr para continuar el estudio de estos algoritmos en los escenarios propuestos en este trabajo.

CAPÍTULO 10

Conclusiones

- Se lograron diseñar e integrar dos dispositivos capaces de obtener la información esencial de unidades de medición inercial y el sistema de captura de movimiento Optitrack, con un enfoque dirigido a la captura de movimiento inercial asistido por herramientas de aprendizaje profundo. En particular, el ensamblaje de los dispositivos incluyó una unidad de medición inercial, un microcontrolador ESP32 — Tiny S3, un marcador personalizado para interactuar con el sistema de captura óptico, y un método de sujeción que mantiene el factor de forma, permitiendo su uso como guantes.
- Se logró establecer una serie de programas encargados de realizar la comunicación entre dispositivos y el servidor del ecosistema Robotat para generar conjuntos de datos ordenados para uso en los algoritmos de aprendizaje. En particular, se implementó una estructura multitarea en el programa de los dispositivos de captura para obtención y envío simultáneo de datos. En consecuencia, se lograron generar conjuntos de datos con una tasa estimada de 1 dato del sistema de captura óptico por cada 12 de la unidad de medición inercial.
- Los filtros adaptables LMS y *kernel* LMS mostraron ser herramientas útiles con potencial para continuar el estudio sobre la estimación de pose y trayectoria de los agentes en tiempo real. Ambos algoritmos presentaron convergencia respecto a las señales deseadas de posición y orientación al exponer las series de datos generados por las unidades de medición inercial.
- Se comprobó que las arquitecturas de red propuestas son capaces de extraer las características de los conjuntos de datos dadas las configuraciones adecuadas. Sin embargo, no fue posible demostrar su capacidad de estimar con precisión la trayectoria o converger adecuadamente al comportamiento deseado, dado un error absoluto medio de 0.49m y 8.85° para la posición y orientación en el mejor caso observado entre los modelos de aprendizaje profundo propuestos.

CAPÍTULO 11

Recomendaciones

- Evaluar el protocolo de comunicación e integrar la capacidad para establecer conexión entre los dispositivos de captura de movimiento inercial para ambas manos en simultáneo.
- Evaluar métodos para mejorar el periodo de obtención de muestras del sistema de captura de movimiento Optitrack con el fin de obtener más resolución de los datos reales de trayectoria y pose de los agentes.
- Evaluar la adición de cámaras para evitar conflictos de oclusión con los marcadores del sistema de captura óptico.
- Ampliar en la estandarización para entrenamiento de los modelos empleados, asegurando un flujo desde la generación de los conjuntos de datos hasta la obtención de resultados en los algoritmos.
- Analizar métodos óptimos para la ejecución de los modelos dentro del programa del microcontrolador seleccionado para generar un sistema de captura de movimiento inercial asistido por herramientas de aprendizaje profundo en tiempo real e independiente.
- Ampliar la investigación en redes temporales convolucionales asegurando que las herramientas empleadas en los modelos demuestren un comportamiento claro de la extracción de caracteristicas a partir de las lecturas de las unidades de medición inercial.
- Si se desea continuar con el estudio de filtros adaptables, considerar el uso de métodos de *kernel* dinámicos con el fin de explorar su capacidad para realizar predicciones en tiempo real. Las técnicas como LANDO [47] o SINDy [46] comprenden potencial para la aplicación.

CAPÍTULO 12

Bibliografía

[1] N. Cohen e I. Klein, "Inertial Navigation Meets Deep Learning: A Survey of Current Trends and Future Directions," *ArXiv*, vol. abs/2307.00014, 2023. dirección: https://api.semanticscholar.org/CorpusID:259316889.

[2] H. M. I. X. Yun E. Bachmann y J. Calusdian, "Self-contained Position Tracking of Human Movement Using Small Inertial/Magnetic Sensor Modules," en *Proceedings 2007 IEEE International Conference on Robotics and Automation*, IEEE, 2007, págs. 2526-2533.

[3] Q. S. H. Yan e Y. Furukawa, "RIDI: Robust IMU Double Integration," en *Computer Vision – ECCV 2018*, European Conference on Computer Vision, 2018, págs. 641-656.

[4] S. A. Herath, "Robust Neural Inertial Navigation in the Wild," Master of Science (Computing Science), SIMON FRASER UNIVERSITY, 2019.

[5] A. B. M. Brossard y S. Bonnabel, "AI-IMU Dead-Reckoning," *IEEE Transactions on Intelligent Vehicles*, vol. 5, n.º 4, págs. 585-595, 2020.

[6] S. U. Kim, J. Lee, J. Yoon, S.-K. Ko y J. Kim, "Robust methods for estimating the orientation and position of IMU and MARG sensors," *Electronics Letters*, vol. 57, n.º 21, págs. 816-818, 2021. DOI: https://doi.org/10.1049/ell2.12263. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/ell2.12263. dirección: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/ell2.12263.

[7] C. P. Montoya, "Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica.," Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[8] C. C. Cossette, M. Shalaby, D. Saussié, J. R. Forbes y J. Le Ny, "Relative Position Estimation Between Two UWB Devices With IMUs," *IEEE Robotics and Automation Letters*, vol. 6, n.º 3, págs. 4313-4320, 2021. DOI: 10.1109/LRA.2021.3067640.

[9] M. Menolotto, D.-S. Komaris, S. Tedesco, B. O'flynn y M. J. Walsh, "Motion Capture Technology in Industrial Applications: A Systematic Review," *Sensors (Basel, Switzerland)*, vol. 20, 2020.

[10] J. Dower y P. Langdale, *Performing for Motion Capture*. Bloomsbury Publishing, 2022.

[11] M. Kitagawa y B. Windsor, *MoCap for Artists: Workflow and Techniques for Motion Capture*. CRC Press, 2020.

[12] OptiTrack. "OptiTrack - Motion Capture Systems." (2024), dirección: `https://www.optitrack.com/`.

[13] OptiTrack. "OptiTrack Primex 41 - Motion Capture Camera." (2024), dirección: `https://optitrack.com/cameras/primex-41/specs.html`.

[14] M. Kok, J. D. Hol y T. B. Schön, "Using Inertial Sensors for Position and Orientation Estimation," *ArXiv*, vol. abs/1704.06053, 2017.

[15] N. Ahmad, R. A. B. R. Ghazilla, N. M. Khairi y V. Kasi, "Reviews on Various Inertial Measurement Unit (IMU) Sensor Applications," en *IEEE Workshop on Signal Processing Systems*, 2013.

[16] M. electronics. "MPU9250 9-Axis IMU Module." (2024), dirección: `%5Curl%7Bhttps://make.net.za/product/mpu9250-9-axis-imu-module/%7D` (visitado 04-06-2024).

[17] *IGELECTRONICS: MPU-9265 (GY 9250) 9-axis 9-DOF Attitude Gyro Magnetometer Accelerator Sensor Module*, 2024. dirección: `https://www.igelectronics.com/products/a3ed52c6be/1726810000001619429`.

[18] S. Bhattacharyya, V. Snasel, A. Hassanien, S. Saha y B. Tripathy, *Deep Learning: Research and Applications* (De Gruyter Frontiers in Computational Intelligence). De Gruyter, 2020, ISBN: 9783110670905.

[19] I. Borrero y M. Arias, *DEEP LEARNING* (Alonso Barba). Editorial de la Universidad de Huelva, 2021, ISBN: 9788418628290. dirección: `https://books.google.com.gt/books?id=kzsvEAAAQBAJ`.

[20] A. Zhang, Z. C. Lipton, M. Li y A. J. Smola, *Dive into Deep Learning*. Cambridge University Press, 2023, `https://D2L.ai`.

[21] Y. Zhang, H. Wang, R. Xu, X. Yang, Y. Wang e Y. Liu, "High-Precision Seedling Detection Model Based on Multi-Activation Layer and Depth-Separable Convolution Using Images Acquired by Drones," *Drones*, vol. 6, pág. 152, jun. de 2022. DOI: `10.3390/drones6060152`.

[22] S. J. Prince, *Understanding Deep Learning*. The MIT Press, 2023. dirección: `http://udlbook.com`.

[23] P. Contributors, *torch.nn — PyTorch 2.5 documentation*, Accessed: 2024-10-30, 2024. dirección: `https://pytorch.org/docs/stable/nn.html#loss-functions`.

[24] F. van Veen, "The Neural Network Zoo," *The Asimov Institute*, 2016. dirección: `https://www.asimovinstitute.org/neural-network-zoo/`.

[25] K. He, X. Zhang, S. Ren y J. Sun, "Deep Residual Learning for Image Recognition," en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, págs. 770-778. dirección: `https://arxiv.org/abs/1512.03385`.

[26] C. Lea, M. D. Flynn, R. Vidal, A. Reiter y G. D. Hager, "Temporal Convolutional Networks for Action Segmentation and Detection," en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, págs. 6568-6577. dirección: `https://arxiv.org/abs/1611.05267`.

[27] S. Bai, J. Z. Kolter y V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv:1803.01271*, 2018.

[28] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention Is All You Need," en *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, NeurIPS, 2017. dirección: https://arxiv.org/abs/1706.03762.

[29] W. Liu, J. C. Príncipe y S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. John Wiley & Sons, 2010, ISBN: 978-0-470-44753-6.

[30] Q. Wan, Q. Wu y L. Zou, "Understanding Several Adaptive Filter Algorithms Based on the Weight-update Strategy," en *2019 IEEE International Conference on Engineering, Technology and Education (TALE)*, 2019, págs. 1-4. DOI: 10.1109/TALE48000.2019.9225868.

[31] E. Wan, "Adjoint LMS: an efficient alternative to the filtered-x LMS and multiple error LMS algorithms," en *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 3, 1996, 1842-1845 vol. 3. DOI: 10.1109/ICASSP.1996.544227.

[32] B. Farhang-Boroujeny, *Adaptive Filters: Theory and Applications*. Wiley, 2013, ISBN: 9781118591338. dirección: https://books.google.com.gt/books?id=Fmf8TumgxEYC.

[33] B. Engelhardt, *Kernel Functions*, STA561: Probabilistic Machine Learning, Princeton University, 2013. dirección: https://www.cs.princeton.edu/~bee/courses/scribe/lec_10_09_2013.pdf.

[34] Appen. "What is Sensor Fusion?" (2021), dirección: %5Curl%7Bhttps://www.appen.com/blog/what-is-sensor-fusion/%7D (visitado 04-06-2024).

[35] S. G. Tzafestas, "12 - Mobile Robot Localization and Mapping," en *Introduction to Mobile Robot Control*, S. G. Tzafestas, ed., Oxford: Elsevier, 2014, págs. 479-531, ISBN: 978-0-12-417049-0. DOI: https://doi.org/10.1016/B978-0-12-417049-0.00012-2. dirección: https://www.sciencedirect.com/science/article/pii/B9780124170490000122.

[36] S. Fan y C. Li, ""Sensor Fusion," en *Encyclopedia of Digital Agricultural Technologies*, Q. Zhang, ed. Cham: Springer International Publishing, 2023, págs. 1224-1238, ISBN: 978-3-031-24861-0. DOI: 10.1007/978-3-031-24861-0_142. dirección: https://doi.org/10.1007/978-3-031-24861-0_142.

[37] K. A. Lamkin-Kennard y M. B. Popovic, "4 - Sensors: Natural and Synthetic Sensors," en *Biomechatronics*, M. B. Popovic, ed., Academic Press, 2019, págs. 81-107, ISBN: 978-0-12-812939-5. DOI: https://doi.org/10.1016/B978-0-12-812939-5.00004-5. dirección: https://www.sciencedirect.com/science/article/pii/B9780128129395000045.

[38] B. Rinner y M. Quaritsch, "CHAPTER 22 - Embedded Middleware for Smart Camera Networks and Sensor Fusion," en *Multi-Camera Networks*, H. Aghajan y A. Cavallaro, eds., Oxford: Academic Press, 2009, págs. 511-537, ISBN: 978-0-12-374633-7. DOI: https://doi.org/10.1016/B978-0-12-374633-7.00024-0. dirección: https://www.sciencedirect.com/science/article/pii/B9780123746337000240.

[39] S. Haykin, "Kalman Filters," en *Kalman Filtering and Neural Networks*. John Wiley y Sons, Ltd, 2001, cap. 1, págs. 1-21, ISBN: 9780471221548. DOI: https://doi.org/10.1002/0471221546.ch1. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471221546.ch1. dirección: https://onlinelibrary.wiley.com/doi/abs/10.1002/0471221546.ch1.

[40] G. Welch y G. Bishop, "An Introduction to Kalman Filter," en *International Conference on Computer Graphics and Interactive Techniques*, 1995. dirección: https://api.semanticscholar.org/CorpusID:215767582.

[41] IONOS. "Introduction to Transmission Control Protocol (TCP)." (2023), dirección: %5Curl%7Bhttps://www.ionos.com/digitalguide/server/know-how/introduction-to-tcp/%7D (visitado 04-06-2024).

[42] R. M. R. Pattamsetti. "Socket programming for TCP - Distributed Computing in Java 9." (2023), dirección: %5Curl%7Bhttps://www.oreilly.com/library/view/distributed-computing-in/9781787126992/02dd04be-0dbb-4732-8bc5-1961644e8875.xhtml%7D (visitado 04-06-2024).

[43] ESP32S3. "TinyS3 Documentation." (2024), dirección: %5Curl%7Bhttps://esp32s3.com/tinys3.html%7D (visitado 04-06-2024).

[44] *EasyEDA*, https://easyeda.com/, Accessed: 2024-09-17.

[45] A. Kono, *MPU9250_asukiaaa*, https://github.com/asukiaaa/MPU9250_asukiaaa, Accessed: 2024-09-17, 2024.

[46] S. L. Brunton, J. L. Proctor y J. N. Kutz, "Sparse Identification of Nonlinear Dynamical Systems from Data," *Proceedings of the National Academy of Sciences*, vol. 113, n.º 15, págs. 3932-3937, 2016.

[47] P. J. Baddoo, B. Herrmann, B. J. McKeon y S. L. Brunton, "Kernel Learning for Robust Dynamic Mode Decomposition: Linear and Nonlinear Disambiguation Optimization (LANDO)," *arXiv preprint arXiv:2002.05935*, 2020.

CAPÍTULO 13

Anexos

Este cápitulo se realizó con el fin de incluir información relevante a los conjuntos de datos empleados para captura de movimiento, pruebas adicionales del entrenamiento de los modelos de aprendizaje profundo, recursos y documentación del trabajo realizado. Principalmente, se consideraron figuras para hacer referencia a los datos generados por las unidades de medición inercial y los mejores casos de entrenamiento para los modelos de aprendizaje profundo.

13.1. Conjuntos de datos generados para unidades de medición inercial

A continuación se encuentran los conjuntos de datos recopilados con los dispositivos para captura de movimiento inercial. En particular, se muestran los datos generados por las unidades de medición inercial de aceleración, velocidad angular e intencidad magnetica.

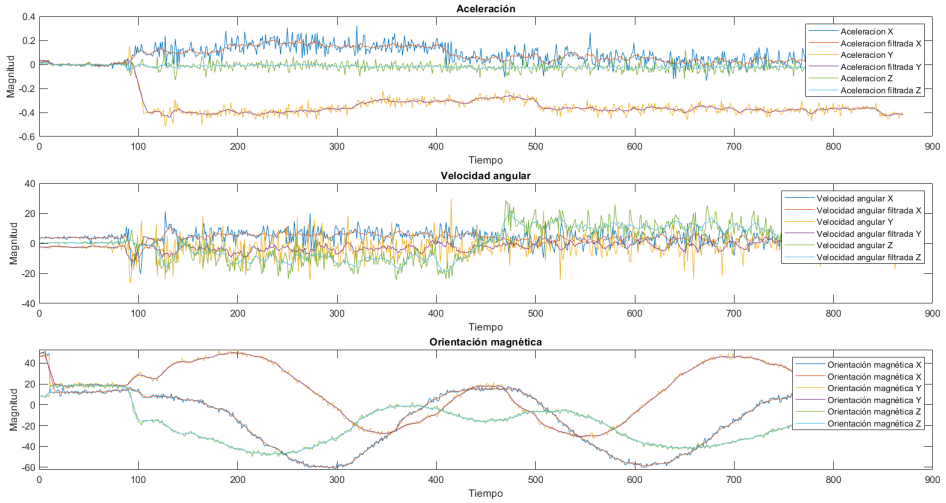

Figura 72: Lecturas de unidad de medición inercial en conjunto de datos de círculos

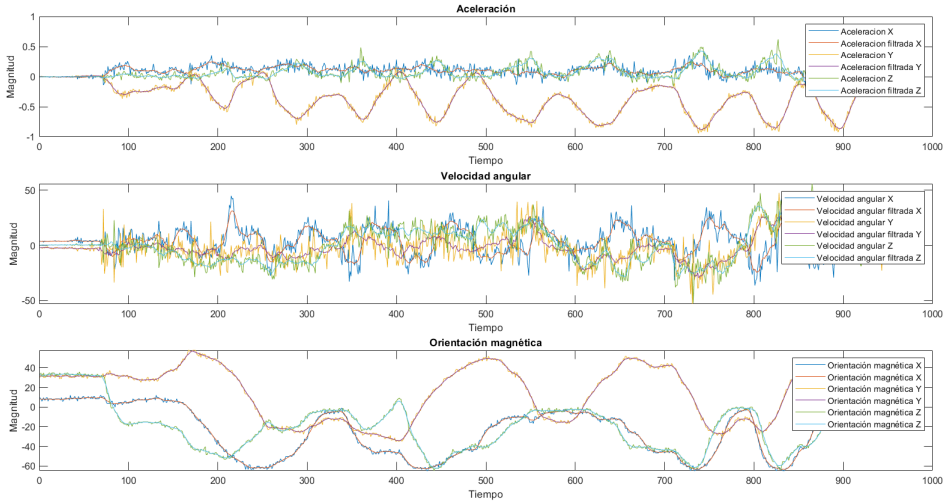

Figura 73: Lecturas de unidad de medición inercial en conjunto de datos de círculos y ondas

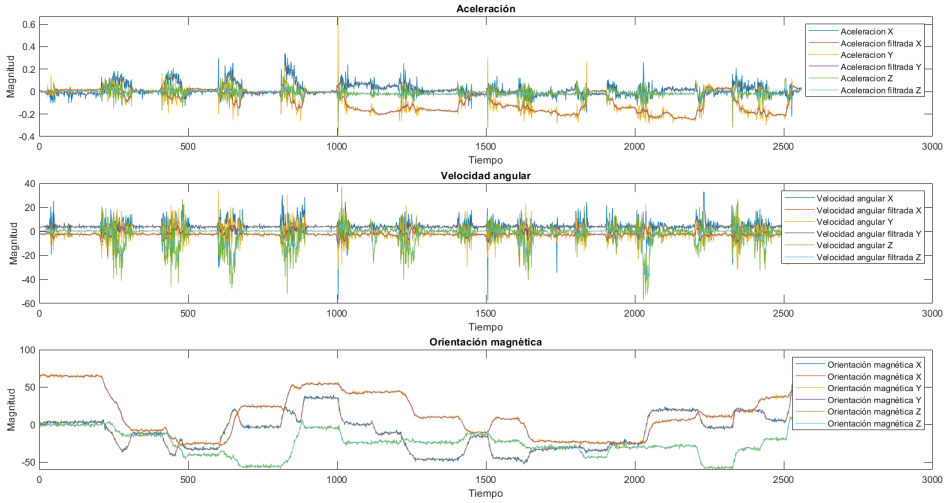

Figura 74: Lecturas de unidad de medición inercial en conjunto de datos de cubo

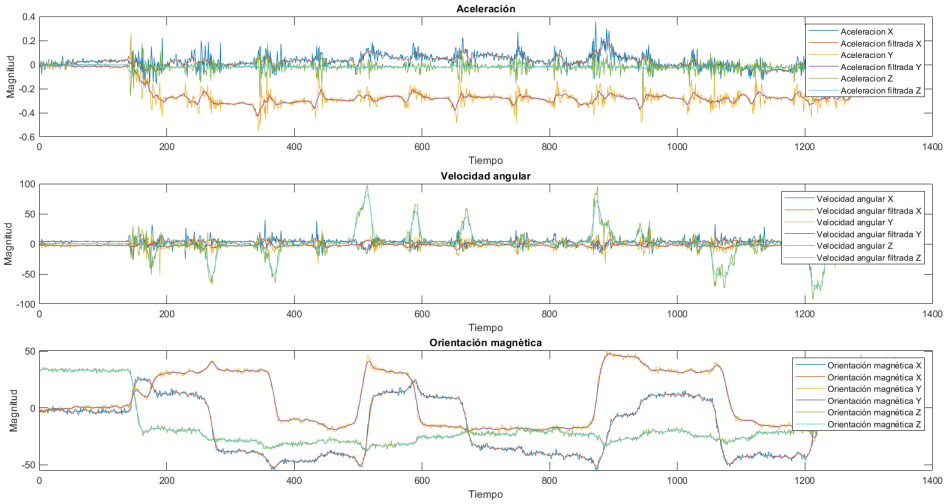

Figura 75: Lecturas de unidad de medición inercial en conjunto de datos de líneas

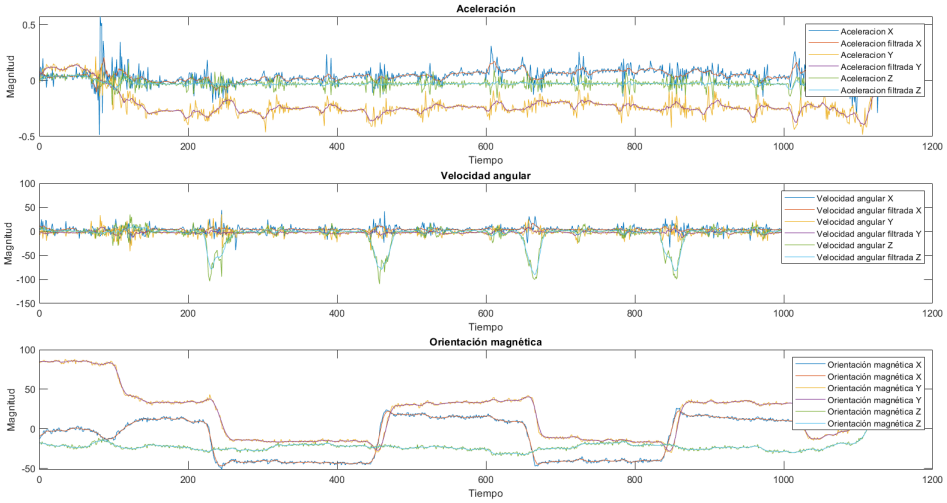

Figura 76: Lecturas de unidad de medición inercial en conjunto de datos de líneas en intervalos

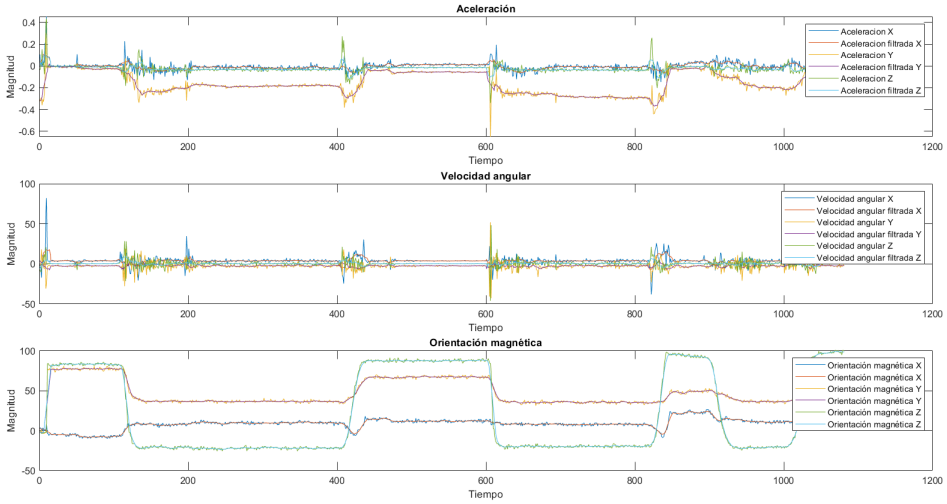

Figura 77: Lecturas de unidad de medición inercial en conjunto de datos de pulsos

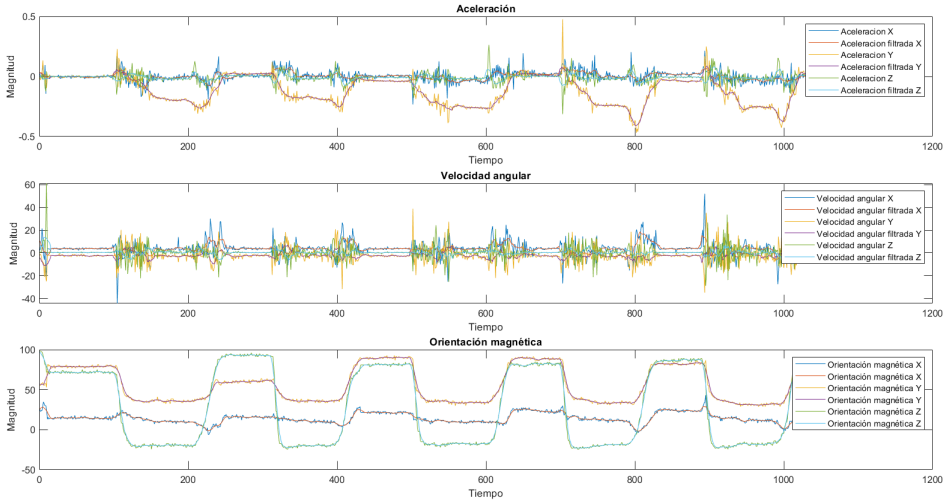

Figura 78: Lecturas de unidad de medición inercial en conjunto de datos de intervalo de pulsos

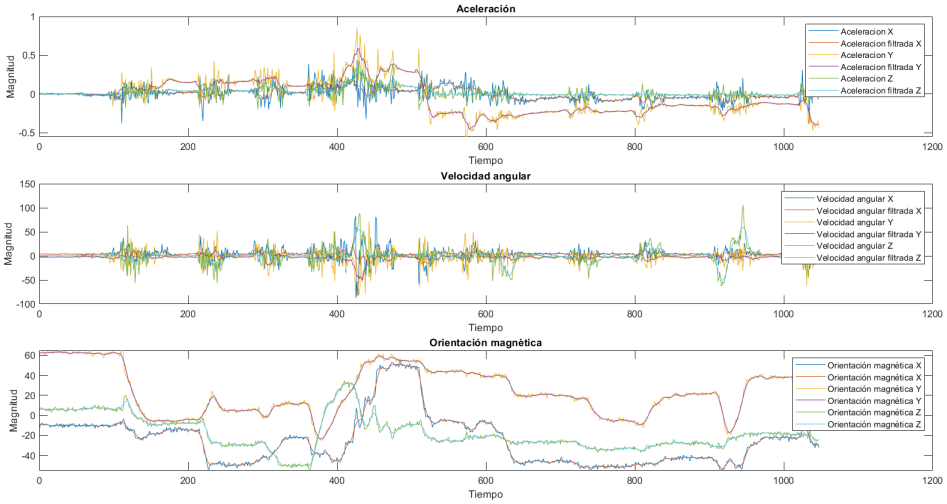

Figura 79: Lecturas de unidad de medición inercial en conjunto de datos de cuadrado

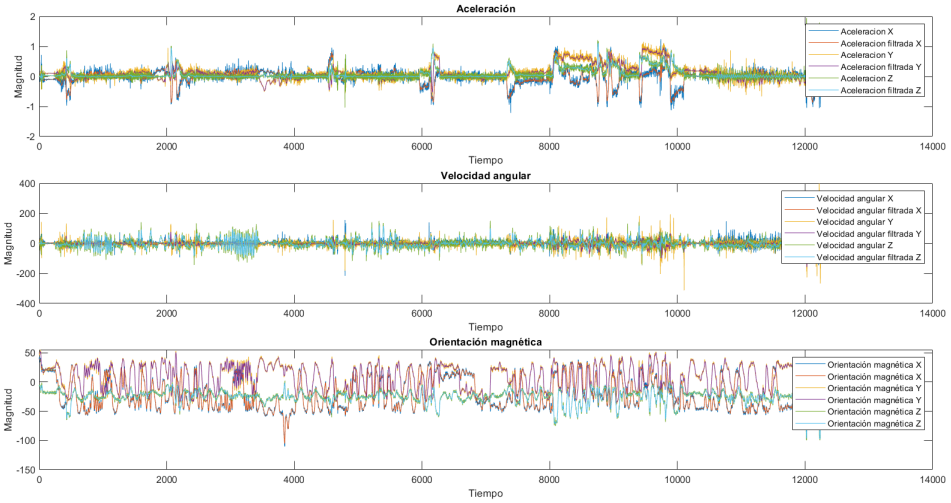

Figura 80: Lecturas de unidad de medición inercial en conjunto de datos para prueba de predicción

13.2. Mejores casos obtenidos con modelos de aprendizaje profundo

En esta sección se ilustran los mejores casos que se observaron entre los modelos generados. En este caso, se toma como referencia el conjunto de datos generado para poner a prueba los modelos de aprendizaje profundo que se muestra en la Figura 80.

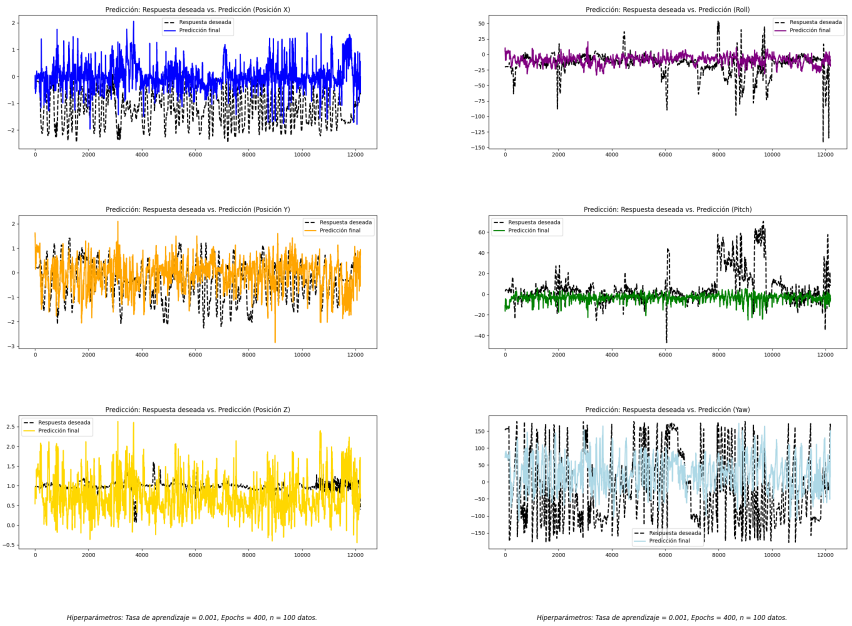

Figura 81: Estimación de pose en módulo de predicción de red convolucional unidimensional en mejor caso

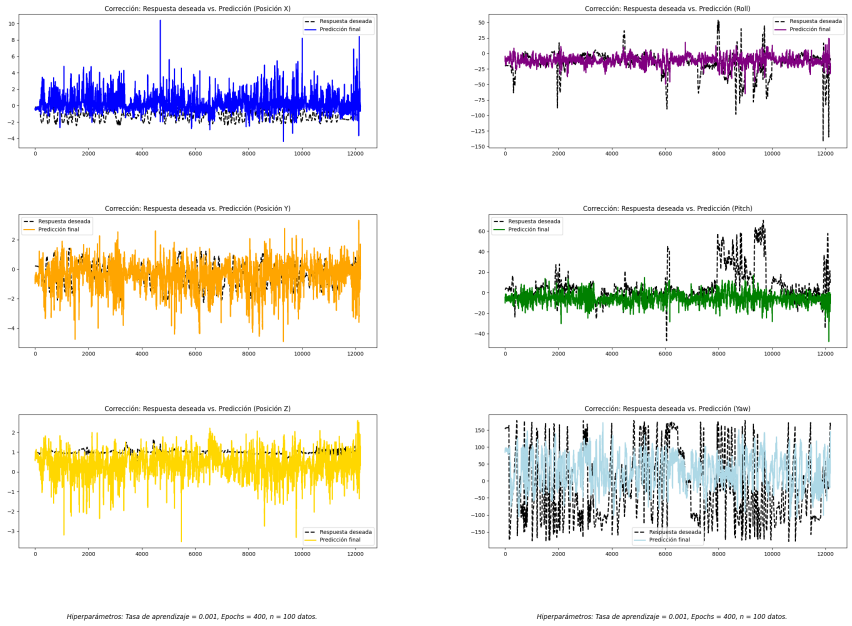

Figura 82: Estimación de pose en módulo de corrección de red convolucional unidimensional en mejor caso

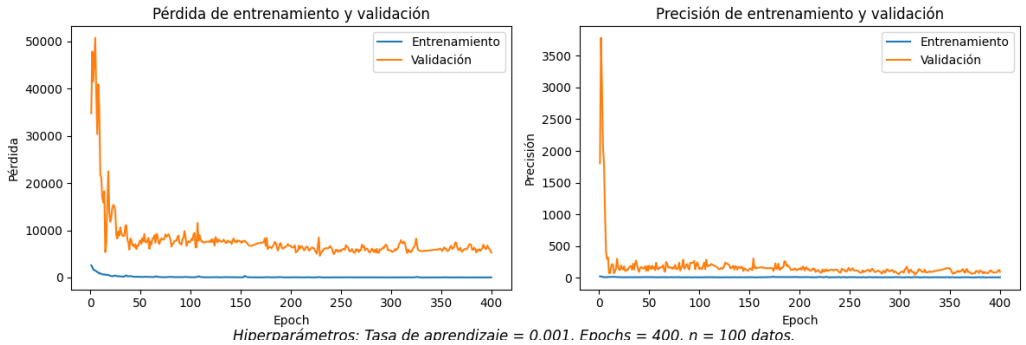

Figura 83: Curvas de aprendizaje y precisión de red convolucional unidimensional en mejor caso

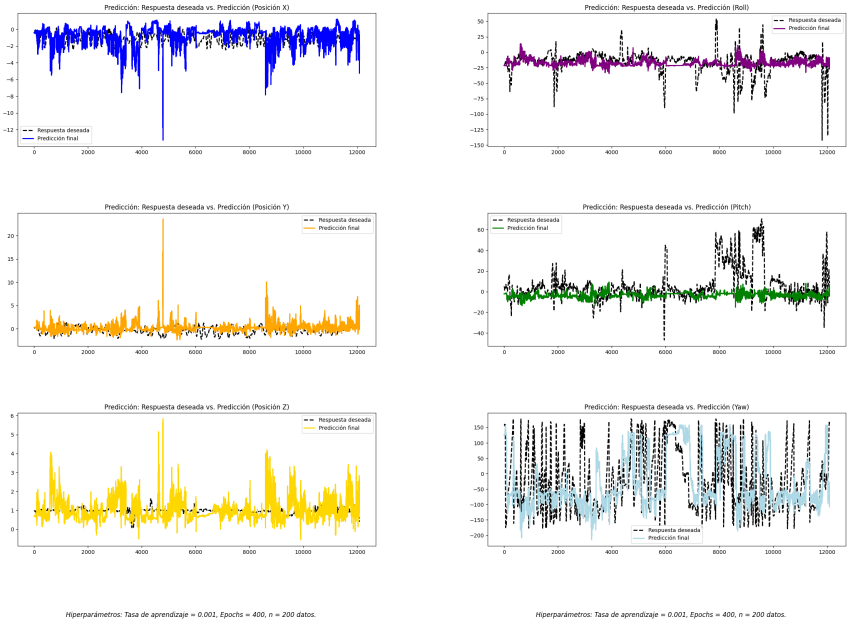

Figura 84: Estimación de pose en módulo de predicción de red residual con mecanismo de atención en mejor caso

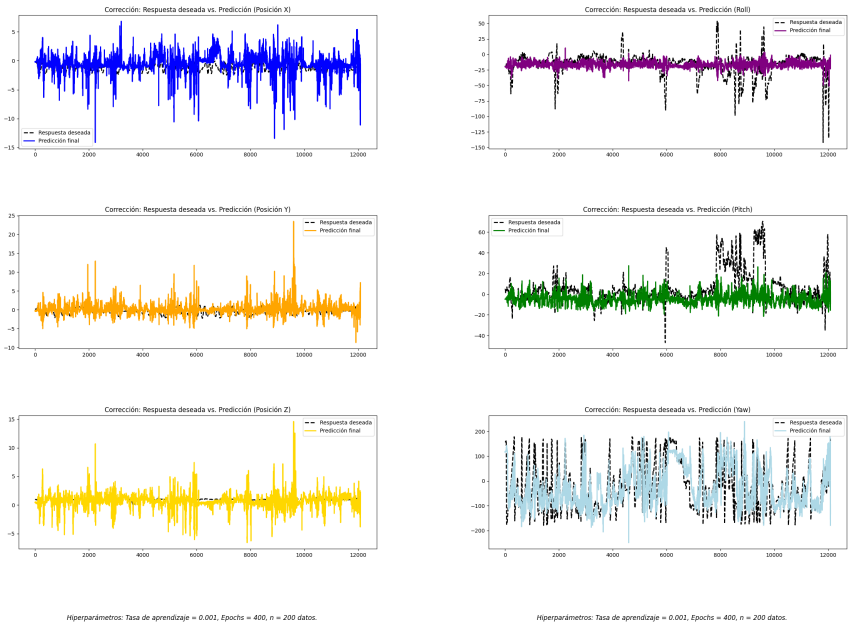

Figura 85: Estimación de pose en módulo de corrección de red residual con mecanismo de atención en mejor caso

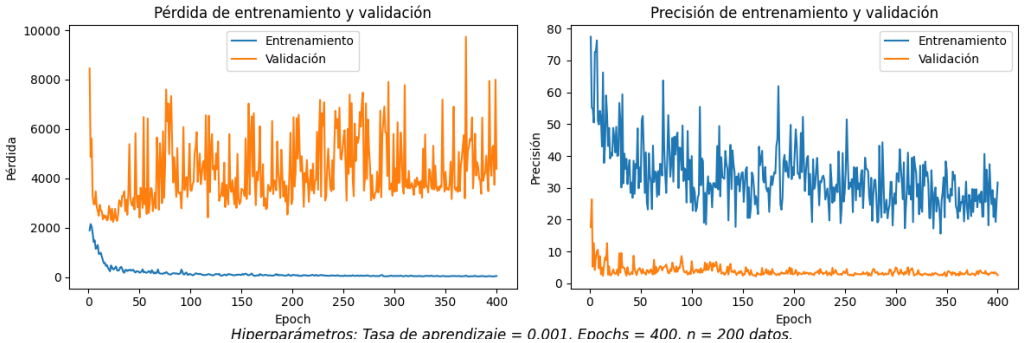

Figura 86: Curvas de aprendizaje y precisión de red residual con mecanismo de atención en mejor caso

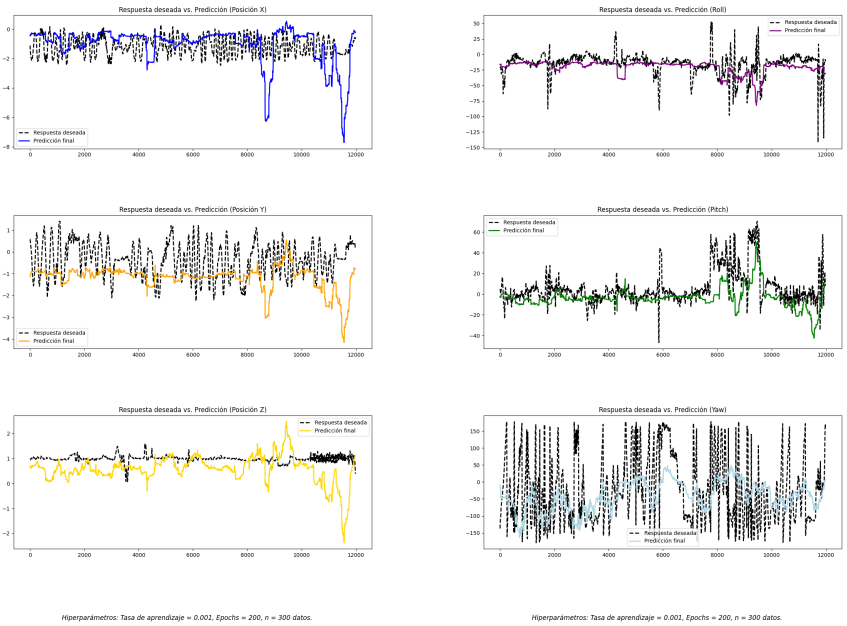

Figura 87: Estimación de pose en módulo de corrección de red convolucional temporal en mejor caso

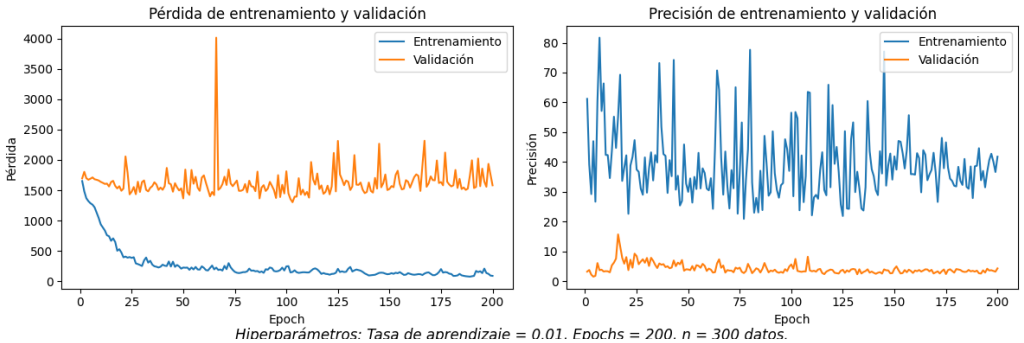

Figura 88: Curvas de aprendizaje y precisión de red convolucional temporal en mejor caso

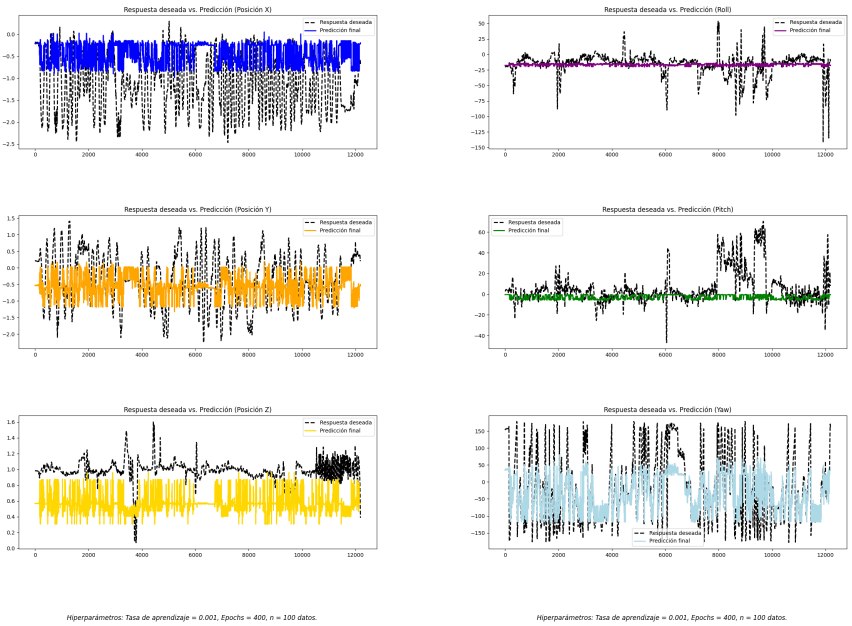

Figura 89: Estimación de pose en módulo de corrección de red convolucional temporal con mecanismo de atención en mejor caso

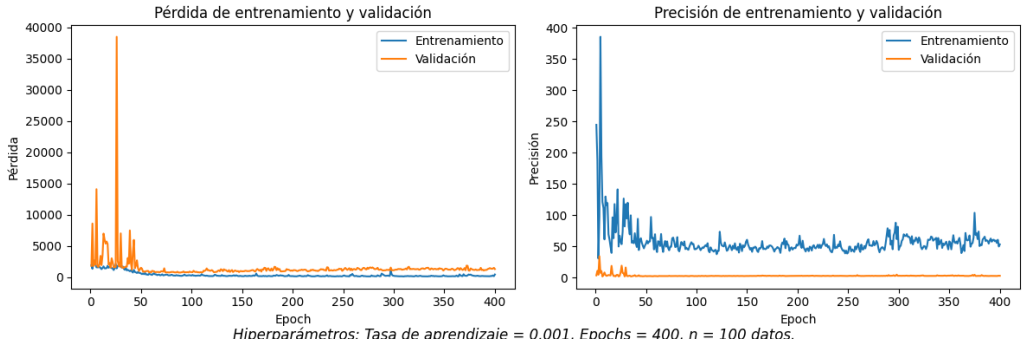

Figura 90: Curvas de aprendizaje y precisión de red convolucional temporal con mecanismo de atención en mejor caso

13.3. Documentación y repositorio

El siguiente enlace incluye la documentación de los programas y modelos CAD empleados en este trabajo.

`https://uvggt.sharepoint.com/:f:/r/sites/Test399/Documentos%20compartidos/15%20-%20Robotat/otros/mocap_inercial_dl/matheu_2024?csf=1&web=1&e=F4dANR`