

La investigación reportada en esta tesis es parte de los programas de investigación del CICESE (Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California).

La investigación fue financiada por el CONAHCYT (Consejo Nacional de Humanidades, Ciencias y Tecnologías).

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México). El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo o titular de los Derechos de Autor.

CICESE © 2023, Todos los Derechos Reservados, CICESE

Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California



Maestría en Ciencias en Ciencias de la Computación

Localización de robots móviles en interiores utilizando aprendizaje de máquina

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de
Maestro en Ciencias

Presenta:

Harumi Andrés Pino Urakami

Ensenada, Baja California, México

2023

Tesis defendida por

Harumi Andrés Pino Urakami

y aprobada por el siguiente Comité

Dr. Ubaldo Ruiz López

Director de tesis

Dr. Hugo Homero Hidalgo Silva

Dr. Javier Pliego Jiménez



Dr. Pedro Gilberto López Mariscal

Coordinador del Posgrado en Ciencias de la Computación

Dra. Ana Denise Re Araujo

Directora de Estudios de Posgrado

Resumen de la tesis que presenta Harumi Andrés Pino Urakami como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Localización de robots móviles en interiores utilizando aprendizaje de máquina

Resumen aprobado por:

Dr. Ubaldo Ruiz López
Director de tesis

Uno de los problemas fundamentales en el área de la robótica móvil es el de la localización, siendo la base de la navegación autónoma y planificación de trayectorias. Mientras que en ambientes exteriores tecnologías como GPS, trilateración o triangulación son ampliamente utilizadas, en ambientes interiores no son aplicables. Los métodos más comunes para abordar este problema son los algoritmos SLAM basados en sensores visuales, como cámaras RGBD o sensores de rango láser. Sin embargo, estos métodos son susceptibles a fallos en ambientes con iluminación pobre, paredes uniformes, carentes de detalle o transparentes. Es por lo anterior que en este trabajo se propone utilizar un método de aprendizaje de máquina que tome como entrada datos de las señales Wi-Fi cercanas y datos inerciales para estimar la posición de un robot móvil. Se implementó un modelo para simular la intensidad de señales Wi-Fi en el ambiente, así como un simulador robótico para generar los datos de entrenamiento y prueba para los modelos de localización. Los mejores resultados se obtuvieron utilizando la arquitectura MobileNetV3 para estimar el desplazamiento del robot utilizando solamente datos inerciales, teniendo una media de error RMS de 0.9794 a través de 10 trayectorias de prueba en el espacio vacío. Después de un segundo entrenamiento, ResNet-18 obtuvo los mejores resultados en dos recorridos de prueba en un ambiente con obstáculos, con una media de error RMS de 0.4571.

Palabras clave: Localización, aprendizaje de máquina, sensor inercial, señal Wi-Fi, robot móvil

Abstract of the thesis presented by Harumi Andrés Pino Urakami as a partial requirement to obtain the Master of Science degree in Computer Science.

Indoor mobile robot localization using machine learning

Abstract approved by:

PhD Ubaldo Ruiz López

Thesis Director

Localization is one of the fundamental problems in mobile robotics, being the basis of autonomous navigation and trajectory planning. While in outdoor environments, technologies such as GPS, trilateration, or triangulation are widely used, they are not applicable in indoor environments. The most common methods to address this problem are SLAM algorithms based on visual sensors, such as RGBD cameras or laser ranging sensors. However, these methods are susceptible to failure in environments with poor lighting and regions without texture or transparent walls. For this reason, in this work, it is proposed to use a machine learning method that takes nearby Wi-Fi signals and inertial data to estimate the position of a mobile robot. A model was implemented to simulate the intensity of Wi-Fi signals in the environment, as well as a robotic simulator to generate training and test data for the location models. The best results were obtained using the MobileNetV3 architecture to estimate the robot's displacement using only inertial data, having a mean RMS error of 0.9794 through 10 test trajectories in an empty space. After a second training, ResNet-18 obtained the best results in two test runs in an environment with obstacles, with a mean RMS error of 0.4571.

Keywords: Localization, machine learning, inertial sensor, Wi-Fi signal, mobile robot

Dedicatoria

A mis padres, hermanos y amigos, por el apoyo, las palabras de aliento y cariño que me han mostrado durante esta etapa.

Agradecimientos

Al Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California por la oportunidad y las herramientas para la elaboración de este trabajo.

Al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT) por brindarme el apoyo económico para realizar mis estudios de maestría con el No. de becario 29413.

A mi asesor, Dr Ubaldo Ruiz López, por la orientación, el apoyo y la atención recibida en la realización de este trabajo.

A mi comité de tesis, Dr. Hugo Homero Hidalgo Silva y Dr. Javier Pliego Jiménez, por las observaciones y comentarios.

Tabla de contenido

	Página
Resumen en español	ii
Resumen en inglés	iii
Dedicatoria	iv
Agradecimientos	v
Lista de figuras	viii
Lista de tablas	x
Capítulo 1. Introducción	
1.1. Motivación	1
1.2. Planteamiento del problema	2
1.3. Objetivos	3
1.3.1. Objetivo general	3
1.3.2. Objetivos específicos	3
1.4. Contribuciones	4
Capítulo 2. Marco teórico	
2.1. Fingerprinting	5
2.2. Multilateración	6
2.3. Unidad de medición inercial	7
2.4. K vecinos más cercanos	8
2.5. Máquina de vectores de soporte	8
2.6. Red neuronal artificial	10
2.6.1. Red neuronal convolucional	10
2.6.2. Red neuronal residual	10
2.6.3. Red neuronal <i>Long Short-Term Memory</i>	12
Capítulo 3. Trabajo relacionado	
3.1. Localización basada en Wi-Fi	13
3.2. Localización basada en datos inerciales	17
Capítulo 4. Localización basada en WiFi	
4.1. Metodología	24
4.1.1. Simulación de señales Wi-Fi	24
4.1.2. Fingerprinting	28
4.1.3. Multilateración	30
4.1.4. Multilateración utilizando agrupamiento de K-medias	32
4.2. Resultados	33
4.2.1. Fingerprinting	34
4.2.2. Multilateración	36
4.2.3. Multilateración utilizando agrupamiento de K-medias	38
4.3. Discusión	39

Capítulo 5. Localización basada en datos inerciales	
5.1. Metodología	41
5.1.1. Generación de datos	41
5.1.2. Modelos para localización	43
5.1.2.1. ResNet	44
5.1.2.2. MobileNetV3	45
5.1.2.3. Red CNN + LSTM	47
5.1.2.4. RBCN-Net	47
5.1.3. Configuración experimental y métricas	47
5.2. Resultados	49
5.3. Discusión	57
Capítulo 6. Conclusiones y trabajo futuro	
6.1. Trabajo futuro	59
Literatura citada	61

Lista de figuras

Figura	Página
1. Fase de calibración fuera de línea	5
2. Fase de localización en línea	6
3. Multilateración utilizando tres nodos.	6
4. Ejemplo de una unidad de medición inercial (IMU), mostrando el marco de referencia local del sensor.	8
5. Ejemplo de un modelo KNN para diferentes valores de k	9
6. Ejemplo de un modelo SVM para datos no separables linealmente	9
7. Operación de convolución realizada por una CNN	11
8. Estructura básica de una red neuronal residual	11
9. Unidad básica de una red LSTM	12
10. Descripción general del bosque difuso profundo	15
11. Arquitectura de DLoc	15
12. Arquitectura del sistema de localización basado en <i>fingerprints</i> utilizando CSI	16
13. Pseudocódigo para el algoritmo de <i>clustering</i> utilizando K-medias	17
14. Proceso de construcción de las firmas temporales	18
15. Arquitectura de IONet	19
16. Red neuronal que combina capas CNN y LSTM	19
17. Arquitectura de AbolDeepIO	20
18. Las tres diferentes arquitecturas de RoNIN	20
19. Diagrama de bloques del sistema TLIO	21
20. Diagrama de bloques del sistema PDRNet	22
21. Arquitectura de RBCN-Net	22
22. Ejemplo de nivel de intensidad de señales Wi-Fi capturadas de cuatro puntos de acceso diferentes durante el mismo recorrido	25
23. Comparación entre la intensidad de la señal Wi-Fi de un punto de acceso real y la intensidad calculada por el modelo Wi-Fi implementado.	26
24. Salida del modelo Wi-Fi para distancias de 1 a 50 metros	27
25. Puntos de acceso Wi-Fi distribuidos en el ambiente	28
26. Cuadrícula de puntos de referencia para <i>fingerprinting</i> , tomados cada 2 metros	29
27. Arquitectura del modelo empleado para el método <i>fingerprinting</i> , que toma como entrada una firma de valores RSSI para los n APs en el ambiente.	30

Figura	Página
28. Arquitectura del modelo empleado para el método de multilateración, que toma como entrada la posición de cada AP y la distancia de estos al punto a localizar.	31
29. Análisis del efecto del numero de grupos K en el error en localización para el método de multilateración utilizando agrupamiento de K-medias	33
30. Resultados para los tres métodos de aprendizaje de máquina utilizados para <i>fingerprinting</i>	34
31. Resultados para los dos métodos utilizados para multilateración.	37
32. Resultados para los dos métodos utilizados para multilateración con agrupamiento de K-medias	38
33. Robot <i>TurtleBot3</i> simulado utilizado para los experimentos.	42
34. Procesamiento para generar las ventanas de datos y cambios en la posición del robot, que serán utilizados como entradas y salidas del modelo respectivamente.	43
35. Estructura de los bloques (a) <i>Basic Block</i> y (b) <i>Bottleneck</i> , modificados para trabajar con datos 1D.	44
36. Arquitecturas de (a) ResNet-18, (b) ResNet-50 y (c) ResNet-Small	45
37. Arquitectura de general MobileNetV3	46
38. Ambiente simulado utilizado para recorridos de comparación con algoritmos de SLAM .	49
39. Trayectorias reconstruidas por todos los modelos para el recorrido de prueba #5	52
40. Recorrido de prueba #1, trayectoria real y estimadas por algoritmos de SLAM	53
41. Recorrido de prueba #1, trayectoria real y estimadas por modelos entrenados	53
42. Recorrido de prueba #2, trayectoria real y estimadas por algoritmos de SLAM	54
43. Recorrido de prueba #2, trayectoria real y estimadas por modelos entrenados	54
44. Ambiente con obstáculos utilizado para generar datos de entrenamiento adicionales. . . .	55
45. Recorrido de prueba #1, trayectoria real y estimadas por modelos entrenados con datos adicionales	56
46. Recorrido de prueba #2, trayectoria real y estimadas por modelos entrenados con datos adicionales	56

Lista de tablas

Tabla		Página
1.	Calidad de la señal en función del rango de valores RSSI	13
2.	Posibles valores del parámetro <i>path loss exponent</i> de acuerdo al tipo de ambiente a simular	25
3.	Resultados para el método de <i>fingerprinting</i> tomando puntos de referencia cada 0.5m, mostrando la media y desviación estándar del error en localización	35
4.	Resultados para el método de <i>fingerprinting</i> tomando puntos de referencia cada 1m, mostrando la media y desviación estándar del error en localización	35
5.	Resultados para el método de <i>fingerprinting</i> tomando puntos de referencia cada 1.5m, mostrando la media y desviación estándar del error en localización	36
6.	Resultados para el método de <i>fingerprinting</i> tomando puntos de referencia cada 2m, mostrando la media y desviación estándar del error en localización	36
7.	Resultados para el método de multilateración, mostrando la media y desviación estándar del error en localización.	37
8.	Resultados para el método de multilateración utilizando agrupamiento de K-medias.	39
9.	Resultados para recorridos de prueba #1 y #2	49
10.	Resultados para recorridos de prueba #3 y #4	50
11.	Resultados para recorridos de prueba #5 y #6	50
12.	Resultados para recorridos de prueba #7 y #8	50
13.	Resultados para recorridos de prueba #9 y #10	50
14.	Resultados promedio para todos los recorridos	51
15.	Resultados para comparación con algoritmos SLAM en recorridos de prueba #1 y #2	52
16.	Resultados para comparación con modelos mejorados en recorridos #1 y #2	55

Capítulo 1. Introducción

En este capítulo se presenta un panorama general del trabajo realizado. Se describe la motivación inicial para llevar a cabo esta investigación, se plantea cual es la problemática que se busca resolver y se dan a conocer los objetivos generales y específicos a alcanzar. Finalmente, se exponen las contribuciones obtenidas con la realización de este trabajo.

1.1. Motivación

Los robots móviles juegan un papel cada vez más importante en la vida diaria, siendo utilizados en aplicaciones en áreas distintas, tales como la médica, manufactura, minería, educación, entre otras (Ruan et al., 2019). La localización es uno de los problemas fundamentales en el área de la robótica móvil. Un método de localización robusto y preciso es la base para la navegación autónoma y planificación de trayectorias robóticas (Xiao et al., 2018).

El Sistema de Posicionamiento Global, o GPS por sus siglas en inglés, es ampliamente utilizado para aplicaciones de localización en exteriores. Sin embargo, las señales de satélite no penetran de manera adecuada en áreas interiores complejas, por lo que no es factible utilizar esta tecnología para localización en interiores. Otros métodos utilizados en exteriores como trilateración y la triangulación requieren de una línea de visión sin obstrucciones, por lo que su uso está limitado en interiores. Es por esto que ha sido necesario desarrollar técnicas específicas para aplicaciones de localización en ambientes interiores (Roy & Chowdhury, 2021).

El problema de Localización y Mapeo Simultáneos, o SLAM por sus siglas en inglés, es el problema que aborda simultáneamente la construcción del ambiente y la localización del robot en el mismo. Esta metodología ha sido ampliamente estudiada en las últimas décadas (Wen et al., 2020). Entre los primeros métodos que abordaban este problema y que siguen siendo de los más utilizados en la actualidad están los basados en el filtro de Kalman extendido (EKF). Hoy en día, existe una amplia variedad de técnicas empleadas para resolver el problema de SLAM basadas principalmente en sensores de visión o cámaras. Entre estas se pueden destacar GraphSLAM (Thrun & Montemerlo, 2006), FastSLAM (Rodríguez-Losada et al., 2007), RTAB-MAP (Labbé, 2013) y ORB-SLAM3 (Campos et al., 2021). No obstante, los métodos que hacen uso de cámaras poseen una serie de desafíos al realizar SLAM en interiores. Por un lado, corredores con paredes blancas, patrones de textura o luces repetidas pueden ocasionar confusión entre lugares similares, lo que resulta en mapas creados incorrectamente y mala localización (Hashemifar et al., 2019).

Otro de los sensores más utilizados en localización son los LiDAR (*Laser Imaging Detection and Ranging*), que pueden determinar la distancia a un objeto utilizando un haz de luz láser. Sin embargo, los métodos que utilizan este tipo de sensores tienden a fallar en ambientes donde se tengan paredes o superficies transparentes o reflejantes, barras o polvo en el aire (Dobrev et al., 2018). Además, tanto cámaras como LiDAR producen grandes volúmenes de información, lo que hace a estos métodos computacionalmente más desafiantes, especialmente en dispositivos con recursos limitados.

Los problemas descritos anteriormente han motivado el surgimiento de otros métodos de localización, tales como localización Wi-Fi o basados en odometría inercial. La localización por señales Wi-Fi se ha vuelto una opción popular debido a la disponibilidad de los puntos de acceso Wi-Fi en todo tipo de lugares, tales como oficinas, escuelas, industrias, entre otros. Por su parte, los sensores empleados para localización se han vuelto cada vez más económicos y compactos, por lo que resulta sencillos incorporarlos en cualquier tipo de robot móvil o vehículo.

Por otro lado, gracias a los recientes avances en inteligencia artificial, diferentes técnicas de aprendizaje de máquina han sido aplicadas con éxito en problemas de localización de robots móviles, como fusión de datos de diferentes sensores (Li et al., 2021), corrección de datos de sensores (Brossard & Bonnabel, 2019), síntesis de datos faltantes (Pfeiffer et al., 2018), entre otros.

1.2. Planteamiento del problema

Los métodos más populares para localización de robots en interiores están basados en visión y mediciones láser. Sin embargo, estos métodos tienden a fallar en ambientes con iluminación pobre, paredes carentes de textura, gran cantidad de obstáculos u oclusiones, o en aplicaciones militares donde el uso de sensores activos no está permitido (Brossard & Bonnabel, 2019).

Como se mencionó anteriormente, estos problemas han motivado el desarrollo de otros métodos de localización, tales como localización Wi-Fi o utilizando odometría inercial. La localización por señales Wi-Fi se ha vuelto una opción popular debido a la disponibilidad de los puntos de acceso Wi-Fi en todo tipo de lugares. Además, los sensores empleados tanto para localización por Wi-Fi como odometría inercial resultan ser más económicos que las cámaras o láser necesarios para otras técnicas.

No obstante, estos métodos también tienen sus problemas. Por un lado, la localización por señales Wi-Fi resulta ser vulnerable a las dinámicas del ambiente, condiciones ambientales, distribución de los ocupantes y movimiento de estos (Zhang et al., 2020). Por otro lado, el error de los sensores utilizados

para odometría inercial, por pequeños que sean, se acumulan a lo largo del tiempo, dando como resultado una deriva que crece sin límites (Atia et al., 2012).

Para superar las deficiencias en el uso de sensores individuales en distintas aplicaciones, se ha implementado fusión de sensores utilizando diversas técnicas de aprendizaje de máquina, tal como la fusión de datos de sensores inerciales y láser (Li et al., 2021). Sin embargo, no se ha explorado de manera extensiva el uso de técnicas de aprendizaje de máquina para la fusión de sensores Wi-Fi y odometría inercial para la localización de robots en interiores, por lo que resulta ser un área de investigación de interés. Este trabajo se enfocará entonces en desarrollar un método de fusión de datos de sensores Wi-Fi e inerciales como base para un algoritmo de localización en interiores de un robot móvil.

1.3. Objetivos

1.3.1. Objetivo general

Implementar un método de aprendizaje de máquina que utilice los datos de un sensor inercial y los puntos de acceso Wi-Fi cercanos en el ambiente, y entregue como resultado la localización aproximada de un robot móvil en un ambiente interior con un mapa conocido.

1.3.2. Objetivos específicos

- Crear una base de datos de valores capturados por sensores Wi-Fi e inerciales por un robot para diferentes ambientes, la cual contendrá también la pose real del robot como datos *ground truth*.
- Implementar un método de aprendizaje de máquina que tome como entrada valores inerciales y los capturados de la red Wi-Fi, y entregue como salida la localización aproximada del robot con respecto a su ambiente.
- Evaluar el método propuesto en ambientes simulados utilizando la plataforma robótica *TurtleBot3*.
- Comparar los resultados obtenidos con otros métodos de localización en la literatura.

1.4. Contribuciones

Las contribuciones principales de este trabajo son las siguientes:

- Implementación de un modelo para simulación de señales Wi-Fi en el simulador robótico *Gazebo*.
- Comparación de diferentes métodos de aprendizaje de máquina para localización utilizando señales Wi-Fi y análisis de la viabilidad de estos para localización robótica.
- Una base de datos con alrededor de 16 horas de recorridos, realizado con el robot simulado *TurtleBot3*, que contiene datos inerciales y la posición real del robot.
- Implementación de distintos modelos de aprendizaje profundo para localización robótica utilizando solamente datos inerciales.
- Comparación de los mejores modelos entrenados utilizando aprendizaje profundo y algunos métodos de SLAM.

Capítulo 2. Marco teórico

En este capítulo se describen algunos los conceptos básicos, técnicas y tecnologías utilizadas durante el transcurso de la investigación.

2.1. Fingerprinting

Fingerprinting es un método de localización que emplea los niveles de intensidad de las señales Wi-Fi de los puntos de acceso de red (APs por sus siglas en inglés) para estimar la posición del robot móvil con respecto a su ambiente. Este método involucra dos fases: la fase de calibración fuera de línea y la fase de localización en línea. Durante la calibración fuera de línea, se definen una multitud de posiciones de referencia en el ambiente en el que se desea realizar la localización. En cada una de estas posiciones se capturan los niveles de intensidad de las señales provenientes de los APs cercanos, creando una “firma” que se asocia a cada posición. Estas firmas generan una base de datos que se emplea para entrenar algún modelo de aprendizaje de máquina, el cual se utilizará en la fase de localización en línea. Esta primera fase se ilustra en la figura 1.

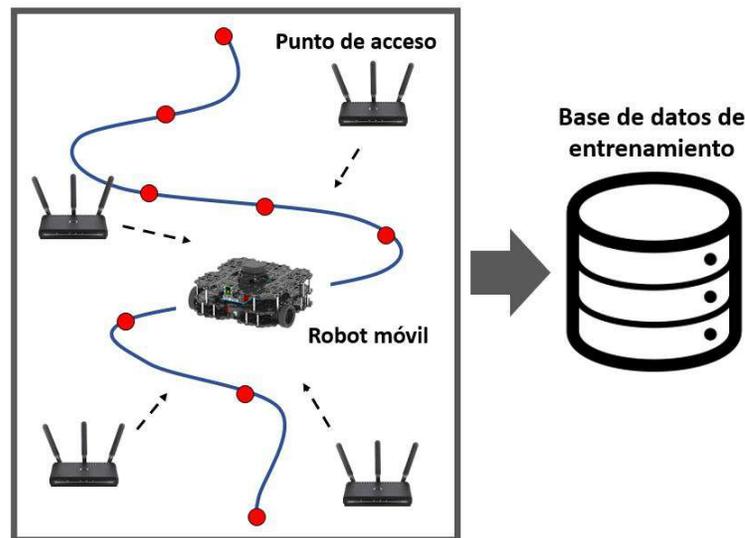


Figura 1. Fase de calibración fuera de línea. Los puntos en rojo en la trayectoria del robot representan los puntos de referencia, en cada uno de los cuales se captura la firma Wi-Fi y la posición correspondiente. Estas firmas se utilizan para construir la base de datos de entrenamiento.

Durante la fase de localización en línea, el robot móvil se encuentra en una posición desconocida en el ambiente, desde la cual se capturan los niveles de intensidad de los APs cercanos. La firma formada por los valores capturados funciona como la entrada del modelo entrenado, el cual genera como salida la localización estimada del robot móvil con respecto a su ambiente. Esta fase se ilustra en la figura 2.

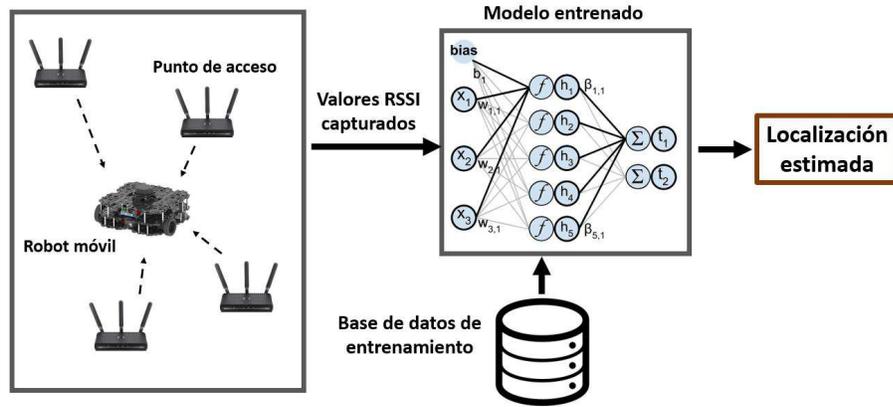


Figura 2. Fase de localización en línea. El robot se encuentra en una posición desconocida del ambiente, en la cual se captura la firma Wi-Fi correspondiente. Esta firma es recibida como entrada por el modelo entrenado durante la fase de calibración fuera de línea, y entrega como salida la localización estimada del robot.

2.2. Multilateración

Se le conoce como multilateración al proceso de calcular la posición de un punto desconocido utilizando las distancias de este a otros puntos de referencia conocidos como nodos. Para poder determinar la posición de un punto en el espacio 2D se requieren como mínimo tres nodos distintos, mientras que para un espacio 3D se requieren al menos cuatro.

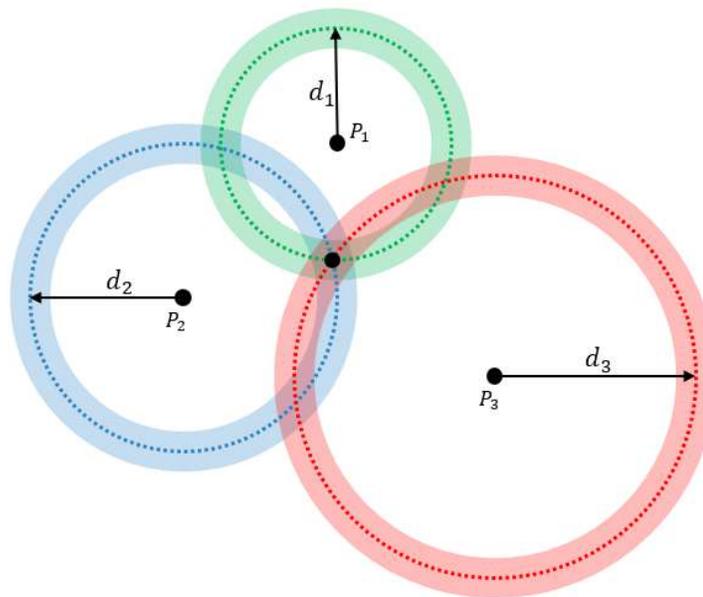


Figura 3. Multilateración utilizando tres nodos, donde la intersección de los tres círculos punteados corresponde al punto a localizar. Las áreas de color representan el posible error en las mediciones de distancia, debido al cual la posición estimada por multilateración estará en algún lugar de la intersección de las áreas sombreadas.

En aplicaciones reales, las distancias utilizadas para el calculo son estimaciones, por lo que la posición obtenida es una aproximación con un nivel de error que dependerá del error en las mismas. Para minimizar este error, es necesario aumentar el número de nodos. Suponiendo que se tienen n nodos distintos, se pueden definir n ecuaciones con la siguiente forma:

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = d_i^2 \quad (1)$$

donde x, y y z son las coordenadas del punto cuya posición deseamos estimar, x_i, y_i y z_i son las coordenadas del nodo i , y d_i es la distancia estimada del nodo i al punto desconocido. Estas n ecuaciones, después de una serie de manipulaciones algebraicas, se pueden representar de la siguiente manera:

$$2 \begin{bmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ \vdots & \vdots & \vdots \\ x_n - x_1 & y_n - y_1 & z_n - z_1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d_1^2 - d_2^2 - k_1 + k_2 \\ d_1^2 - d_3^2 - k_1 + k_3 \\ \vdots \\ d_1^2 - d_n^2 - k_1 + k_n \end{bmatrix} \quad (2)$$

donde $k_i = x_i^2 + y_i^2 + z_i^2$.

Esta ecuación tiene la forma conocida $A \cdot x = b$, por lo que una posición aproximada \hat{p} puede ser calculada por mínimos cuadrados utilizando la siguiente fórmula:

$$\hat{p} = (A^T A)^{-1} A^T b \quad (3)$$

2.3. Unidad de medición inercial

Una unidad de medición inercial (IMU por sus siglas en inglés) es un dispositivo electrónico compuesto de giroscopios, acelerómetros, y, ocasionalmente, magnetómetros, que mide y reporta las aceleraciones, velocidades angulares y orientación del objeto al que se encuentre sujeto. Los últimos avances en tecnología han permitido la fabricación de IMUs cada vez más pequeños y de menor costo, lo que ha hecho que sean ampliamente utilizados en todo tipo de dispositivos móviles, desde teléfonos celulares, hasta robots. Un IMU reporta como mínimo seis valores diferentes: aceleración lineal en los ejes x , y y z , normalmente medida en m/s^2 , y velocidad angular en los ejes x , y y z , normalmente medida en rad/s .

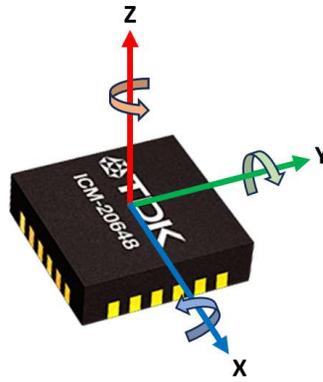


Figura 4. Ejemplo de una unidad de medición inercial (IMU), mostrando el marco de referencia local del sensor.

2.4. K vecinos más cercanos

K vecinos más cercanos es un algoritmo de aprendizaje supervisado no paramétrico utilizado para clasificación y regresión. En este método se toman todos los ejemplos de entrenamiento y se colocan en una tabla. Para el caso de un problema de clasificación, dada una consulta x_q , se encuentran los k ejemplos más cercanos a x_q , se toma la clase mayoritaria entre los ejemplos y se le asigna esta a la consulta x_q . En el caso de regresión, se toma la media o mediana de los k vecinos y se la asigna esta a x_q (Russell & Norvig, 2010). Para medir la distancia entre la consulta x_q y los ejemplos de la tabla, se utiliza típicamente la distancia Minkowski.

$$L^p(X_j, X_q) = \left(\sum_i |x_{j,i} - x_{q,i}|^p \right)^{1/p} \quad (4)$$

2.5. Máquina de vectores de soporte

Máquina de vectores de soporte (SVM por sus siglas en inglés) es otro método de aprendizaje supervisado no paramétrico. Una SVM construye un separador de margen máximo, es decir, una frontera de decisión que tiene la distancia más grande posible a los puntos de ejemplo. Las SVM tienen la capacidad de proyectar los datos a una dimensión más alta, utilizando lo que se le conoce como un *kernel*, de tal manera que en este espacio de alta dimensionalidad los elementos de diferentes clases sean linealmente separables por un hiperplano (Russell & Norvig, 2010). En el caso de regresión la idea es la misma, solamente que en lugar de encontrar el hiperplano que separe las clases de datos, se busca encontrar el hiperplano que mejor se ajuste a estos.

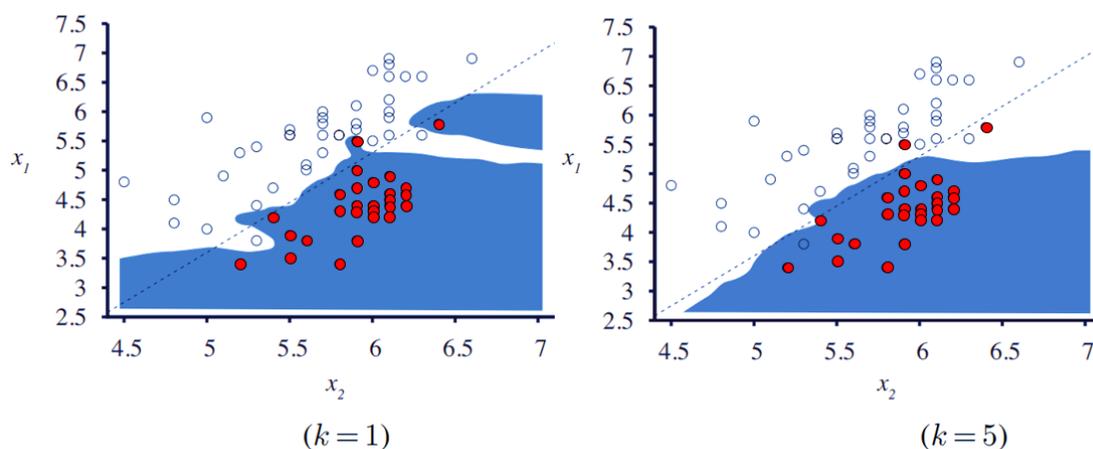


Figura 5. Ejemplo de un modelo KNN para un problema de clasificación binaria. Los círculos rojos representan las muestras de la clase A, y los azules las de clase B. El área sombreada representa la frontera de decisión. Con $k = 1$ (a), la frontera de decisión muestra sobreajuste, mientras que con $k = 5$ (b) este problema desaparece para los datos considerados. Tomado de Russell & Norvig (2010)

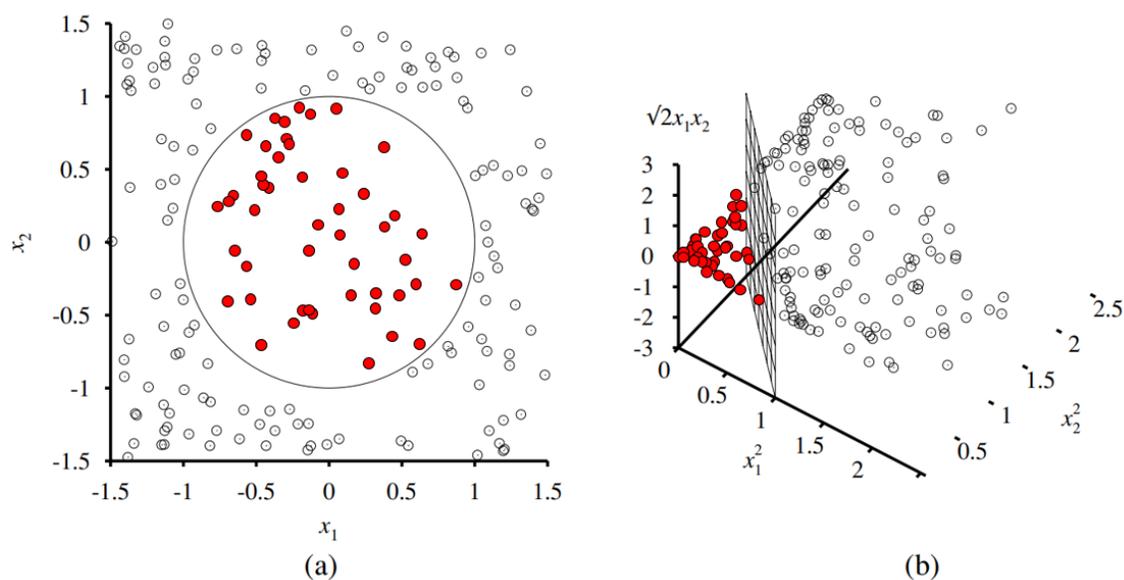


Figura 6. Ejemplo de un modelo SVM para un problema de clasificación binaria. Los círculos rojos representan las muestras de la clase A, y los negros las de la clase B. En (a) se observa como las clases no son linealmente separables, siendo la frontera de decisión un círculo. En (b) se muestra como al proyectar los mismos datos en un espacio tridimensional $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$, la frontera de decisión se vuelve lineal. Tomado de Russell & Norvig (2010).

2.6. Red neuronal artificial

Una red neuronal artificial está compuesta de unidades conectadas unas con otras. Cada una de estas conexiones tiene un peso numérico asociado, que determina la fuerza con la que la señal se propaga a través de esa conexión. Cada unidad calcula su salida haciendo una suma ponderada de sus entradas y luego aplica una función de activación (Russell & Norvig, 2010). Podemos dividir las redes neuronales en dos grupos generales: prealimentadas (*feedforward* en inglés) y recurrentes. Las redes neuronales *feedforward* tiene conexiones solamente en una dirección, es decir, las unidades de una capa reciben información solamente de la capa anterior, no se forman ciclos. Por otro lado, las redes recurrentes retroalimentan sus salidas hacia sus entradas, formando ciclos. Existen muchos tipos y arquitecturas de redes neuronales diferentes, de las cuales se mencionan algunas a continuación.

2.6.1. Red neuronal convolucional

Una red neuronal convolucional (o CNN por sus siglas en inglés) es una red del tipo *feedforward*, compuesta de una serie de capas cuya función es aprender las características de los datos de entrada, con distintos niveles de abstracción. Las primeras capas de la red se encargan de extraer las características de alto nivel, mientras que las capas más profundas extraen las características de más bajo nivel (Ghosh et al., 2019). La operación principal de este tipo de red es la convolución, en la que un *kernel* (una matriz numérica) se desliza a través de los datos de entrada (por ejemplo, una imagen), calculando el producto punto en cada paso y formando un mapa de características como salida.

2.6.2. Red neuronal residual

Una red neuronal residual contiene en ciertos bloques conexiones directas desde la entrada hasta la salida, saltándose las capas que se encuentren en medio. Estas conexiones ayudan a prevenir el problema de “degradación” del rendimiento de una red que ocurre al apilar un alto número de capas, permitiendo crear modelos mucho más profundos (He et al., 2016).

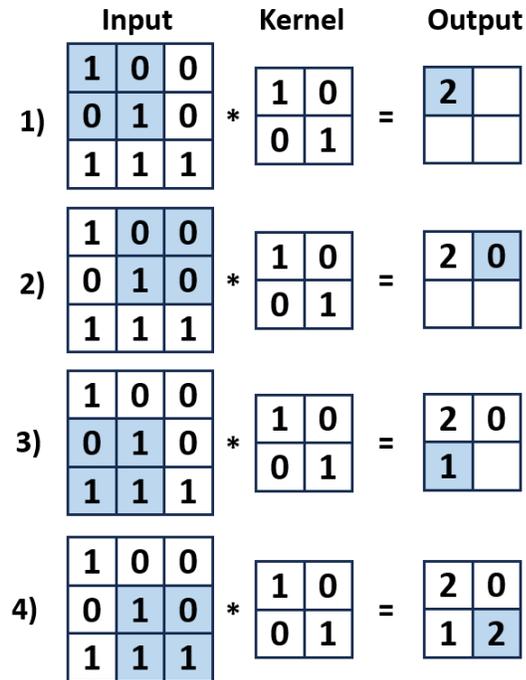


Figura 7. Operación de convolución realizada por una CNN. En cada uno de los pasos, los elementos sombreados de la entrada se multiplican por los elementos del kernel. Estos productos se suman, y el resultado obtenido se asigna al elemento correspondiente en la matriz de salida.

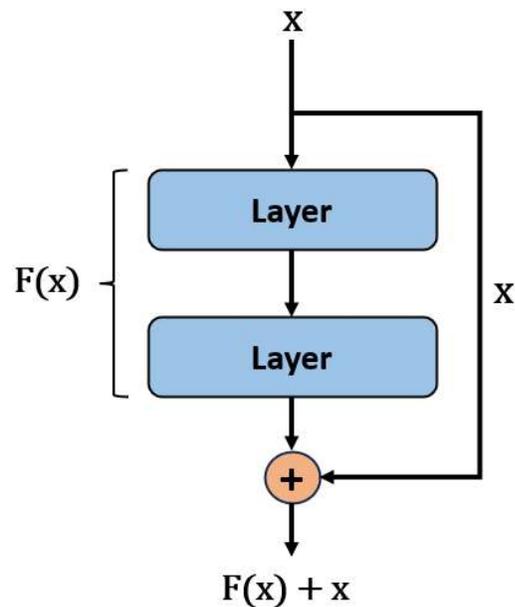


Figura 8. Estructura básica de una red neuronal residual. El pasar los datos de entrada a través de las capas de la red es equivalente a aplicar sobre estos una función $F(x)$. En una red residual, al resultado de esta función se le suma la entrada sin modificar, obteniendo a la salida $F(x) + x$.

2.6.3. Red neuronal Long Short-Term Memory

Una red *Long Short-Term Memory* (o LSTM) es una red del tipo recurrente capaz de retener dependencias entre los datos de entrada a largo plazo, a diferencia de otros tipos de redes recurrentes que solo son capaces de almacenar dependencias a corto plazo debido al problema de desvanecimiento de gradiente (Bengio et al., 1994). Esto lo logran almacenando información seleccionada en una celda, que luego puede ser utilizada o modificada por la red (Jozefowicz et al., 2015). Esta capacidad de aprender dependencias a largo plazo las ha hecho una opción muy popular para problemas que involucran procesamiento de lenguaje natural y series de tiempo.

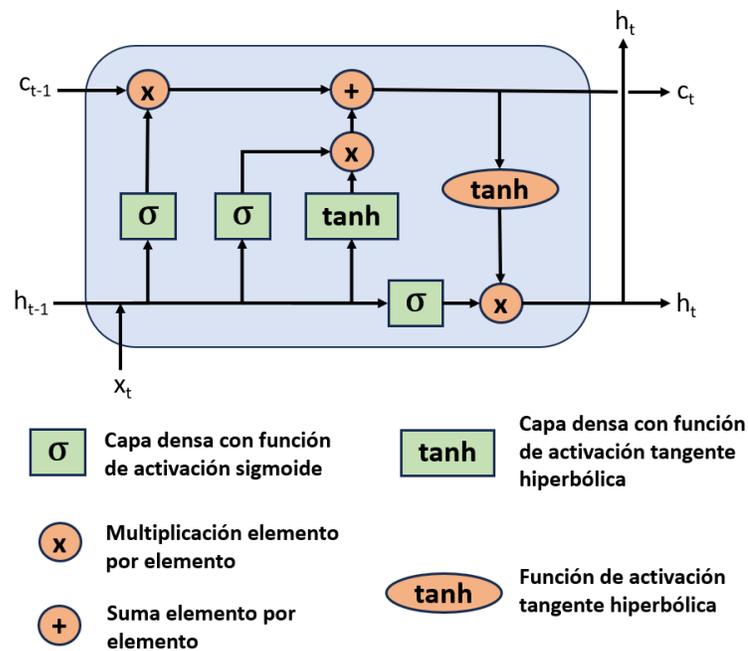


Figura 9. Unidad básica de una red LSTM

Capítulo 3. Trabajo relacionado

En los últimos años se ha visto la adopción de técnicas de aprendizaje de máquina para la localización de robots en interiores, y lo efectivas que estas han sido para mejorar la precisión de la localización (Roy & Chowdhury, 2021).

A continuación, se describen algunos de los trabajos más relevantes en donde se han aplicado técnicas de aprendizaje de máquina de manera exitosa para la localización de robots móviles en interiores, divididos en dos secciones diferentes: localización basada en Wi-Fi, y localización a partir de datos inerciales.

3.1. Localización basada en Wi-Fi

Los métodos de localización basados en Wi-Fi han generado interés en los últimos años, debido a que los puntos de acceso Wi-Fi se pueden encontrar de forma ubicua en la mayoría de los ambientes urbanos (Hashemifar et al., 2019).

Los métodos basados en Wi-Fi para localización pueden dividirse en grupos: basados en rango y no basados en rango. A su vez, los basados en rango pueden separarse en cuatro clases, según los parámetros que utilizan: tiempo de vuelo, diferencia en el tiempo de llegada, ángulo de llegada e indicador de intensidad de señal recibida (RSSI). El RSSI es una medida de la potencia de una señal de radio recibida por un dispositivo, normalmente expresada en dBm (decibel-miliwatt), donde un valor más cercano a cero indica una mayor intensidad de señal (ver tabla 1). Para el caso de métodos no basados en rango existen dos categorías amplias: métodos de *fingerprinting* y algoritmos basados en el conteo de saltos de comunicación (Cui et al., 2020).

Tabla 1. Calidad de la señal en función del rango de valores RSSI. Tomado de Azini et al. (2016).

Rango RSSI (dBm)	Condición de la señal
≥ -60	Excelente
-61 a -75	Buena
-76 a -80	Media-baja
-81 a -89	Mala
≤ -90	Muy mala

Las señales Wi-Fi además han sido empleadas para aumentar las capacidades de algoritmos de SLAM visuales tradicionales. Hashemifar et al. (2019) logran esto utilizando la intensidad de las señales Wi-Fi recibidas como un indicador de localidad espacial aproximada, sin ningún requisito de una fase previa de recopilación de datos Wi-Fi.

Cui et al. (2020) proponen en su trabajo un método de *fingerprinting* en el cual, reconociendo que la fase de calibración suele ser muy laboriosa y que las mediciones de RSSI suelen ser fácilmente influenciadas por condiciones ambientales, diseñan un sistema basado en la arquitectura de red neuronal conocida como máquina de aprendizaje extremo (ELM). ELM es un algoritmo propuesto para resolver redes neuronales *feed forward* de una sola capa oculta (SLFNs). ELM no requiere una retropropagación basada en gradientes para funcionar. Utiliza la inversa generalizada de Moore-Penrose para establecer sus pesos. En la mayoría de los casos, los pesos de los nodos ocultos son memorizados usualmente en un solo paso, lo que resulta en una forma muy rápida de aprendizaje. Para este trabajo, ELM se combina con análisis de componentes principales robusto (RPCA), al que los autores llaman RPCA-ELM. El modelo basado en RPCA-ELM se entrena durante la fase de calibración fuera de línea recolectando datos RSSI de los puntos de acceso (APs) Wi-Fi en el ambiente en diferentes localizaciones de referencia. Este modelo entrenado es usado posteriormente para la fase de localización en línea.

Zhang et al. (2020) abordan el problema de manera similar, basándose en el mismo método de *fingerprinting*, solo que en este caso, el modelo utilizado para la fase de calibración fuera de línea emplea un bosque difuso profundo (*deep fuzzy forest*) (figura 10). Este está formado de K árboles de decisión difusos (*fuzzy decision tree*) (Yuan & Shaw, 1995). La ubicación final de cada muestra se obtiene mediante un promedio ponderado de los resultados con su correspondiente probabilidad de cada nodo hoja.

Siguiendo la misma línea de trabajo, Zou et al. (2020) proponen un método al que llaman *WiGAN*. Este método basado en regresión de procesos Gaussianos (GPR) (Seeger, 2004) y redes generativas adversarias (GANs), se encarga de la construcción del mapa de radio, es decir, la base de datos de valores RSSI del ambiente, sintetizando los valores en las localizaciones del ambiente no navegables, i.e., a las que el robot no puede llegar.

En lugar de valores RSSI, Ayyalasomayajula et al. (2020) miden el tiempo de vuelo y ángulo de llegada de las señales de los puntos de acceso. Proponen *MapFind*, una plataforma robótica que recorre el ambiente de forma autónoma y construye el mapa del mismo con un algoritmo SLAM tradicional, al mismo tiempo que recolecta las mediciones de tiempo de vuelo y ángulo de llegada de las señales de los puntos de acceso Wi-Fi cercanos. Estos datos se codifican en mapas de calor 2D, los cuales son utilizados para entrenar una red neuronal convolucional (CNN). Este último modelo, al cual los autores llaman *DLoc*, puede ser utilizado por robots móviles u otros dispositivos para localizarse en el ambiente (figura 11).

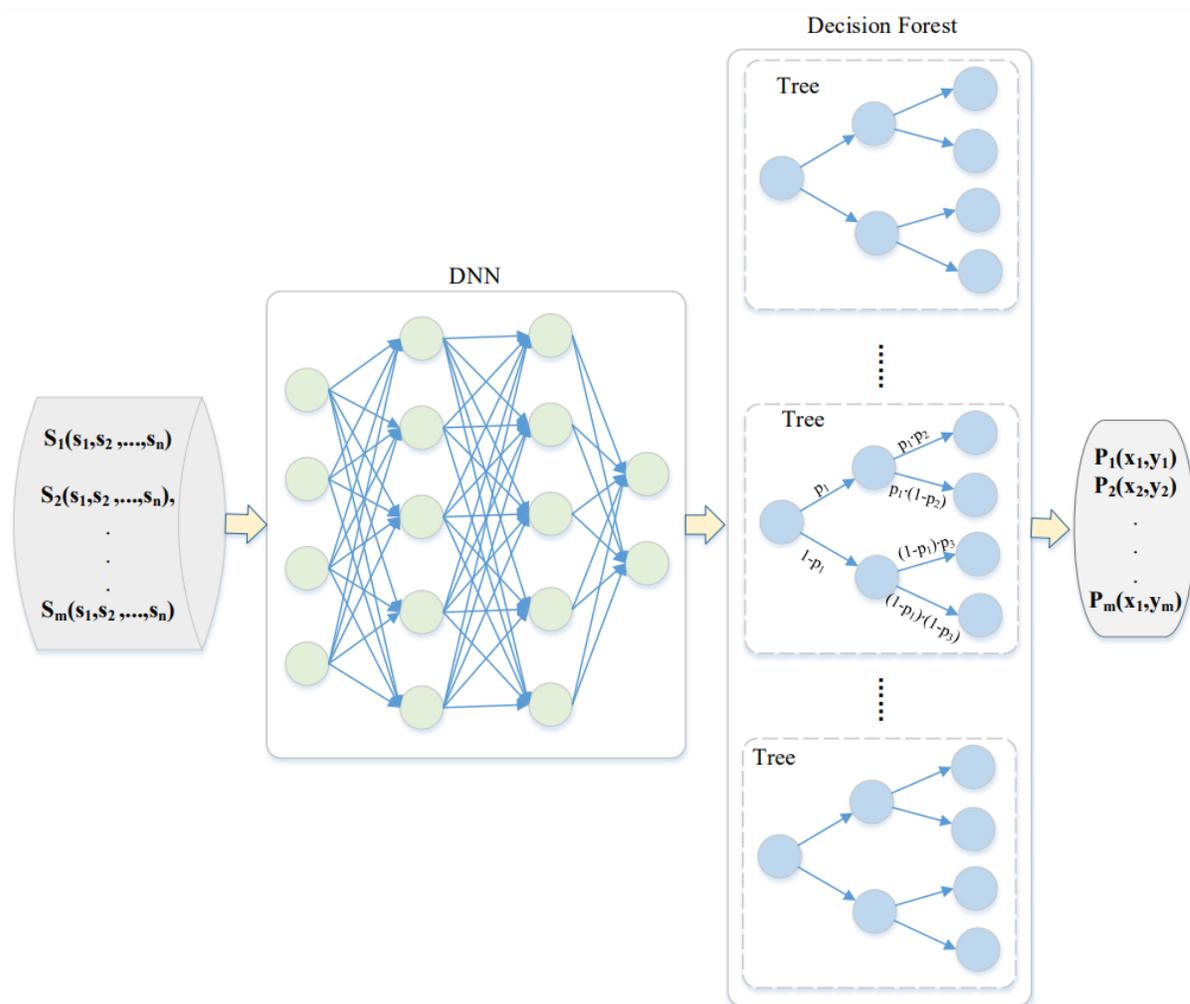


Figura 10. Descripción general del bosque difuso profundo. Una red neuronal extrae las características profundas de las firmas Wi-Fi de entrada. Un bosque decisión, formado por múltiples árboles de decisión difusos, toman las características extraídas y estiman la posición. La localización final es obtenida realizando un promedio ponderado de las estimaciones de todos los árboles. Tomado de Zhang et al. (2020).

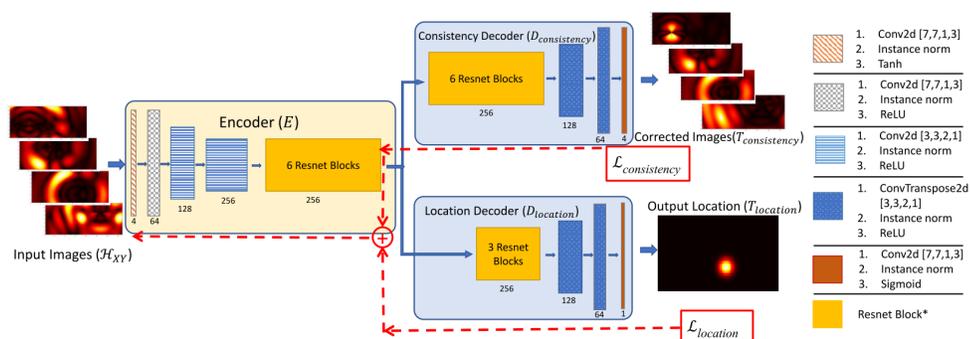


Figura 11. Arquitectura de DLoc. Se toman los mapas de radio 2D como entrada, siendo procesadas por la sección *encoder* de la red, cuya salida pasa a su vez a dos secciones *decoder* paralelas. El primer *decoder* corrige las imágenes de entrada para asegurar que la red tenga una vista uniforme del entorno para diferentes ejemplos de entrenamiento, así como para diferentes puntos de acceso. El segundo *decoder* entrega la estimación de localización. Tomado de Ayyalasomayajula et al. (2020).

Dado que los valores RSSI de una señal Wi-Fi suelen contener ruido debido a los efectos de interferencia entre múltiples señales o reflexión de estas mismas con obstáculos en el ambiente, Wang & Park (2021) proponen un método de *fingerprinting* que no solo mide el RSSI de las señales Wi-Fi durante la fase de calibración, si no que además incorpora información del estado de la señal, llamado CSI (*Channel State Information*). En este caso, del CSI se extraen dos valores, la amplitud y la fase de la señal, los cuales tienen a ser más robustos a las condiciones ambientales que el RSSI. Estos valores son entonces preprocesados junto con el RSSI utilizando filtros Gaussianos y de Kalman, para después formar la base de datos. Se utiliza entonces K vecinos más cercanos para estimar la posición durante la fase de localización en línea (figura 12).

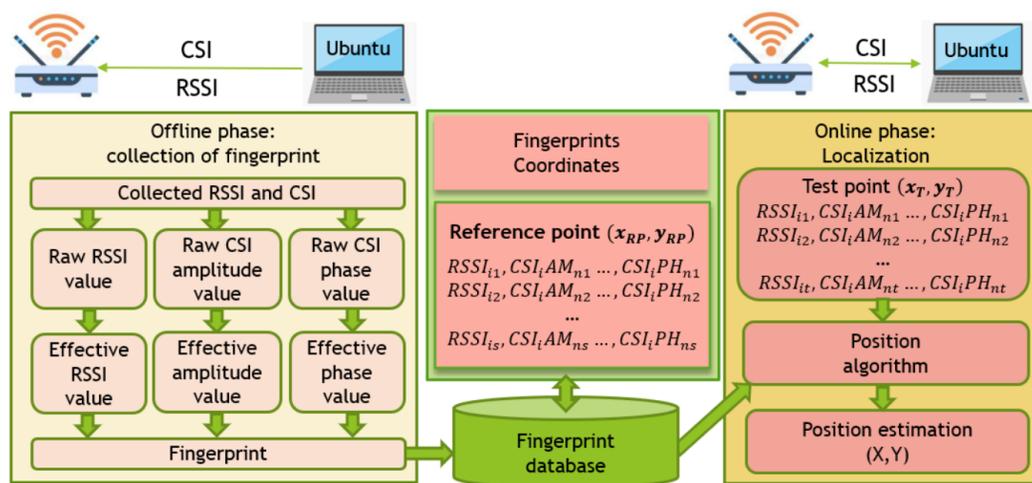


Figura 12. Arquitectura del sistema de localización basado en *fingerprints* utilizando CSI. Cada firma para la base de datos se forma tomando el valor RSSI, la amplitud y la fase de las señales Wi-Fi. Esta base se utiliza para entrenar un modelo KNN que estima entonces la posición. Tomado de Wang & Park (2021).

Otro de los métodos empleados para realizar la localización es el de multilateración. En el caso de las señales Wi-Fi, se utilizan los valores RSSI para obtener una distancia aproximada de los nodos al punto a localizar. Con estas distancias, y conociendo la posición de los nodos, se puede estimar la posición del robot en el ambiente. Sin embargo, debido a que las distancias utilizadas cuentan con un nivel considerable de error, la posición estimada con este método cuenta también con un nivel de error alto. Luo et al. (2022), con la idea de reducir este error, implementan un algoritmo en el que, en lugar de estimar una sola posición utilizando todos los nodos de referencia a la vez, generan múltiples combinaciones de grupos de tres nodos, y estiman una posición con cada uno de estos grupos. Después, emplean el método de *clustering* K-medias, mediante el cual eliminan las posiciones estimadas con mayor nivel de error (figura 13). Las posiciones restantes son utilizadas entonces para calcular la estimación final.

Entrada:
 $D = \{d_1, d_2, \dots, d_n\}$ // Conjunto de elementos
 k // Numero de clusters

Salida:
 Un conjunto de k clusters

Algoritmo K-medias:

- 1 Escoger aleatoriamente k elementos de D como centroides iniciales
- 2 **repetir**
- 3 asignar cada elemento d_i al cluster que tenga el centroide más cercano
- 4 calcular la nueva media o centroide de cada cluster
- 5 **hasta** converger

Figura 13. Pseudocódigo para el algoritmo de *clustering* utilizando K-medias.

Los métodos de localización Wi-Fi también han sido utilizados para aumentar o mejorar técnicas de localización tradicionales. Arun et al. (2022) implementan un método en el que emplean las mediciones de un sensor Wi-Fi como entrada a un método SLAM, buscando superar las limitaciones de los sensores visuales usados normalmente en algoritmos SLAM tradicionales. En particular, este método utiliza las posiciones de los puntos de acceso Wi-Fi en el ambiente como puntos de referencia, sin tener conocimiento previo de la localización de estos, usando las mediciones de ángulo de llegada de las señales W-Fi para estimar sus posiciones.

Con la idea de mejorar el método tradicional de *fingerprinting*, Wang et al. (2023a) introducen lo que ellos llaman firmas temporales, las cuales, a diferencia de las firmas normalmente utilizadas, se construyen por medio de una ventana de tiempo deslizante, e incluyen la marca de tiempo en el que cada firma fue tomada, además de los valores RSSI (figura 14). Las firmas son preprocesadas para que los valores de intensidad se encuentren en el rango de 0 a 1, y se utilizan para entrenar un modelo basado en una red convolucional temporal (TCN)

3.2. Localización basada en datos inerciales

Avances recientes en los sistemas microelectromecánicos (MEMs) han permitido el desarrollo de unidades de medición inercial (IMUs) lo suficientemente pequeñas y económicas para ser implementadas en robots y otros dispositivos móviles. Sin embargo, estos sensores de bajo costo cuentan con altos niveles de ruido que se propagan durante la navegación, lo que lleva a un rápido crecimiento de los errores durante la estimación de la localización, conocido como error de deriva (Chen et al., 2018).

Para abordar el problema anterior, diversas técnicas de aprendizaje de máquina han sido utilizadas para intentar corregir los errores que resultan de emplear métodos de localización basados en IMUs.

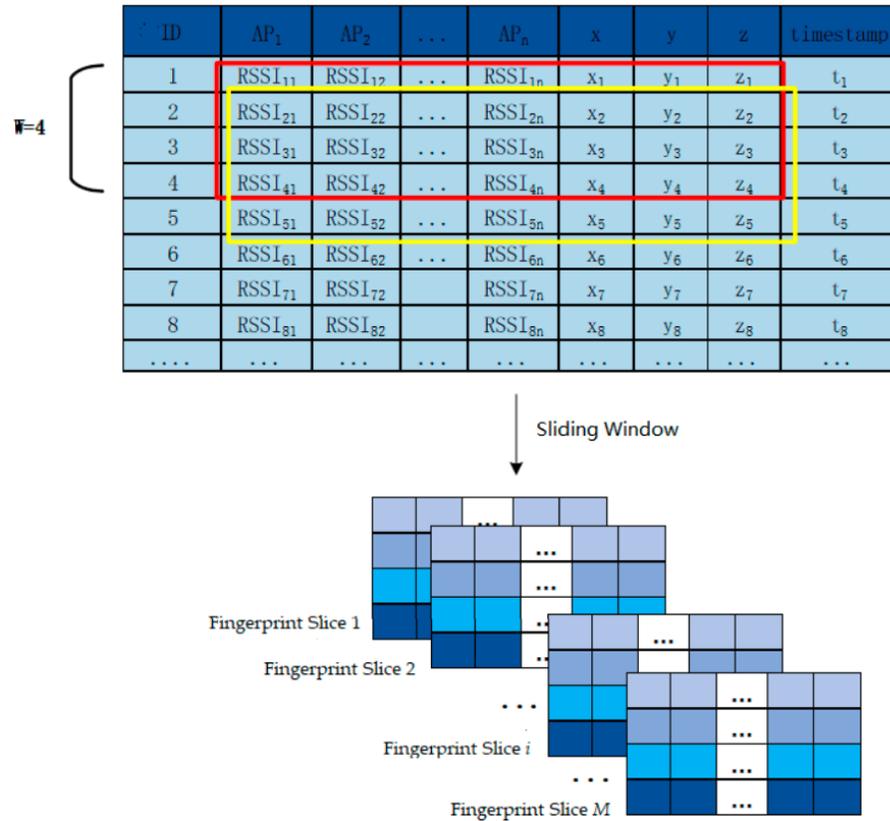


Figura 14. Proceso de construcción de las firmas temporales. Se utiliza una ventana de tamaño $W = 4$, que se desliza una unidad a la vez a través de la base de datos de firmas, generando una nueva base de datos con estas ventanas que contienen información temporal. Tomado de Wang et al. (2023a).

Chen et al. (2018) presentan un método llamado *Inertial Odometry Neural Network* (IONet) para restringir la inevitable deriva de los sistemas inerciales. Este método implementa una red neuronal profunda (DNN) formada por capas *long short-term memory* (LSTM), la cual se entrena utilizando mediciones inerciales de un IMU divididas en ventanas de tiempo independientes. El modelo recibe como entrada una de estas ventanas y entrega como resultado el cambio en la posición y orientación del dispositivo durante este tiempo (figura 15).

Lima et al. (2019) presentan un modelo basado en una CNN, combinada con dos capas LSTM bidireccionales, el cual recibe como entrada los datos provenientes de un IMU de seis grados de libertad, es decir, tres valores de aceleración lineal y tres valores de velocidad angular (figura 16). Los datos se dividen en ventanas de 200 mediciones cada una, y el modelo devuelve el cambio en la posición y orientación entre dos ventanas consecutivas. Una vez que se tienen las posiciones relativas para todas las ventanas de tiempo, se puede reconstruir la trayectoria del agente en el espacio 3D.

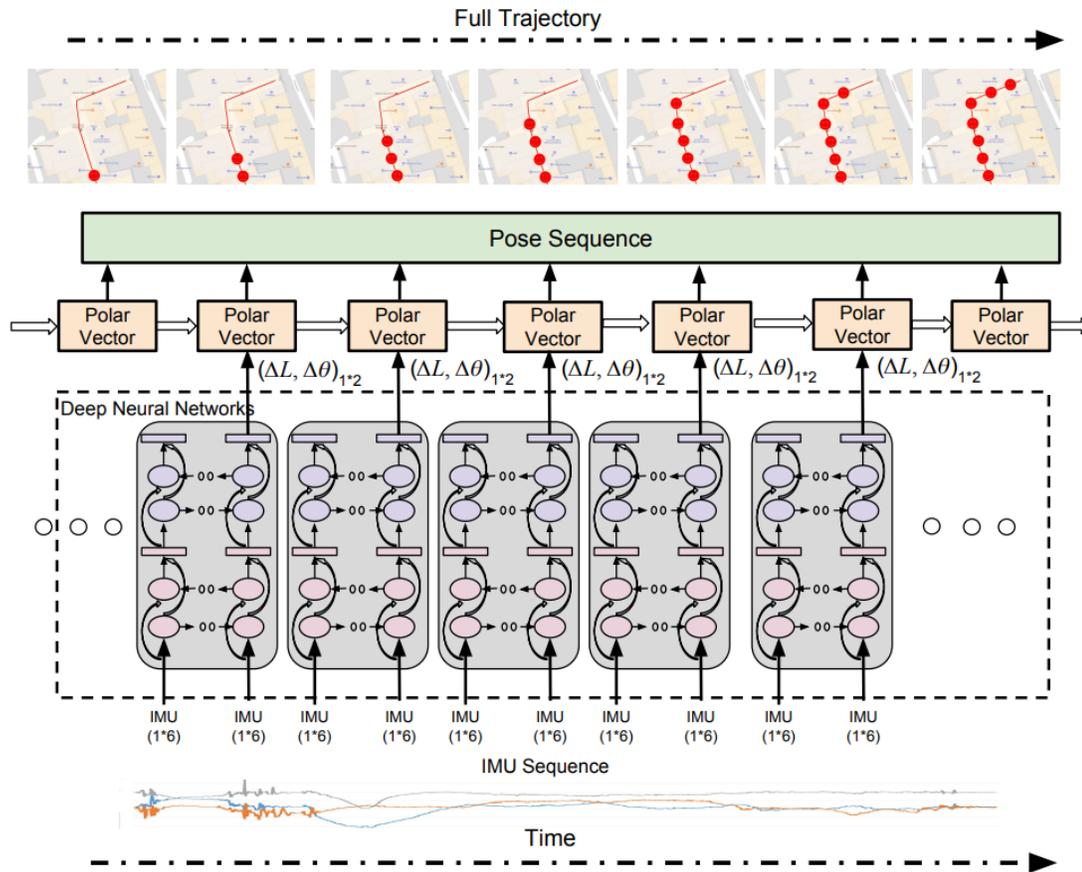


Figura 15. Arquitectura de IONet. Una red neuronal basada en dos capas LSTM recibe como entrada ventanas de datos provenientes de un IMU, para cada una de las cuales entrega un vector polar. Vectores polares provenientes de ventanas de datos consecutivas permiten reconstruir la trayectoria estimada. Tomado de Chen et al. (2018).

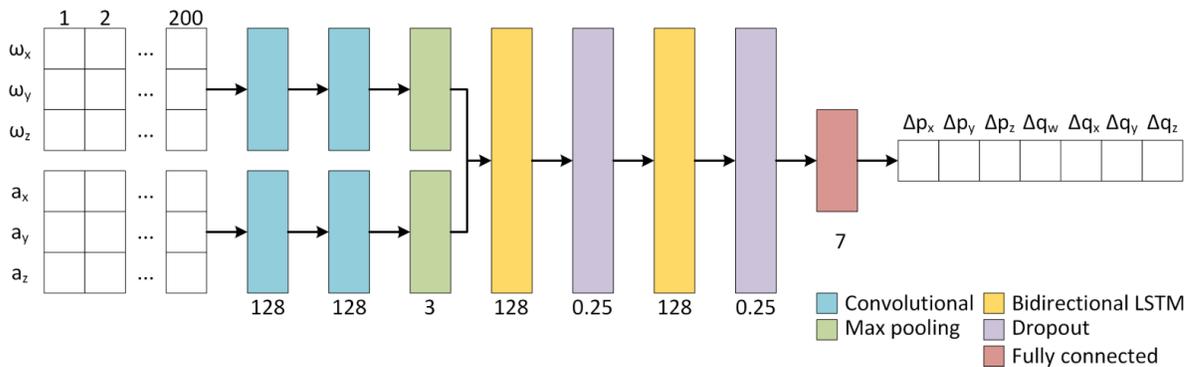


Figura 16. Red neuronal que combina capas CNN y LSTM. La red recibe como entrada las aceleraciones lineales y velocidades angulares por separado, y genera como salida los cambios relativos en posición y orientación, este ultimo representado por un cuaternión. Tomado de Lima et al. (2019).

Para resolver el mismo problema de error acumulado en el uso de sensores inerciales, Abolfazli Esfahani et al. (2020) presentan una arquitectura de red profunda *Long Short-Term Memory* (LSTM) de triple

canal basada en los modelos físicos y matemáticos de las IMU, al cual llaman *AbolDeepIO* (figura 17). El modelo recibe como entrada las medidas de aceleración, velocidad angular, y el intervalo de tiempo entre dos medidas consecutivas. Además, el método propuesto simula el modelo de ruido en la fase de entrenamiento y se vuelve robusto al ruido durante la fase de pruebas.

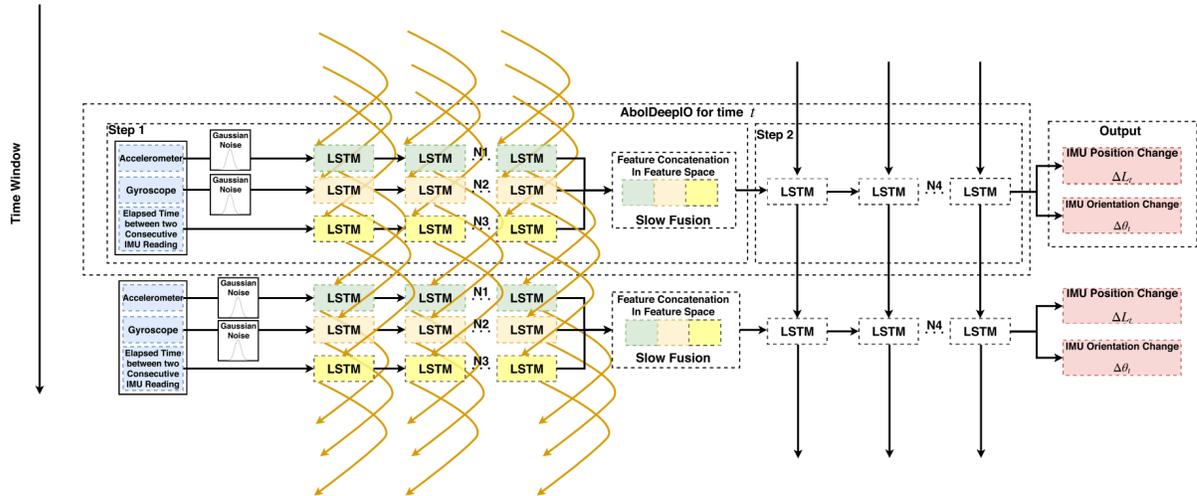


Figura 17. Arquitectura de AbolDeepIO. Una red LSTM de triple canal aprende la representación de características de las medidas de acelerómetro, giroscopio y el intervalo de tiempo entre mediciones. Estas características son concatenadas y procesadas por un segundo número de capas LSTM, las cuales entregan como salida el cambio en orientación y posición. Tomado de Abolfazli Esfahani et al. (2020).

Herath et al. (2020) presentan un método llamado RoNIN (*Robust Neural Inertial Navigation*) para reconstruir la trayectoria de usuarios a través de datos inerciales de sus teléfonos móviles, en el que prueban tres arquitecturas diferentes de redes neuronales (ResNet, LSTM y TCN) como base para la localización (figura 18). De estas tres arquitecturas, LSTM y TCN tuvieron un desempeño ligeramente mejor que ResNet en estimar la posición del usuario, pero toman de 3 a 4 veces más de tiempo de entrenamiento.

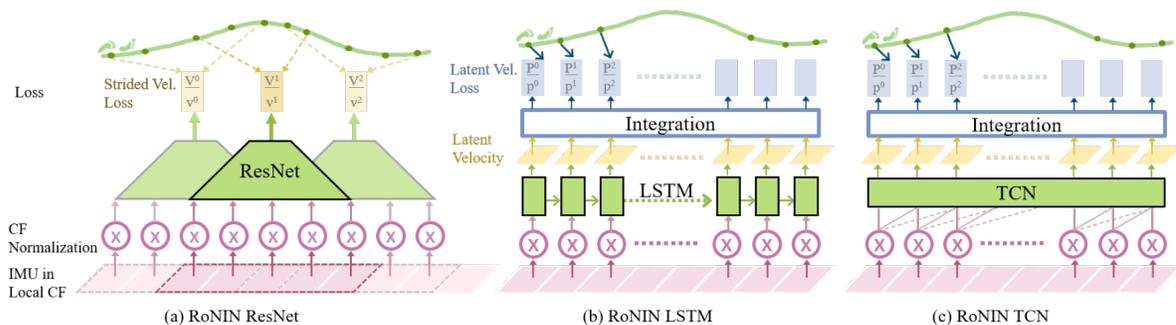


Figura 18. Las tres diferentes arquitecturas de RoNIN, (a) basada en ResNet-18, (b) basada en tres capas LSTM, y (c) basada en seis bloques residuales TCN. Tomado de Herath et al. (2020).

Liu et al. (2020) proponen un método al que llaman TLIO (*Tight Learned Inertial Odometry*). Este está compuesto por una red neuronal convolucional, específicamente la arquitectura ResNet18 propuesta por He et al. (2016), que se encarga de calcular el desplazamiento relativo 3D y la covarianza entre dos instantes de tiempo dado el segmento de datos del IMU entre ellos. La salida de este modelo se envía entonces al segundo componente, un filtro de Kalman extendido (EKF) que estima el estado actual del agente: posición 3D, velocidad, orientación y sesgos del IMU, además de un conjunto de poses pasadas (figura 19).

Asraf et al. (2022) presentan un método al que llaman PDRNet, el cual utiliza los datos del acelerómetro y giroscopio de un teléfono móvil para determinar la posición del usuario. Este método se divide en dos pasos: determinar la posición del teléfono móvil en el usuario (en las manos o en el bolsillo, por ejemplo) y después determinar la posición del usuario en el ambiente. Para el primer paso se implementa una red convolucional de una sola capa, mientras que para el segundo paso se utiliza la arquitectura ResNet-50, que predice el cambio en la posición y orientación del usuario (figura 20).

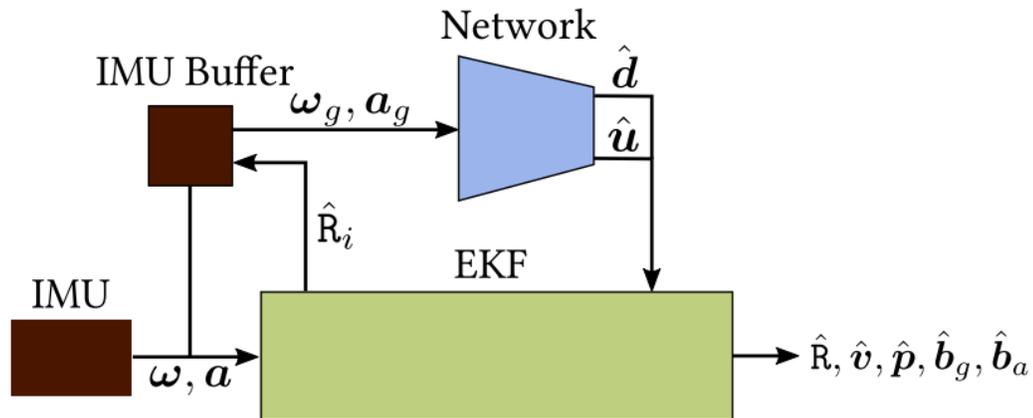


Figura 19. Diagrama de bloques del sistema TLIO. El bloque *IMU buffer* provee segmentos de mediciones del IMU que la red neuronal recibe como entrada, a partir de las cuales entrega como salida el desplazamiento \hat{d} y la incertidumbre \hat{u} , y estas son utilizadas por el EKF para estimar la rotación, velocidad, posición y los sesgos de las mediciones. Tomado de Liu et al. (2020).

Zhu et al. (2023), inspirándose en el trabajo de Herath et al. (2020), crean un nuevo modelo al que llaman RBCN-Net para inferir la trayectoria seguida por un usuario. Esto lo hacen tomando la arquitectura ResNet18 como base, y combinándola con capas LSTM bidireccionales, además de módulos de atención de bloque convolucional (CBAM) ((Woo et al., 2018)) y módulos de atención basados en normalización (NAM) ((Liu et al., 2021)) (figura 21). Los módulos de atención agregados ayudan a la red a distinguir los datos más importantes de la entrada, mientras que las capas LSTM bidireccionales le permite aprender las dependencias temporales entre los datos.

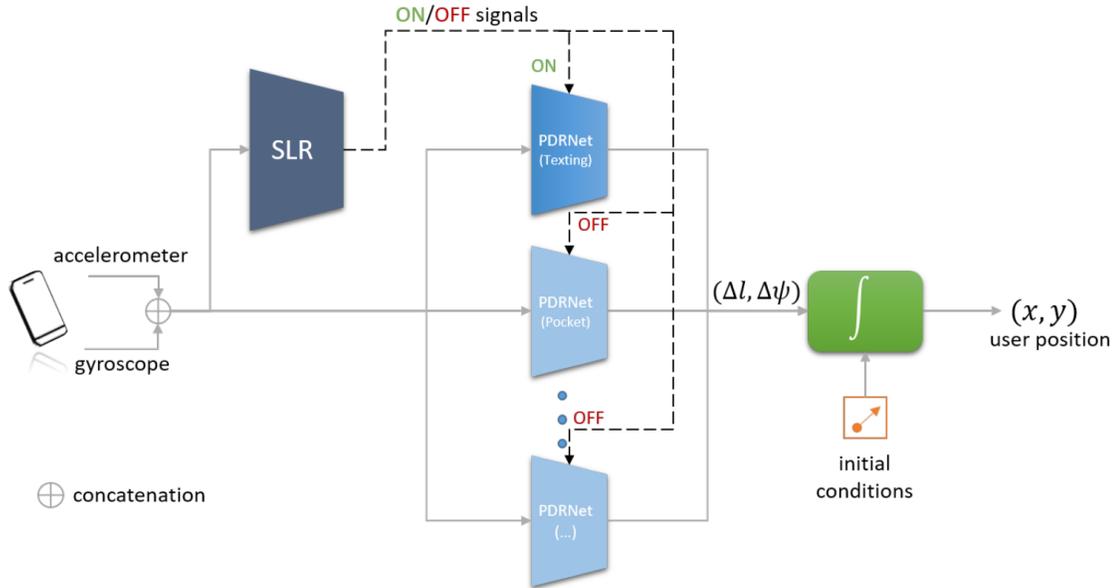


Figura 20. Diagrama de bloques del sistema PDRNet. La red convolucional SLR (Smartphone Location Recognition) estima la posición del teléfono móvil en el usuario utilizando sus mediciones inerciales, y con base en la posición estimada, activa una de las tres redes correspondientes. La red activada predice entonces el cambio en la posición y orientación del usuario. Tomado de Asraf et al. (2022).

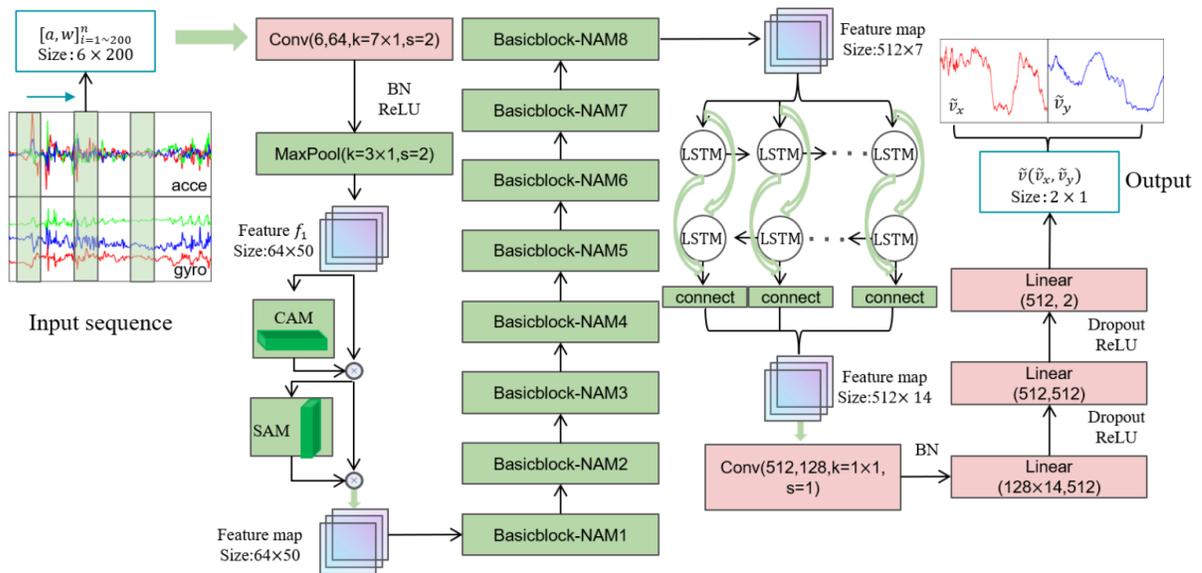


Figura 21. Arquitectura de RBCN-Net. Tomando como base la arquitectura de ResNet-18, se añade un módulo de atención CBAM a la entrada y módulos de atención NAM a cada bloque residual. A la salida se cuenta con dos capas LSTM bidireccionales para aprender las relaciones temporales entre los datos. Al final, la red neuronal entrega como salida el vector de velocidad correspondiente a la ventana de datos inerciales de la entrada. Tomado de Zhu et al. (2023).

Wang et al. (2023b) expanden en el trabajo de Liu et al. (2020), argumentando que el uso de TLIO es inadecuado para dispositivos móviles, debido a su alto costo computacional. Proponen por esto un

sistema más ligero llamado LLIO (*Lightweight Learned Inertial Odometer*), en el cual reemplazan la red ResNet18 original de TLIO por una red perceptrón multicapa (MLP). Prueban entonces que este nuevo sistema cuenta con un desempeño similar a TLIO, mientras que su tiempo de procesamiento es de 1.9 a 12 veces más rápido en dispositivos móviles.

Capítulo 4. Localización basada en WiFi

En este capítulo se describe la metodología empleada para determinar la efectividad de las señales Wi-Fi como base para resolver el problema de localización en interiores, los experimentos realizados y los resultados obtenidos.

4.1. Metodología

El desarrollo de esta parte del trabajo se dividió en dos fases principales: implementación de un modelo para simular las señales Wi-Fi, e implementación y evaluación de distintos métodos de localización utilizando los datos recolectados.

4.1.1. Simulación de señales Wi-Fi

De entre los distintos datos y métricas extraíbles de las señales Wi-Fi, se seleccionó la intensidad de la señal como los datos a utilizar para la localización, debido a que otros tipos de datos, como el tiempo de vuelo o ángulo de llegada de la señal, requieren de equipo de red especializado y son mucho más difíciles de modelar y simular. Existen gran variedad de modelos para simular la intensidad de una señal Wi-Fi, o una señal de radio en general. Se seleccionó el modelo llamado *Log Distance Path Loss* o *Log Normal Shadowing* (LND) (Etter-Olguín et al., 2019), descrito a continuación:

$$PL = PL_0 - 10n \log_{10}\left(\frac{d}{d_0}\right) + X_g \quad (5)$$

donde PL es la intensidad de la señal a una distancia d , PL_0 es una intensidad de referencia a una distancia d_0 , n es un parámetro llamado *path loss exponent*, y X_g es una variable aleatoria con distribución gaussiana.

Este modelo fue elegido debido a que es el más utilizado en la literatura revisada, a pesar de ser de los más sencillos. El que sea un modelo muy sencillo, sin embargo, significa que no modela fenómenos que afectan la intensidad de la señal Wi-Fi en un ambiente real, por ejemplo, las reflexiones de la señal con los objetos. En su lugar, este modelo pretende capturar el efecto de estos fenómenos por medio del parámetro n . Se han identificado en la literatura posibles valores para este parámetro de acuerdo al tipo de ambiente que se desea simular, de forma que sea lo más parecido a la realidad. Estos valores obtenidos por medio de datos experimentales se puede observar en la tabla 2.

Tabla 2. Posibles valores del parámetro *path loss exponent* de acuerdo al tipo de ambiente a simular (El Khaled et al., 2022).

Ambiente	<i>Path loss exponent</i> (n)
Espacio libre	2
Radio celular de área urbana	2.7 a 3.5
Radio celular urbana obstruida	3 a 5
Dentro de un edificio, línea de visión	1.6 a 1.8
Obstruida en edificio	4 a 6
Obstruida en una fábrica	2 a 3

Para este trabajo, se consideró un valor $d_0 = 1$ para facilitar los cálculos. A esta distancia de un metro, se observó que la intensidad promedio de las señales Wi-Fi emitidas por los puntos de acceso disponibles para experimentación era de -30 dBm. Por ello, se consideró para el modelo un valor $PL_0 = -30$.

Para elegir el valor de n , se realizaron una serie de recorridos en el Departamento de Ciencias de la Computación de CICESE utilizando la plataforma robótica *TurtleBot3*, durante los cuales se registraron los puntos de acceso Wi-Fi detectados y la intensidad de la señal de cada uno durante todo el recorrido. Ejemplo de las intensidades de las señales durante un recorrido se pueden observar en la figura 22

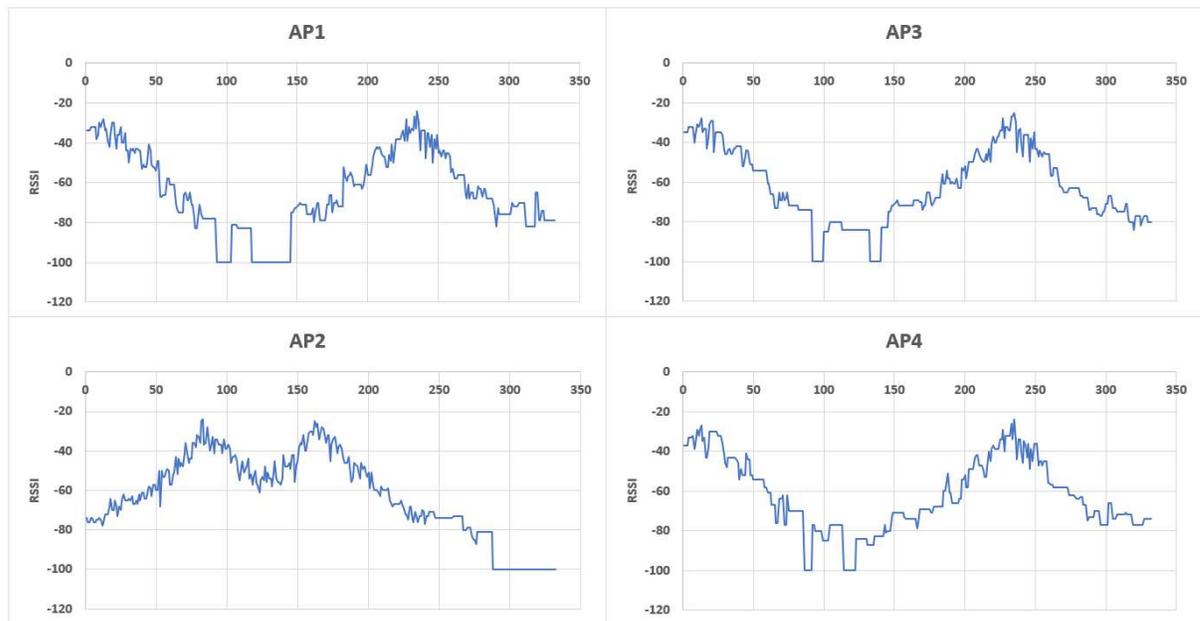


Figura 22. Ejemplo de nivel de intensidad de señales Wi-Fi capturadas de cuatro puntos de acceso diferentes durante el mismo recorrido. Un nivel RSSI de -100 dBm significa que la señal Wi-Fi dejó de ser detectada.

Observando las gráficas de la figura 22, se puede notar que el valor RSSI mínimo antes de que la señal Wi-Fi deje de ser detectada es de alrededor de -80 dBm. Suponiendo que el promedio de la distancia

máxima de detección de una señal Wi-Fi en un espacio interior es de 45 metros, y considerando $d_0 = 1$ y $PL_0 = -30$, se tomó el valor de n de tal forma que a 45 metros el modelo tenga como salida una intensidad de -80 dBm, resultando en un valor de $n = 3$.

Con base en los mismos datos de la figura 22, seleccionó un ruido $X_g \sim \mathcal{N}(0, 3)$, es decir, un ruido con distribución normal con media de 0 y desviación estándar de 3, de forma que el valor calculado por el modelo se aproxime lo más posible a la realidad.

El modelo completo con los valores seleccionados queda entonces de la siguiente manera:

$$PL = -30 - 10 * 3 \log_{10}(d) + \mathcal{N}(0, 3) \quad (6)$$

En la figura 23 se puede observar una comparación entre la intensidad de un punto de acceso Wi-Fi registrado durante uno de los recorridos realizados en el Departamento de Ciencias de la Computación de CICESE y la intensidad calculada por el modelo Wi-Fi implementado para las mismas distancias.

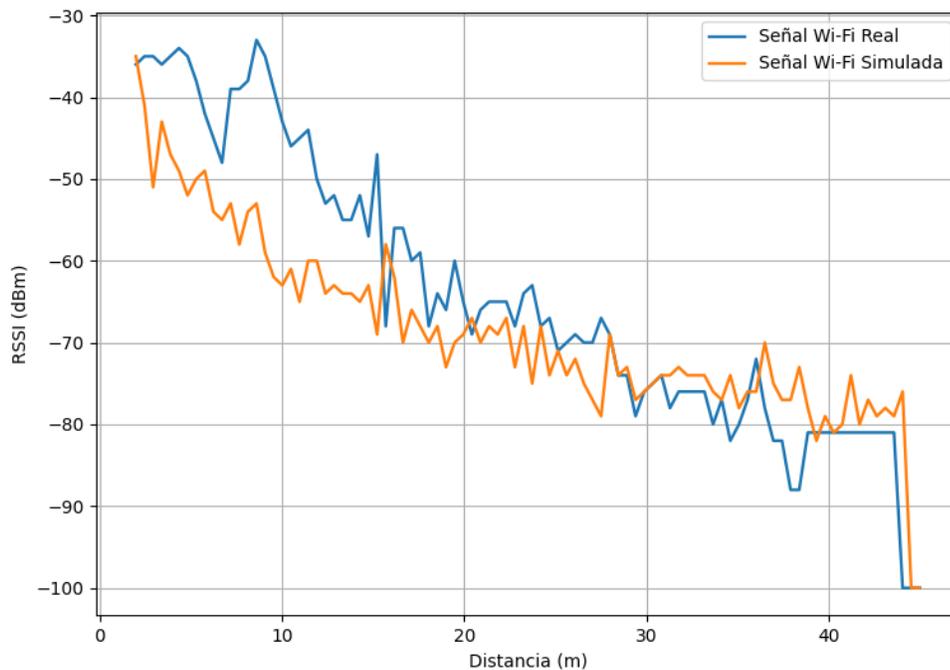


Figura 23. Comparación entre la intensidad de la señal Wi-Fi de un punto de acceso real y la intensidad calculada por el modelo Wi-Fi implementado.

Una vez hecho esto, el siguiente paso fue llevar el modelo a la plataforma de simulación *Gazebo*. Para esto, se tomó como base el trabajo realizado por D'avella et al. (2022), en el cual los autores implementan

en *Gazebo* la simulación de etiquetas RFID, así como las antenas que detectan la intensidad de la señal de las mismas. Se modificó este trabajo para simular señales Wi-Fi en lugar de RFID, reemplazando el modelo original por el modelo descrito anteriormente. De esta forma, los objetos dentro de *Gazebo* que simulaban las etiquetas RFID simulan ahora los puntos de acceso Wi-Fi.

Para aproximar el modelo lo más posible a la realidad, se agregó un rango máximo de detección, de forma que si la distancia entre el robot y un punto de acceso es mayor al valor establecido, la intensidad reportada por el punto de acceso será de -100 dBm, significando que está fuera del rango de detección. Se seleccionó un rango máximo de detección de 45 metros, teniendo también un ruido gaussiano con media de 0 y desviación estándar de 1. Además, el valor RSSI entregado por el modelo para una distancia dada es redondeado al número entero más cercano, dado que en la vida real este valor se reporta únicamente con números enteros. Se puede observar el RSSI calculado por el modelo para distancias de 1 a 50 metros en la figura 24.

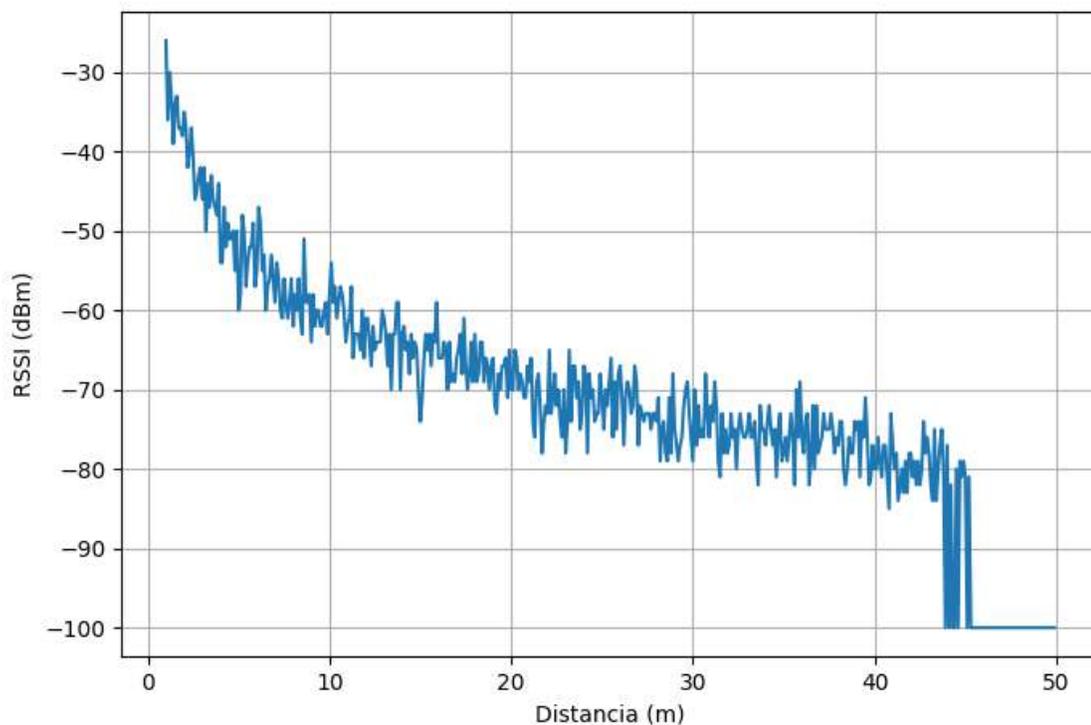


Figura 24. Salida del modelo Wi-Fi para distancias de 1 a 50 metros

Con el simulador listo para generar datos de señales Wi-Fi a la vez que el robot realiza su recorrido a través del ambiente, se prosiguió entonces a seleccionar tres métodos diferentes de localización para probar y comparar: método de *fingerprinting*, multilateración y multilateración utilizando agrupamiento de K-medias.

4.1.2. Fingerprinting

Los detalles de este método de localización se describen en la sección 2.1. Se tomó como base un ambiente de 50 por 50 metros para realizar los experimentos, en el cual se colocaron 50 puntos de acceso Wi-Fi (APs) aleatoriamente distribuidos de forma uniforme de tal manera que cubran todo el ambiente, como se muestra en la figura 25.

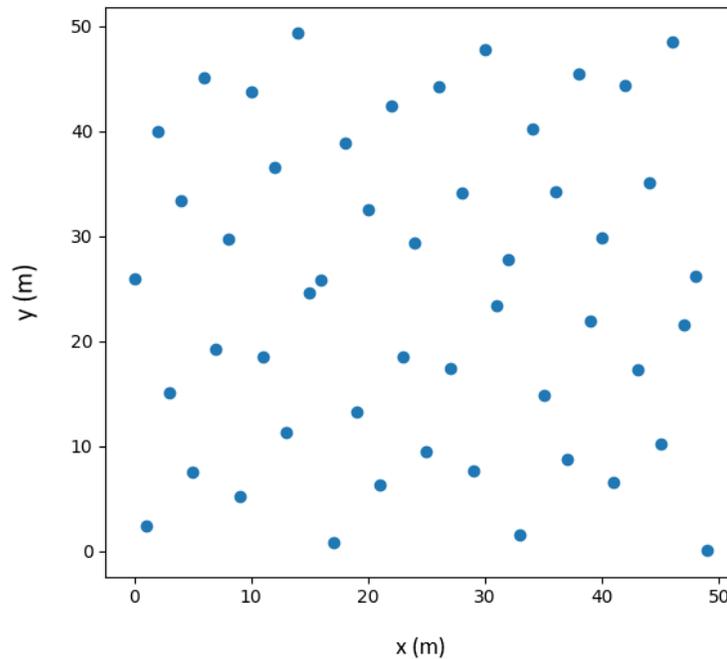


Figura 25. Puntos de acceso Wi-Fi distribuidos en el ambiente

Para la fase de calibración fuera de línea se crearon cuatro bases de datos diferentes, tomando puntos de referencia cada 0.5, 1, 1.5 y 2 metros, formando cuadrículas de 10,000, 2,500, 1,156 y 625 puntos respectivamente, que cubren todo el ambiente. Esto se hizo con la intención de observar el efecto que tiene la cantidad y densidad de puntos de referencia en el error de localización. En cada punto entonces se capturó el valor RSSI de todos los APs, generando un vector con la forma $[RSSI_0, RSSI_1, \dots, RSSI_n]$, donde $RSSI_i$ es la intensidad del AP_i . Este vector es lo que funcionará eventualmente como entrada al modelo de localización.

Para cada base de datos, se hicieron experimentos utilizando 5, 10, 15, 20, 25, 30, 35, 40, 45 y 50 APs para realizar la localización, con el objetivo de observar el efecto que la cantidad de APs en el ambiente tiene en el error de localización. Se utilizaron tres modelos de aprendizaje de máquina diferentes para realizar la localización: SVM, KNN y una red neuronal (NN), la arquitectura de la cual se muestra en la

figura 27.

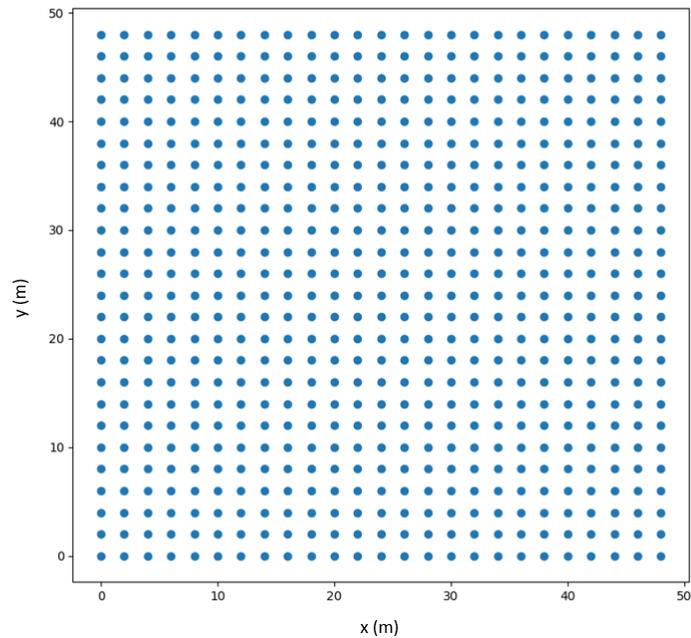


Figura 26. Cuadrícula de puntos de referencia para *fingerprinting*, tomados cada 2 metros

Cada una de las red neuronales empleadas para cada número de APs se entrenó durante 300 épocas, es decir, 300 pasadas completas del conjunto de datos de entrenamiento a través del algoritmo de aprendizaje, utilizando como función de pérdida el error absoluto medio (MAE por sus siglas en inglés), una tasa de aprendizaje de 0.0001, el optimizador Adam, siendo este el optimizador estándar, y un *batch size* de 32, un tamaño pequeño que ayuda a la convergencia durante el entrenamiento, aunque aumenta el tiempo del mismo. Para los modelos KNN se utilizó un valor de K de 5, y para los modelos SVM se seleccionó como kernel la función de base radial (RBF por sus siglas en inglés), siendo este el kernel predeterminado debido a su facilidad de uso y buenos resultados en la práctica.

Para probar los modelos entrenados, se seleccionaron 500 posiciones aleatorias en el ambiente, diferentes a las posiciones de referencia, y se generó la firma de RSSI para cada punto, las cuales funcionan como la entrada a los modelos. Se calcula entonces el error entre la posición real del punto para cada firma y la posición estimada por los modelos.

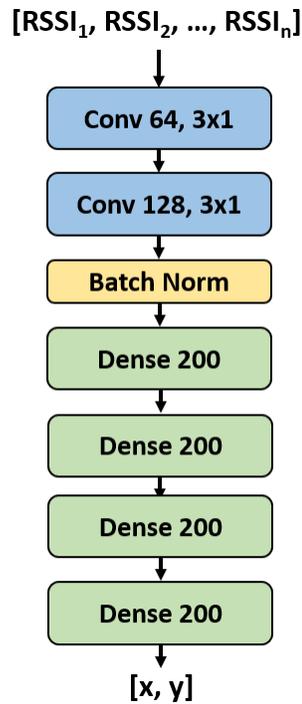


Figura 27. Arquitectura del modelo empleado para el método *fingerprinting*, que toma como entrada una firma de valores RSSI para los n APs en el ambiente. Las capas convoluciones al inicio se encargan de extraer las características profundas de la firma de entrada, mientras que las capas densas al final aprenden las relaciones entre estas características. La capa *batch normalization* ayuda a acelerar y estabilizar el entrenamiento de la red.

4.1.3. Multilateración

Este método de localización se describe en la sección 2.2. Se tomó como base un ambiente de 50 por 50 metros para realizar los experimentos. Para resolver el problema de multilateración, se necesita conocer la posición de los APs en el ambiente y la distancia de los mismo al punto a localizar. La posición de los APs se asume que es conocida, pero no se tiene acceso directamente a las distancias de estos al punto de interés, si no que debemos obtenerlas a partir de los valores RSSI capturados en el punto. Para esto tomamos el modelo descrito en la sección 4.1.1, despejando la distancia d para ponerla en función del valor RSSI,

$$d = d_0 10^{\frac{PL_0 - PL}{10n}} \quad (7)$$

Reemplazando los valores obtenidos en la sección 4.1.1 ($d_0 = 1$, $PL_0 = -30$, $n = 3$), se obtiene la siguiente ecuación:

$$d = 10^{\frac{-30-PL}{30}} \quad (8)$$

Para construir la base de datos, se generaron 10,000 escenarios diferentes, en los que se colocaron aleatoriamente 50 APs en el ambiente, con alturas que varían entre 0 y 5 metros, y un punto a localizar. Para cada uno de estos escenarios se forma un vector de la forma $[x_0, y_0, z_0, d_0, x_1, y_1, z_1, d_1, \dots, x_n, y_n, z_n, d_n]$, donde x_i, y_i, z_i es la posición del AP i en el ambiente, y d_i es la distancia del AP i al punto a localizar. Este vector es la entrada al modelo, mientras que el objetivo o la salida es la coordenada (x, y) del punto a localizar. Aunque el robot se desplaza únicamente en el plano XY, se emplea multilateración 3D dado que la altura a la que se encuentran los APs en el ambiente es variable, por lo que es necesario considerar su coordenada Z para la estimación de la localización del robot. Con esta base de datos de 10,000 muestras creada, se entrenaron 10 modelos de redes neuronales considerando 5, 10, 15, 20, 25, 30, 35, 40, 45 y 50 APs. La arquitectura de las redes neuronales entrenadas se muestran en la figura 28.

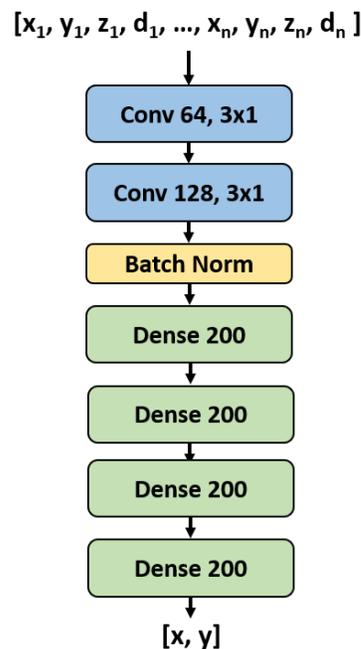


Figura 28. Arquitectura del modelo empleado para el método de multilateración, que toma como entrada la posición de cada AP y la distancia de estos al punto a localizar. Las capas convoluciones al inicio se encargan de extraer las características profundas de la firma de entrada, mientras que las capas densas al final aprenden las relaciones entre estas características. La capa *batch normalization* ayuda a acelerar y estabilizar el entrenamiento de la red.

Cada red neuronal empleada para cada número de APs se entrenó durante 300 épocas, utilizando como función de pérdida el error absoluto medio (MAE por sus siglas en inglés), una tasa de aprendizaje de 0.0001, el optimizador Adam y un *batch size* de 32.

Para probar los modelos entrenados, se generaron 1,000 escenarios aleatorios para cada número de APs que se emplearon para el entrenamiento. Para cada escenario, se calcula el error entre la posición real del punto a localizar y la estimada por el modelo. Esos resultados se comparan por los obtenidos utilizando el método de mínimos cuadrados descrito en la sección 2.2.

4.1.4. Multilateración utilizando agrupamiento de K-medias

En esta sección se adopta el trabajo realizado por Luo et al. (2022), en el que buscan mejorar la localización por multilateración utilizando un método de agrupamiento (clustering en inglés) de K-medias. Considerando que tenemos n APs en el ambiente, podemos formar N diferentes grupos de m APs ($m \leq n$). Se tomó $m = 4$ para este trabajo, dado que este es el número mínimo de nodos de referencia requeridos para estimar la posición de un punto en espacio 3D utilizando multilateración. El número de grupos N de cuatro elementos se puede calcular utilizando la siguiente ecuación:

$$N = C_4^n = \frac{n!}{4!(n-4)!} = \frac{n(n-1)(n-2)(n-3)}{4!} \quad (9)$$

Para cada uno de los N grupos podemos obtener una posición estimada del punto de interés. Para esto, se utiliza el método de mínimos cuadrados y la red neuronal empleadas en la sección 4.1.3.

Después de realizar la localización utilizando cada uno de los grupos, se obtienen N posiciones estimadas del punto de interés. Si se toma el promedio o el centroide de las N estimaciones como la posición final, esta se verá influenciada por las estimaciones con un nivel alto de error. Para lidiar con esto, se emplea el método de agrupamiento de K-medias, con la idea de que las estimaciones con menor error se encontrarán más cerca las unas de las otras, y se pueden diferenciar de las que tienen más error, efectivamente eliminando las estimaciones con más error. El proceso para calcular K-medias se muestra en la figura 13.

Habiendo calculado los K grupos, se toma el centroide del grupo con más elementos como la estimación final.

Para los experimentos, se consideró un ambiente de 50 por 50 metros, en el cual se colocaron 50 puntos de acceso Wi-Fi (APs) aleatoriamente distribuidos de forma uniforme, con alturas que varían entre 0 y 5 metros, de tal manera que cubran todo el ambiente. Con estos 50 APs, se toman distintos grupos de 4 APs para realizar la localización. Se realizaron pruebas tomando 10, 20, 30, 40, 50, 60, 70, 80, 90 y 100

grupos de 5 APs, con los cuales se utiliza el algoritmo de agrupamiento de K-medias. Para cada grupo, se estima la localización utilizando el método de mínimos cuadrados descrito en la sección 2.2 y la red neuronal mostrada en la figura 28.

Para el algoritmo de agrupamiento de K-medias se necesita proporcionar el número de grupos K que se crearan. En el trabajo original de Luo et al. (2022) se tomó un valor $K=2$, después de realizar un análisis del efecto que tiene el valor de K en el error de localización. En este análisis se mostró como el error en la localización disminuye a medida que aumenta el número de grupos, estabilizándose alrededor de $K=6$, como se observa en la figura 29. Es por esto que para nuestro trabajo se tomó un valor de $K=6$.

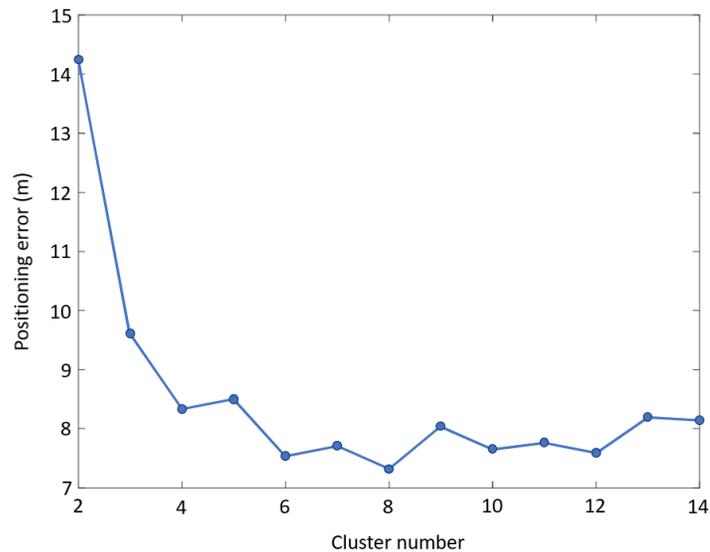


Figura 29. Análisis del efecto del número de grupos K en el error en localización para el método de multilateración utilizando agrupamiento de K-medias. Tomado de Luo et al. (2022).

La red neuronal empleada se entrenó utilizando 10,000 ejemplos de entrenamiento generados aleatoriamente, es decir, 10,000 escenarios de 50 por 50 metros donde se colocaron aleatoriamente 50 APs y un punto a localizar. Se entrenó durante 300 épocas, utilizando como función de pérdida el error absoluto medio (MAE por sus siglas en inglés), una tasa de aprendizaje de 0.0001, el optimizador Adam y un *batch size* de 32.

4.2. Resultados

En esta sección se presentan los resultados obtenidos utilizando los tres métodos descritos para localización utilizando señales de Wi-Fi: *fingerprinting*, multilateración y multilateración utilizando agrupamiento de K-medias.

4.2.1. Fingerprinting

En primer lugar, en la figura 30 se muestran los resultados para el método de *fingerprinting*, empleando los tres tipos de modelos de aprendizaje de máquina (KNN, SVM y red neuronal (NN)) descritos en la sección 4.1.2. En las tablas 3, 4, 5 y 6 se muestran los mismos resultados de forma numérica para los experimentos tomando muestras cada 0.5, 1, 1.5 y 2 metros respectivamente.

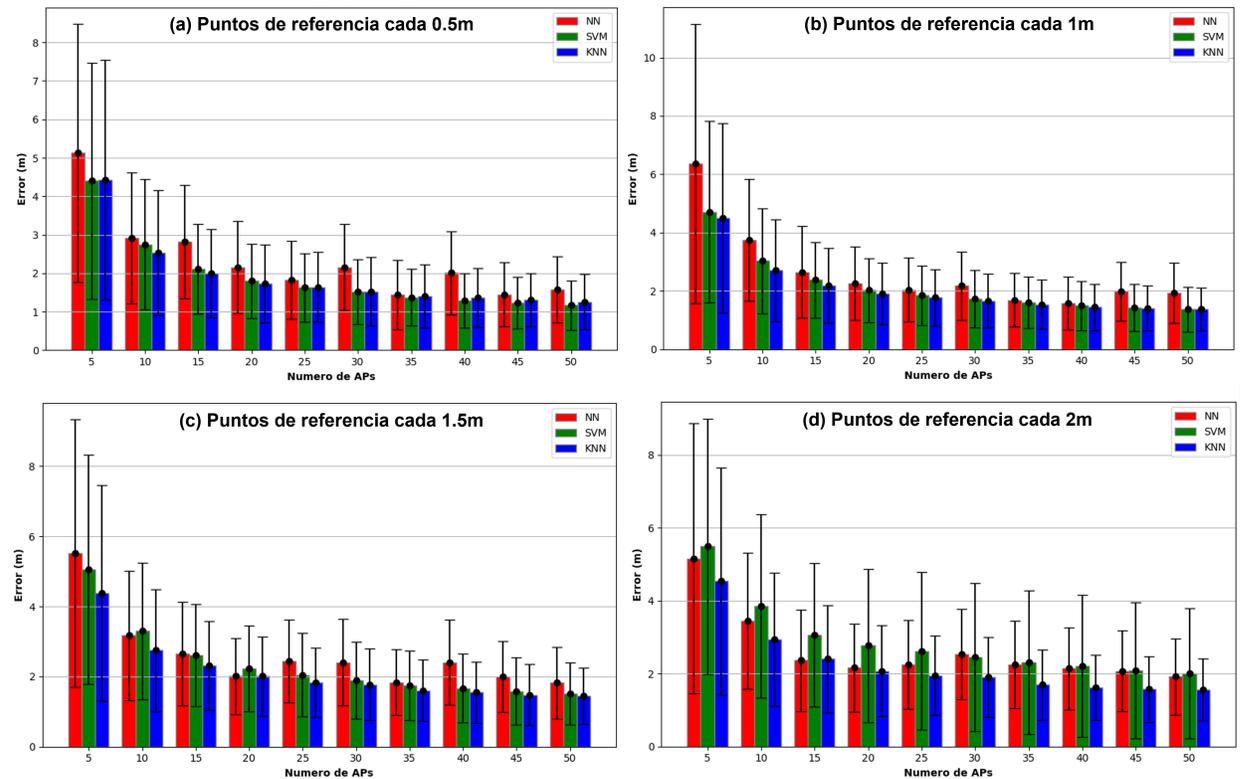


Figura 30. Resultados para los tres métodos de aprendizaje de máquina (KNN, SVM y red neuronal (NN)) utilizados para *fingerprinting*, empleando diferentes números de APs y considerando distancias entre puntos de referencia de (a) 0.5m, (b) 1m, (c) 1.5 y (d) 2m, mostrando la media y desviación estándar del error en localización

Como se observa en la figura 30 y tablas 3, 4, 5 y 6, los mejores resultados se obtuvieron al tomar muestras de referencia cada 0.5m, y a medida que aumentaba la distancia entre las muestras de referencia, aumentaba también la media del error. De igual manera, se observa como a medida que aumentan el número de APs la media del error en la localización disminuye. Para la mayoría de los casos, el modelo que obtuvo los mejores resultados en términos de la media del error fue KNN. La excepción fueron aquellos experimentos donde se tomaron muestras cada 0.5m y se tienen más de 20 APs en el ambiente, en estos casos, SVM mostró mejor desempeño. El menor error promedio obtenido, 1.1672m, fue utilizando SVM, con muestras de referencia cada 0.5m y 50 APs. Por otro lado, el modelo que obtuvo los peores

resultados en términos de la media del error fue la red neuronal diseñada, existiendo una única excepción, cuando se tomaron muestras de referencia cada 2m, siendo SVM el que presentó peor desempeño en este caso.

Tabla 3. Resultados para el método de *fingerprinting* tomando puntos de referencia cada 0.5m, mostrando la media y desviación estándar del error en localización

Puntos de Referencia Cada 0.5m						
# APs	Red Neuronal		SVM		KNN	
	Media	StDev	Media	StDev	Media	StDev
5	5.1320	3.3545	4.4003	3.0631	4.4303	3.1179
10	2.9148	1.7018	2.7515	1.6932	2.5379	1.6227
15	2.8223	1.4794	2.1203	1.1661	1.9983	1.1527
20	2.1541	1.1945	1.8008	0.9621	1.7301	1.0075
25	1.8322	1.0128	1.6336	0.8898	1.6381	0.9121
30	2.1519	1.1181	1.5163	0.8433	1.5260	0.8904
35	1.4353	0.8980	1.3713	0.7401	1.4060	0.8191
40	2.0088	1.0836	1.2947	0.7104	1.3610	0.7670
45	1.4522	0.8275	1.2251	0.6676	1.3109	0.6958
50	1.5774	0.8693	1.1672	0.6360	1.2599	0.7094

Tabla 4. Resultados para el método de *fingerprinting* tomando puntos de referencia cada 1m, mostrando la media y desviación estándar del error en localización

Puntos de Referencia Cada 1m						
# APs	Red Neuronal		SVM		KNN	
	Media	StDev	Media	StDev	Media	StDev
5	6.3694	4.7886	4.7121	3.1125	4.4877	3.2484
10	3.7331	2.0878	3.0282	1.7948	2.6976	1.7485
15	2.6443	1.5645	2.3706	1.2991	2.1855	1.2851
20	2.2573	1.2641	2.0229	1.0898	1.8984	1.0602
25	2.0409	1.0881	1.8509	1.0218	1.7693	0.9774
30	2.1722	1.1701	1.7359	0.9795	1.6583	0.9157
35	1.6887	0.9077	1.6001	0.8730	1.5368	0.8374
40	1.5669	0.9073	1.4885	0.8471	1.4427	0.7929
45	1.9813	1.0047	1.4294	0.8059	1.4006	0.7685
50	1.9313	1.0287	1.3623	0.7615	1.3834	0.7321

Aunque en los resultados de los experimentos se observó que la media del error obtenida por KNN es menor en general, esto no nos permite concluir que sea el mejor modelo utilizado. Lo anterior es debido a que la desviación estándar del error es muy grande para todos los modelos, siendo en la mayoría de los casos mayor a la mitad de la media. Esto nos indica que existe una gran variación en el error de las estimaciones, demasiada para poder determinar con certeza que un modelo es mejor que otro en general.

Tabla 5. Resultados para el método de *fingerprinting* tomando puntos de referencia cada 1.5m, mostrando la media y desviación estándar del error en localización

Puntos de Referencia Cada 1.5m						
# APs	Red Neuronal		SVM		KNN	
	Media	StDev	Media	StDev	Media	StDev
5	5.5197	3.8076	5.0501	3.2687	4.3789	3.0777
10	3.1737	1.8416	3.3023	1.9499	2.7500	1.7436
15	2.6572	1.4800	2.6047	1.4592	2.3196	1.2695
20	2.0134	1.0859	2.2314	1.2226	2.0079	1.1309
25	2.4418	1.1832	2.0410	1.1921	1.8311	0.9904
30	2.4041	1.2279	1.8893	1.0886	1.7718	1.0289
35	1.8368	0.9395	1.7464	0.9839	1.6012	0.8767
40	2.4022	1.2171	1.6658	0.9831	1.5482	0.8797
45	1.9918	1.0157	1.5841	0.9511	1.4700	0.8752
50	1.8205	1.0175	1.5110	0.8899	1.4509	0.8067

Tabla 6. Resultados para el método de *fingerprinting* tomando puntos de referencia cada 2m, mostrando la media y desviación estándar del error en localización

Puntos de Referencia Cada 2m						
# APs	Red Neuronal		SVM		KNN	
	Media	StDev	Media	StDev	Media	StDev
5	5.1604	3.7121	5.4914	3.5058	4.5351	3.1189
10	3.4506	1.8725	3.8517	2.5182	2.9349	1.8206
15	2.3593	1.3890	3.0599	1.9761	2.3988	1.4675
20	2.1552	1.2058	2.7651	2.0940	2.0697	1.2541
25	2.2465	1.2192	2.6132	2.1624	1.9489	1.0936
30	2.5339	1.2351	2.4546	2.0270	1.8930	1.0975
35	2.2433	1.2033	2.3062	1.9614	1.6914	0.9596
40	2.1376	1.1338	2.2065	1.9571	1.6173	0.8945
45	2.0729	1.1099	2.0813	1.8745	1.5667	0.9003
50	1.9141	1.0413	2.0005	1.7866	1.5572	0.8587

4.2.2. Multilateración

El segundo método evaluado fue el de multilateración. En la figura 31 se muestran los resultados para este método, empleando la red neuronal (NN) descrita en la sección 4.1.3 y el método de mínimos cuadrados (LS) descrito en la sección 2.2. En la tabla 7 se muestran los mismos resultados de forma numérica.

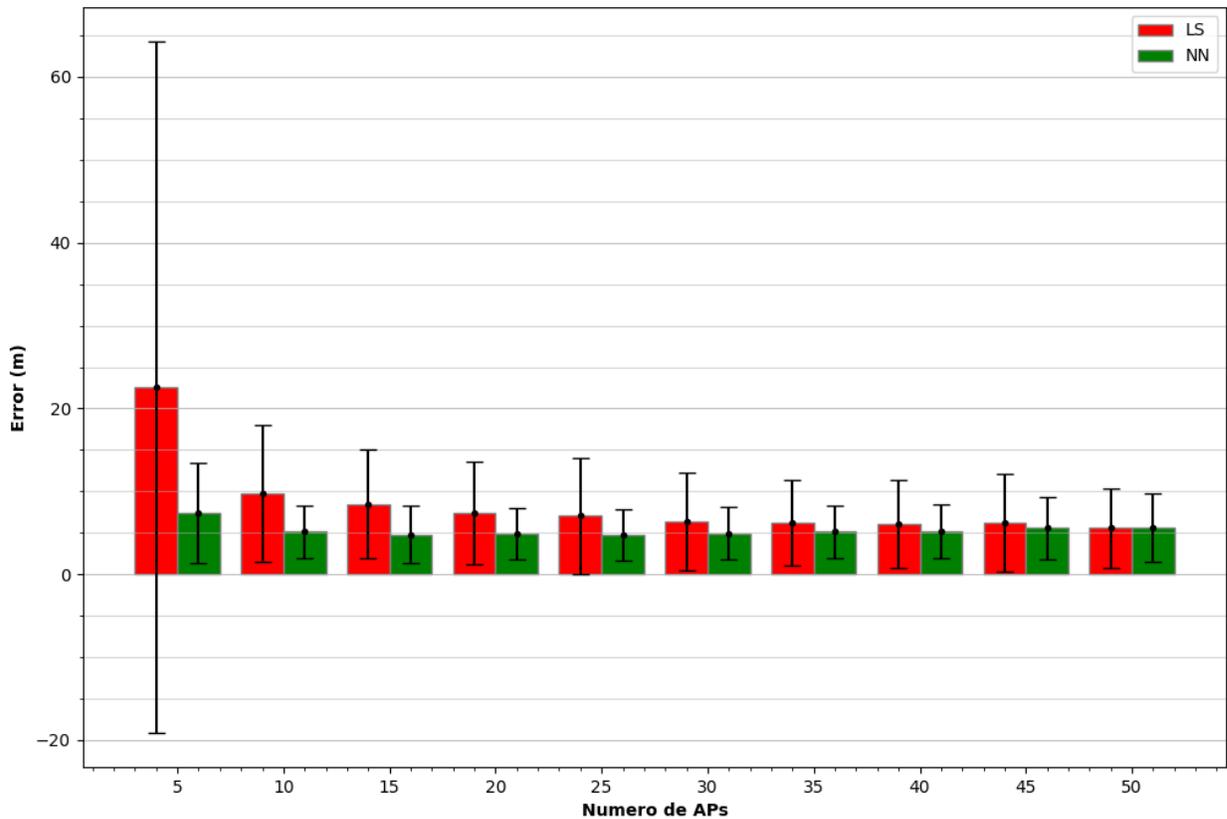


Figura 31. Resultados para los dos métodos utilizados para multilateración, mínimos cuadrados (LS) y una red neuronal (NN), empleando diferentes números de APs, mostrando la media y desviación estándar del error en localización.

Tabla 7. Resultados para el método de multilateración, mostrando la media y desviación estándar del error en localización.

# APs	Red Neuronal		Mínimos Cuadrados	
	Media	StDev	Media	StDev
5	7.3746	6.0012	22.5656	41.6626
10	5.1438	3.1321	9.7327	8.2061
15	4.8153	3.4614	8.4636	6.5314
20	4.8662	3.0837	7.3696	6.2039
25	4.7799	3.1083	7.0441	6.9601
30	4.9162	3.1811	6.3809	5.8990
35	5.1159	3.1621	6.1900	5.1877
40	5.1819	3.2553	6.0369	5.2748
45	5.5593	3.6877	6.2218	5.9044
50	5.6181	4.0750	5.6024	4.7838

De la misma manera que con el método de *fingerprinting*, se puede observar como al aumentar el número de APs en el ambiente, la media del error en la localización disminuye. Para la red neuronal utilizada, la media del error alcanza su mínimo cuando se toman como entrada los datos de 25 APs, alcanzando 4.7799 metros, y aumenta para números mayores de APs. Usando el método de mínimos cuadrados, la

media del error disminuye aumentando el numero de APs utilizados, llegando hasta una media mínima de 5.6024 al emplear 50 APs.

4.2.3. Multilateración utilizando agrupamiento de K-medias

El tercer y último método evaluado para localización con señales Wi-Fi fue el de multilateración utilizando agrupamiento de K-medias. En la figura 32 se muestran los resultados para este método, empleando la red neuronal descrita en la sección 4.1.3 y el método de mínimos cuadrados descrito en la sección 2.2. En la tabla 8 se muestran los mismos resultados de forma numérica.

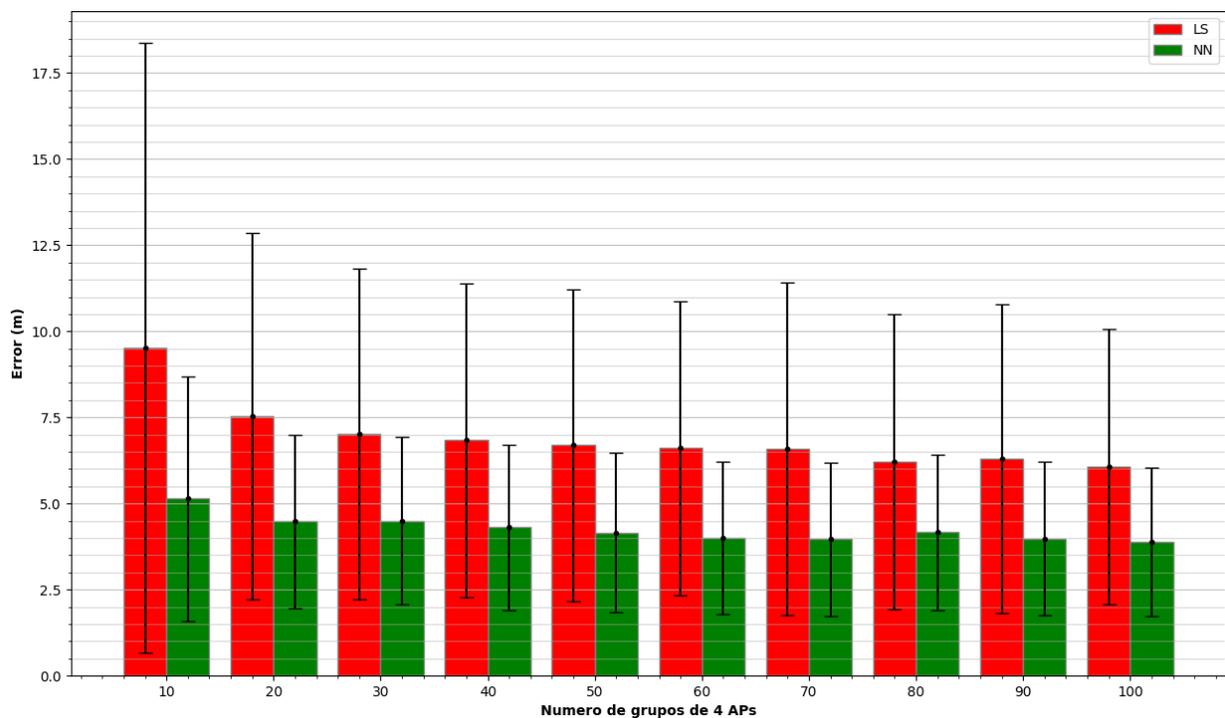


Figura 32. Resultados para los dos métodos utilizados para multilateración con agrupamiento de K-medias, mínimos cuadrados (LS) y una red neuronal (NN), empleando diferentes números grupos de 4 APs, mostrando la media y desviación estándar del error en localización.

De la figura 32 y tabla 8 se puede observar que al aumentar el número de grupos de 4 APs en el ambiente, la media del error en la localización disminuye para ambos métodos evaluados, aunque parece estabilizarse para más de 50 grupos. El mejor resultado se obtuvo utilizando la red neuronal con 100 grupos, con una media del error de 3.8781 metros. Sin embargo, no parece haber una diferencia significativa entre los resultados de 60, 70, 80 90 y 100 grupos. Algo similar sucede los resultados obtenidos utilizando mínimos cuadrados, donde se obtuvo un error mínimo de 6.0794 metros.

Tabla 8. Resultados para el método de multilateración utilizando agrupamiento de K-medias.

# Grupos	Red Neuronal		Minimos Cuadrados	
	Media	StDev	Media	StDev
10	5.1372	3.5404	9.5222	8.8645
20	4.4737	2.5196	7.5275	5.3208
30	4.4969	2.4255	7.0226	4.8070
40	4.3015	2.4129	6.8354	4.5654
50	4.1472	2.3163	6.6897	4.5203
60	4.0059	2.2097	6.6025	4.2690
70	3.9607	2.2339	6.5828	4.8325
80	4.1602	2.2562	6.2100	4.2916
90	3.9784	2.2299	6.3003	4.4926
100	3.8781	2.1535	6.0794	3.9961

4.3. Discusión

Como se puede observar en la sección 4.2, el método de localización utilizando señales Wi-Fi que presentó los mejores resultados fue el método de *fingerprinting*, lo cual es consistente con lo reportado en la literatura, siendo uno de los métodos más populares de localización en interiores que emplea señales Wi-Fi. En el mejor de los casos, el método de *fingerprinting* presentó un error medio de 1.55 metros, utilizando un modelo KNN y suponiendo que se tienen 50 APs en el ambiente. Este nivel de error puede ser aceptable para algunas aplicaciones particulares, por ejemplo, determinar si está dentro de una habitación específica o no. Sin embargo, es superior a lo que se esperaría para un método de localización en interiores que permita realizar tareas como la navegación autónoma. Además, es importante mencionar que contar con 50 APs en un ambiente de 50 por 50 metros no es un escenario muy realista. Para un ambiente con esa área se esperaría tener en realidad entre 15 y 25 APs aproximadamente, dependiendo también de la geometría del ambiente, por lo que el error en la estimación de posición aumentaría.

Dentro de la literatura analizada se observó que el error mínimo en la localización utilizando únicamente la intensidad de señales Wi-Fi en el ambiente es de alrededor de un metro, siendo un error relativamente alto comparado con otros métodos de localización que utilizan otros tipos de datos. La gran cantidad de error presente en las señales Wi-Fi se debe en buena medida a que estas son muy susceptibles a las condiciones ambientales, la geometría del espacio interior y los obstáculos presentes en este. Además, el método de *fingerprinting* tiene como desventaja, lo tardado y laborioso que puede resultar la fase de calibración fuera de línea. Considerando el número mínimo de puntos de referencia utilizados en este trabajo, 2500, si asumimos que para cada punto tomaría 10 segundos capturar los datos necesarios (una

cantidad de tiempo bastante conservadora), se traduce en casi 7 horas para capturar la base de datos completa, que aumentaría si deseamos capturar puntos adicionales para tener un mejor desempeño en la localización. Esto se complica aún más si consideramos los desafíos que conlleva conocer la posición exacta de los puntos de referencia, cosa que es necesaria para entrenar el modelo de localización a ser utilizado, pudiendo llegar a requerirse un complejo sistema de posicionamiento para obtener estos datos *ground truth*. Esta es una de las razones por las que se optó por trabajar totalmente en simulación, siendo trivial en este caso conocer la posición verdadera de los puntos de referencia y capturar tantos como sea necesario.

Otra desventaja del método de *fingerprinting* surge de lo dependiente que es la intensidad de las señales Wi-Fi a las condiciones del ambiente. La fase de calibración en fuera de línea se realiza con el ambiente en un cierto estado, y si este cambia para la fase de localización en línea, por ejemplo al mover un mueble, agregar un obstáculo o un punto de acceso Wi-Fi adicional, el error en la localización puede aumentar drásticamente, al punto de que se requiera realizar la fase de calibración nuevamente. Por lo tanto, para ambientes que cambian constantemente, esta no puede ser una opción factible, debido a lo tardado y laborioso que puede ser capturar cada punto de referencia en el ambiente manualmente.

De lo mencionado anteriormente se concluye que utilizar la intensidad de las señales Wi-Fi en el ambiente como único dato para estimar la localización de un robot móvil no es una solución eficaz ni eficiente. Los métodos aquí presentes pudieran ser más útiles para aplicaciones en donde se requiera localizar un dispositivo solamente a nivel de habitación, por ejemplo, si se desea conocer en que oficina de un piso se encuentra una persona.

Existen otros métodos más precisos de localización que utilizan datos adicionales de las señales Wi-Fi del ambiente, por ejemplo el tiempo de vuelo o ángulo de llegada de la señal. No obstante, para poder estimar estos valores se requiere de equipo especializado, a diferencia de la intensidad de la señal, que cualquier tarjeta de red disponible en computadoras o teléfonos celulares puede medir. Además, poder simular estos datos adicionales presenta un desafío mucho más grande, por lo que métodos de localización que hagan uso de estos no se exploraron en este trabajo.

Dado que en este trabajo se implementó un modelo simplificado para simular la intensidad de una señal Wi-Fi que ignora muchos aspectos de como se comportaría una señal en un ambiente real, se espera que el resultado de los métodos presentados en este capítulo no muestren el mismo nivel de desempeño si son aplicados en ambientes reales. Sin embargo, se pueden considerar como una buena aproximación a métodos que pudieran implementarse en situaciones reales.

Capítulo 5. Localización basada en datos inerciales

En este capítulo se describe la metodología seguida para la implementación de un método de aprendizaje de máquina que utilice los datos de un IMU montado sobre el robot para resolver el problema de localización en interiores, además de los experimentos realizados y los resultados obtenidos.

5.1. Metodología

El desarrollo de esta parte del trabajo se dividió en dos fases principales: 1) generación y recolección de datos para entrenamiento utilizando un robot simulado, y 2) entrenamiento, implementación y evaluación de distintos modelos de localización empleando la base de datos creada.

5.1.1. Generación de datos

Para simular el robot móvil y los ambientes a recorrer se utilizó el simulador 3D para robótica de código abierto *Gazebo* (Open Robotics, 2023), junto con el paquete de simulación de *TurtleBot3* (ROBOTIS, 2021). Este paquete incluye el modelo físico y de control del robot, así como los modelos de los diferentes sensores que se encuentran instalados en este. El modelo del robot como se observa en el simulador es mostrado en la figura 33. Dentro de estos sensores simulados se encuentra el IMU, el cual otorga, entre otros datos, la aceleración lineal en X, Y y Z, y la velocidad angular en X, Y y Z del robot. Estos valores se reportan a una frecuencia de 200 Hz, y son los que se almacenarán y utilizarán para estimar la localización del robot.

Se colocó el robot simulado en un ambiente vacío de 100 por 100 metros, en el cual se realizaron 26 recorridos diferentes. Durante cada recorrido se controló el robot de forma manual, variando su velocidad lineal entre 0 y 0.26 m/s, y su velocidad angular entre -1.82 y 1.82 rad/s. Se almacenaron las aceleraciones lineales en X, Y y Z, las velocidades angulares en X, Y y Z, y la posición real del robot en el plano XY. En total, se capturaron 13.32 horas de datos, lo que se traduce en 8,154,299 registros en la base de datos.

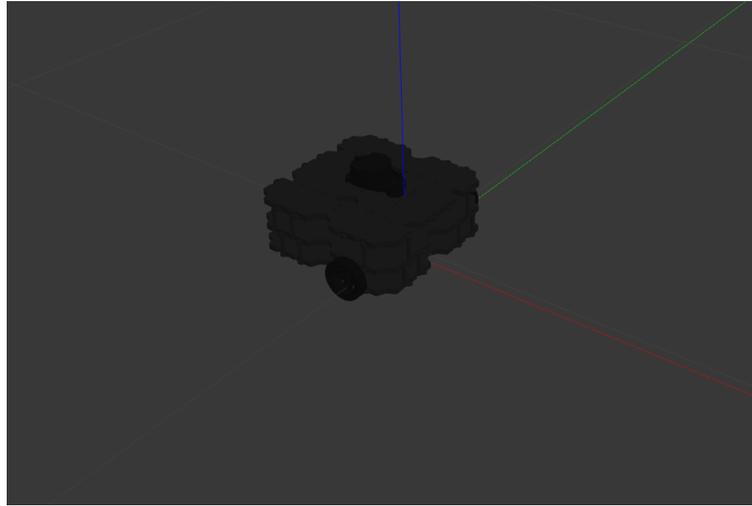


Figura 33. Robot *TurtleBot3* simulado utilizado para los experimentos.

Las aceleraciones lineales y velocidades angulares reportadas por el IMU del robot se encuentran en el marco de referencia local del sensor, por lo que el primer paso es convertir estas mediciones al marco de referencia global. Para esto, se utiliza la orientación del robot reportada por el mismo IMU, la cual es representada por un cuaternión $q = q_w + q_x i + q_y j + q_z k$. En la práctica, solo los cuatro coeficientes son utilizados para especificar un cuaternión, de la forma $q = (q_w, q_x, q_y, q_z)$. Las aceleraciones lineales pueden ser rotadas por el cuaternión q utilizando la siguiente fórmula:

$$a' = q \otimes a \otimes q^{-1} \quad (10)$$

donde $a = (0, a_x, a_y, a_z)$ es el vector de aceleración lineal en forma de cuaternión, a' es el vector de aceleración rotado, $q^{-1} = (q_w, -q_x, -q_y, -q_z)$ es la conjugación del cuaternión de rotación q , y \otimes es el producto de Hamilton. De la misma manera, las velocidades angulares pueden ser rotadas al marco de referencia global con la siguiente fórmula:

$$\omega' = q \otimes \omega \otimes q^{-1} \quad (11)$$

donde $\omega = (0, \omega_x, \omega_y, \omega_z)$ es el vector de velocidad angular en forma de cuaternión, y ω' es el vector de velocidad angular rotado.

Para procesar los datos convertidos al marco de referencia global, se tomó como base lo realizado por

Lima et al. (2019). Se divide el total de los datos en ventanas de tiempo de 200 muestras cada una (1 segundo de datos), con un tamaño de salto o *stride* de 50 muestras (0.25 segundos). Para cada una de estas ventanas se calcula el cambio en la posición del robot correspondiente, de forma que dada una ventana de 200 muestras y un tamaño de salto de 50, el cambio en la posición se calcula de la muestra #75 a la muestra #125. Esto se ilustra en la figura 34.

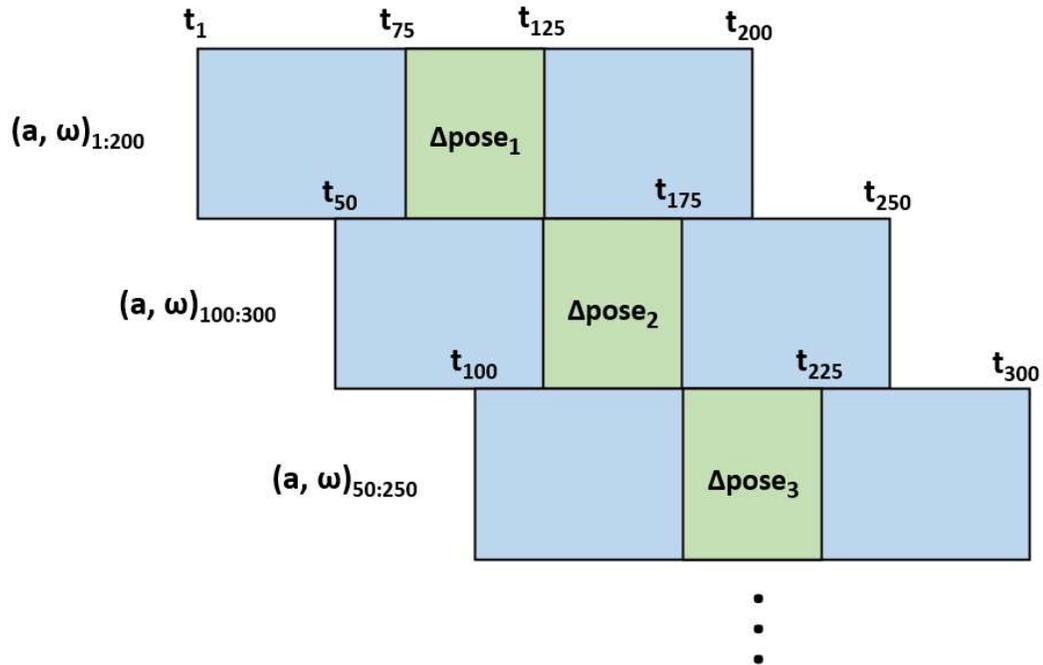


Figura 34. Procesamiento para generar las ventanas de datos y cambios en la posición del robot, que serán utilizados como entradas y salidas del modelo respectivamente.

Procesando todos los recolectados durante los recorridos realizados, se generaron un total de 162,995 ventanas o muestras de entrenamiento, cada una con el cambio correspondiente en la posición relativa del robot.

5.1.2. Modelos para localización

Una vez creada la base de datos, se seleccionaron de la literatura cuatro arquitecturas diferentes para su implementación y comparación. A continuación, se presenta una descripción de las mismas.

ResNet-Small. Las tres versiones de ResNet utilizadas en el trabajo se muestran en la figura 36.

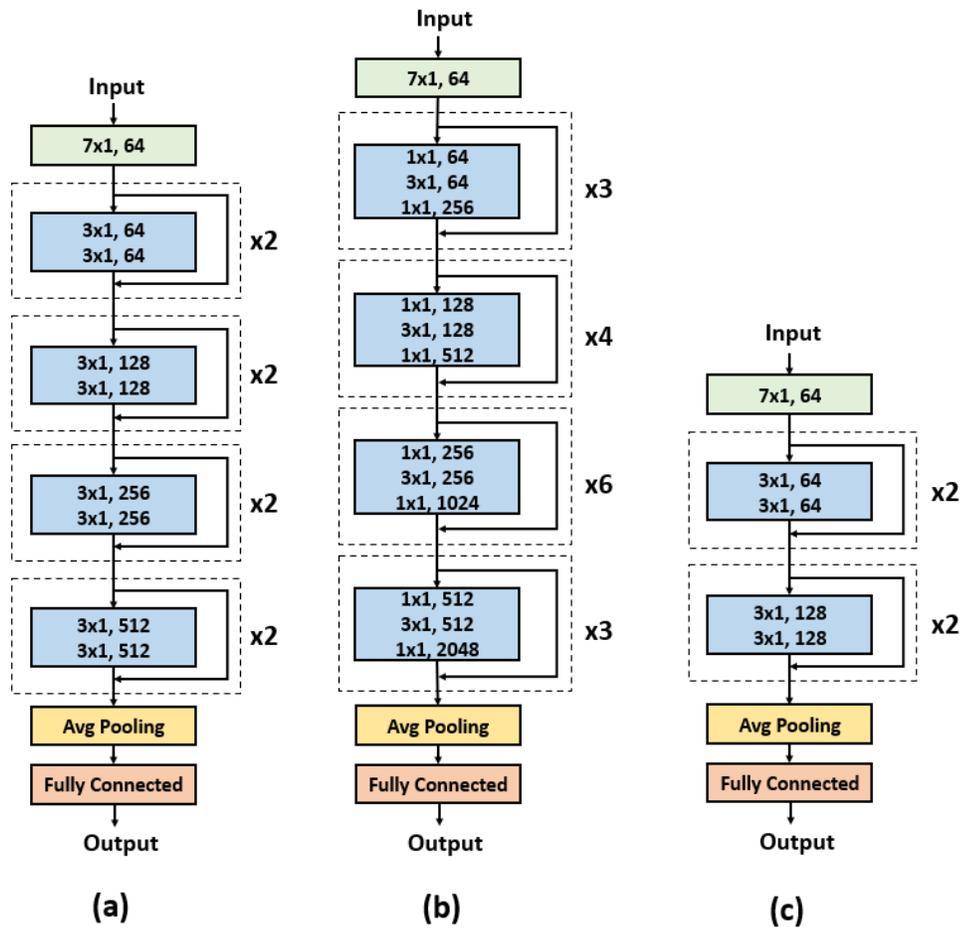


Figura 36. Arquitecturas de (a) ResNet-18, (b) ResNet-50 y (c) ResNet-Small

5.1.2.2. MobileNetV3

Presentada por Howard et al. (2019), MobileNetV3 es la tercera generación de MobileNet, una red neuronal convolucional para problemas de clasificación, detección y segmentación de imágenes, diseñada para correr de manera eficiente en dispositivos móviles y embebidos de recursos limitados. Existen dos variaciones de esta red, MobileNetV3-Small y MobileNetV3-Large, para dispositivos de bajos y altos recursos de cómputo respectivamente. MobileNetV1 (Howard et al., 2017) introdujo el concepto de convolución a profundidad (*depth-wise convolution*) para reducir el número de parámetros de la red y mejorar el rendimiento en dispositivos móviles. MobileNetV2 (Sandler et al., 2018) agregó bloques residuales invertidos (*Inverted Residual Blocks*), que tienen una estructura de expansión-filtrado-compresión, a diferencia de los bloques residuales regulares, que siguen una estructura de compresión-filtrado-expansión. Mobile-

NetV3 añade al modelo capas *squeeze and excitation* (Hu et al., 2018), las cuales ayudan a asignar pesos desiguales a los diferentes canales de la entrada al crear los mapas de características, a diferencia de una red convolucional regular que asigna pesos iguales. Para este trabajo, se implementaron versiones de MobileNetV3-Small y MobileNetV3-Large para datos de 1 dimensión. La arquitectura general de MobileNetV3 se muestra en la figura 37.

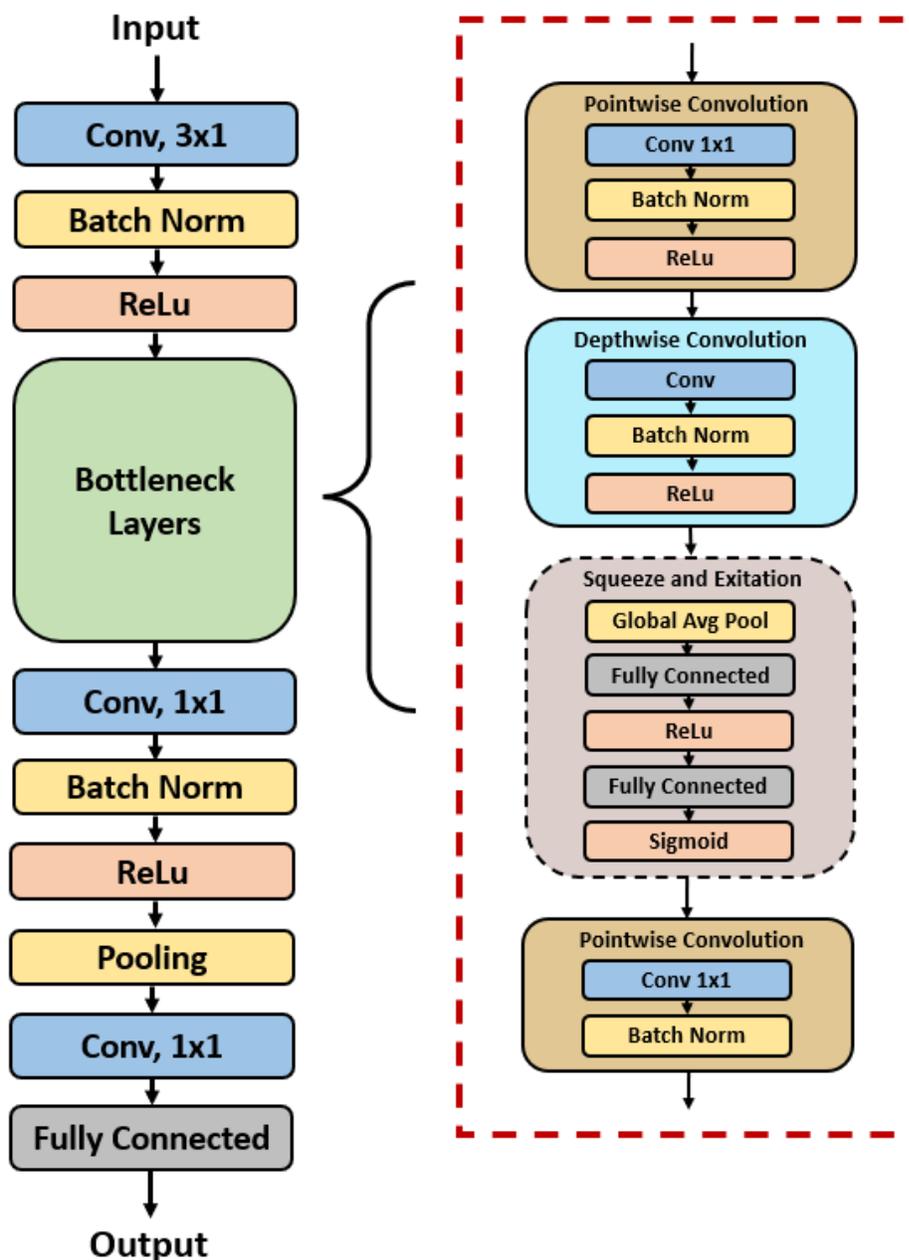


Figura 37. Arquitectura general de MobileNetV3. MobileNetV3-Small y MobileNetV3-Large se diferencian por el número de capas *Bottleneck* que emplean, contando con 11 y 15 capas respectivamente.

5.1.2.3. Red CNN + LSTM

Lima et al. (2019) proponen una red neuronal compuesta por cuatro capas convolucionales (dos capas para la aceleración lineal de entrada, y dos para la velocidad angular de entrada) y dos capas LSTM bidireccionales. En el trabajo original se utiliza este modelo para estimar la posición y orientación de un dispositivo móvil en el espacio tridimensional. Sin embargo, para esta investigación, solo nos interesa estimar la posición del robot en el plano XY, por lo que se reduce la capa totalmente conectada final de siete a dos parámetros.

5.1.2.4. RBCN-Net

Propuesta por Zhu et al. (2023), RBCN-Net es una arquitectura de red neuronal para localización de un dispositivo móvil utilizando datos inerciales. Toman como base la arquitectura de ResNet-18, y se agregan módulos de atención NAM (Liu et al., 2021) a cada uno de los ocho bloques *Basic Block* de la red, y un bloque de atención CBAM (Woo et al., 2018) al inicio de la red. Estos bloques de atención se agregan buscando que aprendan a identificar las características más importantes de los datos. Además, se agregan a la salida de la red dos capas LSTM bidireccionales, con la idea de aprender las diferencias temporales entre los datos. Dado que el código de la implementación original de esta arquitectura no está disponible para el público, se implementó una versión propia siguiendo lo descrito en el artículo de la manera más cercana posible.

5.1.3. Configuración experimental y métricas

En total, se entrenaron y probaron siete modelos diferentes:

- ResNet-18
- ResNet-50
- ResNet-Small
- MobileNetV3-Small
- MobileNetV3-Large
- CNN + LSTM

- RBCN-Net

Todos estos modelos se entrenaron durante 1000 épocas con los 162,995 ejemplos de entrenamiento generados. Se utilizó una tasa de aprendizaje de 0.0001, que es reducida por un factor de 0.1 si la pérdida de validación no mejora después de 20 épocas. Se utiliza el error absoluto medio (MAE por sus siglas en inglés) como función de pérdida, el optimizador Adam y un *batch size* de 32.

Para probar los modelos entrenados, se generaron datos de prueba de 10 recorridos, siguiendo las mismas especificaciones utilizadas para generar los datos de entrenamiento, terminando con 4.41 horas de datos para probar los modelos.

Para una ventana de datos individual, el modelo entrega como salida el cambio en la posición relativa del robot durante ese periodo de tiempo. Podemos reconstruir la trayectoria estimada del robot sumando cada uno de estos cambios de posición relativos, asumiendo que se conoce el punto inicial.

Las trayectorias reconstruidas con las salidas de los modelos se comparan con las trayectorias reales utilizando cuatro métricas diferentes: la raíz del error cuadrático medio (*RMSE*), precisión de posicionamiento promedio (*RTTE*), y error absoluto medio en X y Y (*MAE_x* y *MAE_y*).

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n [(x_i - \tilde{x}_i)^2 + (y_i - \tilde{y}_i)^2]} \quad (12)$$

$$RTTE = \frac{\sum_{i=1}^n \sqrt{[(x_i - \tilde{x}_i)^2 + (y_i - \tilde{y}_i)^2]}}{n * \text{trayectoria_length}} \quad (13)$$

$$MAE_x = \frac{1}{n} \sum_{i=1}^n |x_i - \tilde{x}_i| \quad (14)$$

$$MAE_y = \frac{1}{n} \sum_{i=1}^n |y_i - \tilde{y}_i| \quad (15)$$

De acuerdo a estas métricas, se seleccionan los dos mejores modelos y se comparan estos con dos métodos de SLAM, *GMapping* (Grisetti et al., 2007) y *Karto* (Gerkey, 2019), utilizando tres recorridos de prueba en un ambiente simulando el interior de un edificio (figura 38).

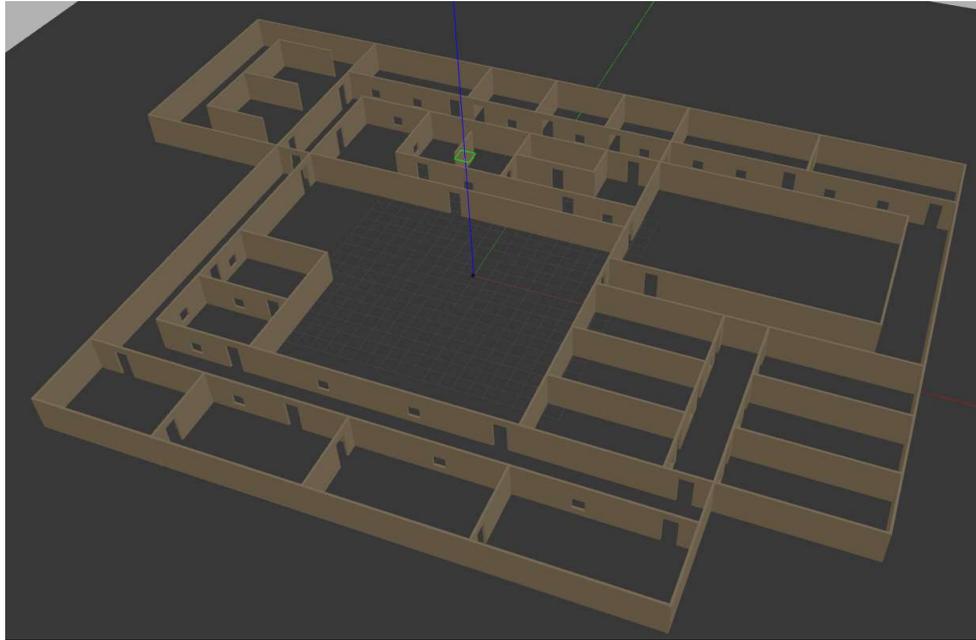


Figura 38. Ambiente simulado utilizado para recorridos de comparación con algoritmos de SLAM

5.2. Resultados

En esta sección se presentan los resultados obtenidos y la comparación entre los diferentes modelos entrenados, evaluándolos en 10 recorridos de prueba distintos. Los resultados para cada uno de estos recorridos se muestran en las tablas 9, 10, 11, 12 y 13. En la tabla 14 se presentan los promedios resultantes de las pruebas con todos los recorridos. Las celdas de color verde en las tablas indican el modelo que tuvo el mejor resultado en cada métrica particular.

Tabla 9. Resultados para recorridos de prueba #1 y #2

	Recorrido Prueba #1				Recorrido Prueba #2			
	RMSE	RTE	MAEx	MAEy	RMSE	RTE	MAEx	MAEy
ResNet-18	0.6416	0.1156	0.4277	0.3122	1.0616	0.2038	0.6930	0.6583
ResNet-50	1.5088	0.2738	1.162	0.6475	1.4269	0.2365	0.5519	0.8928
ResNet-Small	1.4648	0.2570	1.2894	0.2551	1.0456	0.1898	0.4576	0.7572
MobileNetV3-Small	0.8059	0.1402	0.2014	0.6795	0.8884	0.1512	0.5847	0.4437
MobileNetV3-Large	0.8916	0.1593	0.7857	0.1968	0.6187	0.1184	0.3744	0.3715
CNN + LSTM	1.4959	0.2592	0.931	0.8195	2.0668	0.3762	0.5243	1.7495
RBCN-Net	11.1637	2.0186	6.5044	6.8806	9.971	1.8577	4.1994	7.7963

Tabla 10. Resultados para recorridos de prueba #3 y #4

	Recorrido Prueba #3				Recorrido Prueba #4			
	RMSE	RTE	MAEx	MAEy	RMSE	RTE	MAEx	MAEy
ResNet-18	2.5598	0.7157	0.8580	1.7486	0.7967	0.2513	0.6453	0.3078
ResNet-50	4.4705	1.4035	1.3058	3.6815	1.1056	0.3563	0.7001	0.5551
ResNet-Small	3.1833	1.0376	1.0376	2.6340	0.6510	0.2017	0.4389	0.3619
MobileNetV3-Small	3.0186	0.9947	1.9684	1.7777	0.6697	0.2133	0.2599	0.5375
MobileNetV3-Large	2.2497	0.4470	1.1300	0.5670	0.7016	0.2266	0.4126	0.4997
CNN + LSTM	2.7388	0.9317	0.8105	2.409	0.9696	0.2934	0.7562	0.331
RBCN-Net	5.3366	1.6357	2.9768	2.9473	13.1708	4.1104	10.5325	4.9518

Tabla 11. Resultados para recorridos de prueba #5 y #6

	Recorrido Prueba #5				Recorrido Prueba #6			
	RMSE	RTE	MAEx	MAEy	RMSE	RTE	MAEx	MAEy
ResNet-18	0.9943	0.1676	0.6128	0.4632	0.5015	0.1722	0.3449	0.2145
ResNet-50	3.5502	0.6912	1.8084	2.8012	1.3791	0.4892	1.0649	0.6131
ResNet-Small	2.0609	0.4051	0.6733	1.7891	0.6577	0.2273	0.1917	0.5267
MobileNetV3-Small	1.1743	0.2331	0.9140	0.5714	0.4729	0.1761	0.1604	0.4194
MobileNetV3-Large	2.1416	0.4204	1.9637	0.4736	1.2467	0.4674	0.7556	0.8869
CNN + LSTM	2.9443	0.5853	1.3255	2.4366	1.8732	0.7121	0.5501	1.6823
RBCN-Net	12.2355	2.1376	4.4572	8.9675	5.0156	1.6492	2.6558	2.9395

Tabla 12. Resultados para recorridos de prueba #7 y #8

	Recorrido Prueba #7				Recorrido Prueba #8			
	RMSE	RTE	MAEx	MAEy	RMSE	RTE	MAEx	MAEy
ResNet-18	1.3603	0.3199	1.0573	0.7357	2.5739	0.6137	2.2411	0.7573
ResNet-50	2.2473	0.5263	1.1677	1.8315	4.6289	1.1107	3.5679	2.2722
ResNet-Small	0.8365	0.1843	0.3061	0.6749	3.684	0.8926	0.8594	3.3474
MobileNetV3-Small	0.6116	0.129	0.2358	0.4318	1.1048	0.2684	0.9618	0.3080
MobileNetV3-Large	0.5006	0.1068	0.3027	0.3013	1.4392	0.3225	1.1478	0.5043
CNN + LSTM	0.8574	0.1955	0.5782	0.4804	4.2898	1.0605	3.654	1.7334
RBCN-Net	14.5396	3.3319	12.0094	5.6612	10.8696	2.3811	4.452	7.5131

Tabla 13. Resultados para recorridos de prueba #9 y #10

	Recorrido Prueba #9				Recorrido Prueba #10			
	RMSE	RTE	MAEx	MAEy	RMSE	RTE	MAEx	MAEy
ResNet-18	0.906	0.2493	0.7312	0.2637	0.3359	0.0955	0.1536	0.23
ResNet-50	1.2365	0.3287	0.2157	1.1055	1.4244	0.3894	0.298	1.1425
ResNet-Small	1.0886	0.3056	0.2711	1.0111	0.9686	0.2547	0.3906	0.6742
MobileNetV3-Small	0.5970	0.1434	0.3865	0.2889	0.4505	0.1291	0.2401	0.2674
MobileNetV3-Large	0.5394	0.1393	0.3340	0.2958	0.7285	0.2019	0.5372	0.1723
CNN + LSTM	0.4593	0.1218	0.2498	0.3049	1.7452	0.4917	1.4627	0.3492
RBCN-Net	8.4454	2.2423	6.3783	3.8656	9.637	2.9076	4.6708	6.5769

Tabla 14. Resultados promedio para todos los recorridos

Promedios totales				
	RMSE	RTE	MAEx	MAEy
ResNet-18	1.1732	0.2905	0.7765	0.5691
ResNet-50	2.2978	0.5806	1.1842	1.5543
ResNet-Small	1.5641	0.3956	0.5916	1.2032
MobileNetV3-Small	0.9794	0.2579	0.5913	0.5725
MobileNetV3-Large	1.1058	0.2610	0.7744	0.4269
CNN+LSTM	1.9440	0.5027	1.0842	1.2296
RBCN-Net	10.0385	2.4272	5.8837	5.8100

Para ilustrar la diferencia entre los resultados de los diferentes modelos, en la figura 39 se muestran las trayectorias reconstruidas por los siete modelos entrenados para el recorrido de prueba #5.

De la tabla 14, que contiene los promedios de los recorridos de prueba por arquitectura, se puede observar que los mejores resultados en general fueron obtenidos por MobileNetV3-Small, siendo la que presentó el menor error promedio en la mayoría de las métricas, siendo superada únicamente por MobileNetV3-Large en una de las métricas de error, la que se encuentra en segundo lugar, y ResNet-18 en tercero. El modelo que mostró el peor desempeño fue RBCN-Net, con alrededor de 5 veces más error para todas las métricas que el segundo peor modelo, ResNet-50.

Con base en esto se seleccionaron dos modelos entrenados para compararlos con dos algoritmos de SLAM: MobileNetV3-Small, fue el modelo que presentó los mejores resultados promedio, y ResNet-18, que fue el modelo de la familia ResNet que mostró el mejor desempeño. Como algoritmos de SLAM se seleccionaron GMapping (Grisetti et al., 2007) y Karto (Gerkey, 2019), siendo estos los más populares y los empleados por defecto para la plataforma robótica *TurtleBot3* utilizada para esta investigación, además de que ambos hacen uso de dos fuentes de información diferentes: odometría calculada por *encoders* en las llantas del robot, y un sensor de rango láser.

Los resultados para MobileNetV3-Small, ResNet-18, GMapping y Karto en dos recorridos de prueba diferentes se muestran en las tablas 15 y 16. Las trayectorias reconstruidas por los algoritmos de SLAM para ambos recorridos de prueba se muestran en las figuras 40 y 42, mientras que las reconstruidas por los modelos entrenados se muestran en las figuras 41 y 43.

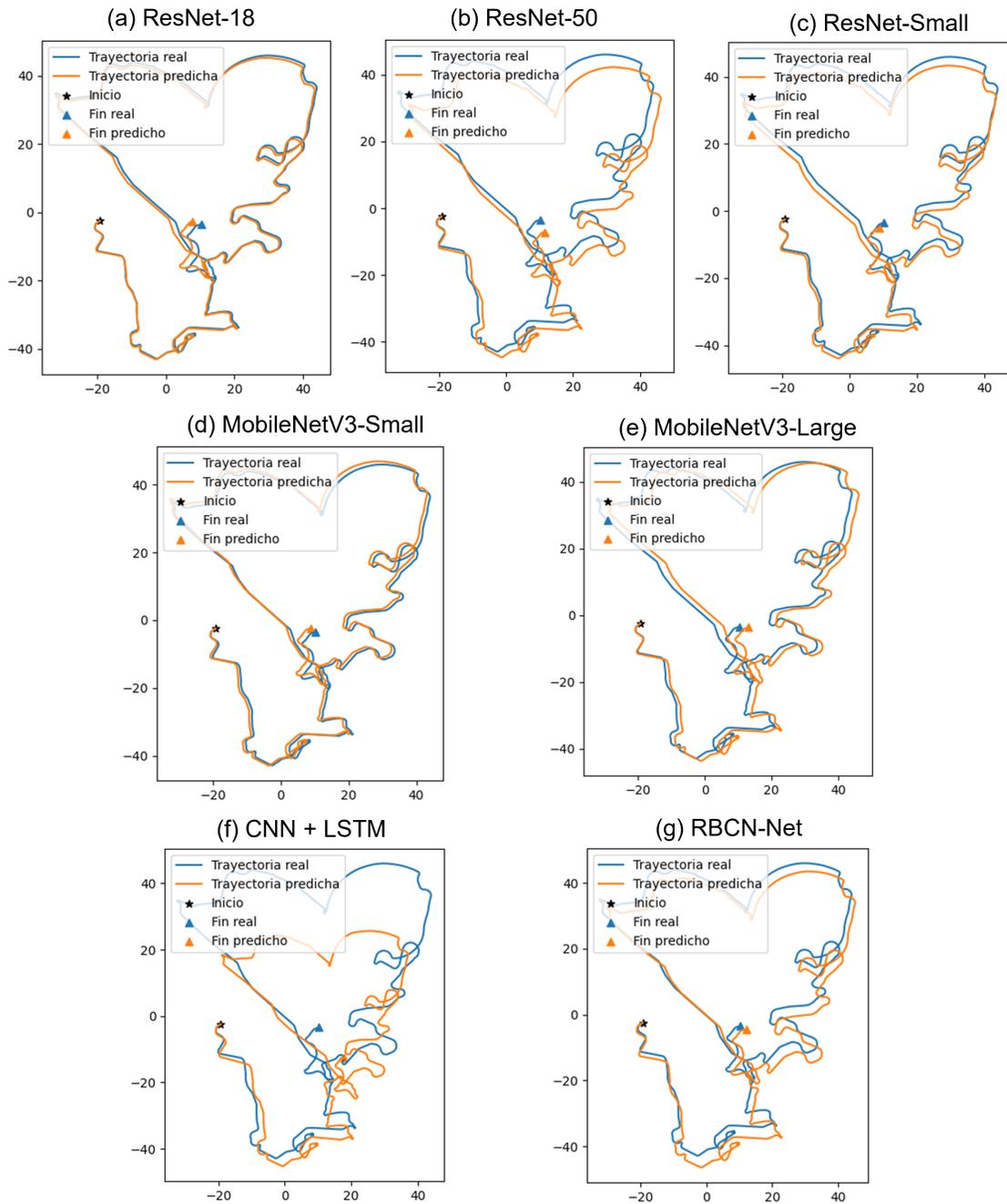


Figura 39. Trayectorias reconstruidas por todos los modelos para el recorrido de prueba #5

Tabla 15. Resultados para comparación con algoritmos SLAM en recorridos de prueba #1 y #2

	Prueba SLAM #1				Prueba SLAM #2			
	RMSE	RTE	MAEx	MAEy	RMSE	RTE	MAEx	MAEy
ResNet-18	4.0318	1.8263	0.5514	3.5839	11.5302	4.0571	5.0153	7.4983
MobileNetV3-Small	3.0658	1.3117	0.6668	2.5114	7.1285	2.4943	2.5105	4.8837
GMapping	7.7348	3.5691	6.2504	2.6399	5.0749	1.9010	3.6389	1.4521
Karto	1.3999	0.6139	0.8695	0.7986	1.2305	0.4071	0.7510	0.4479

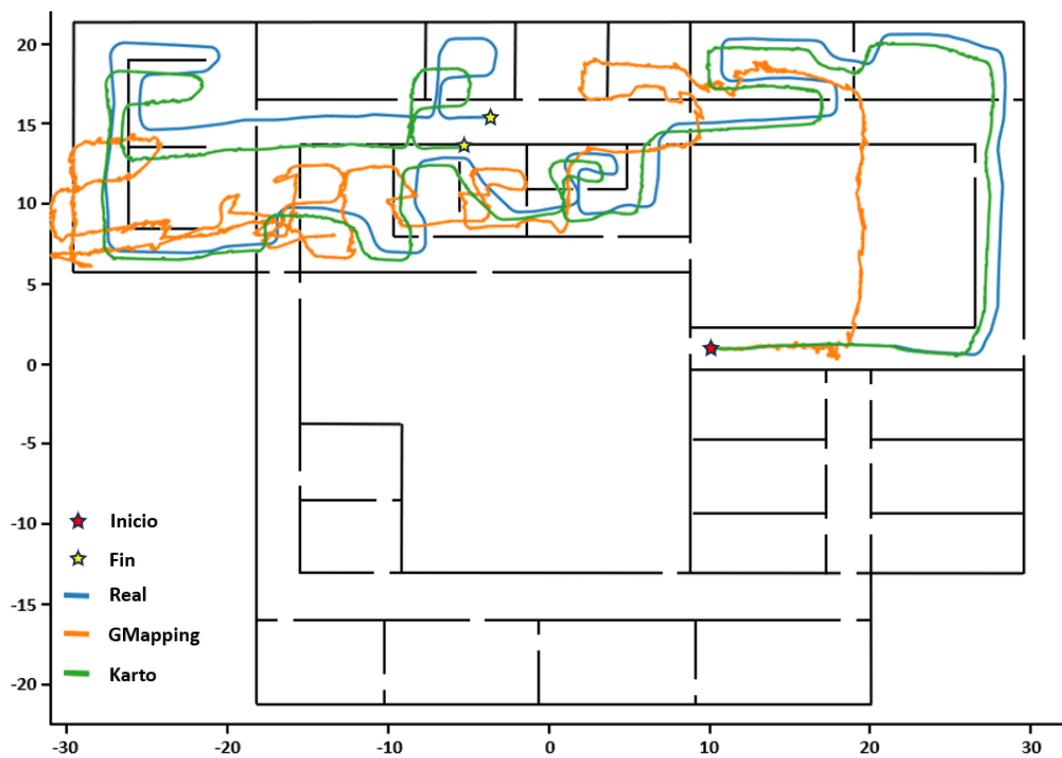


Figura 40. Recorrido de prueba #1, trayectoria real y estimadas por algoritmos de SLAM

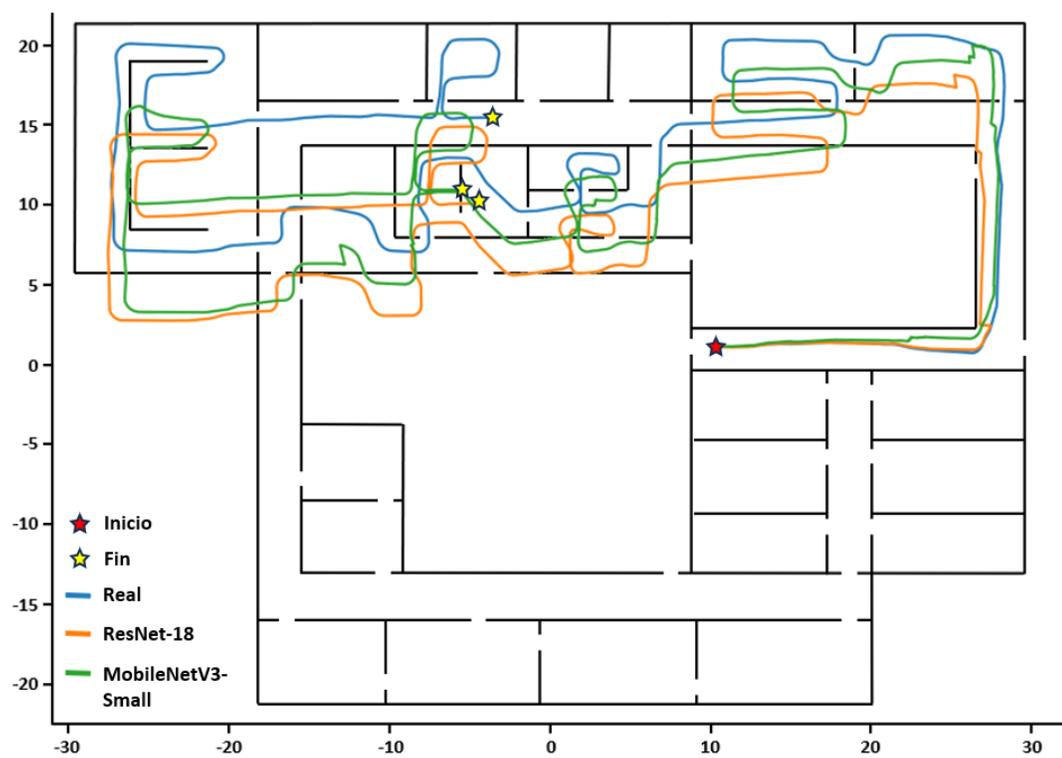


Figura 41. Recorrido de prueba #1, trayectoria real y estimadas por modelos entrenados

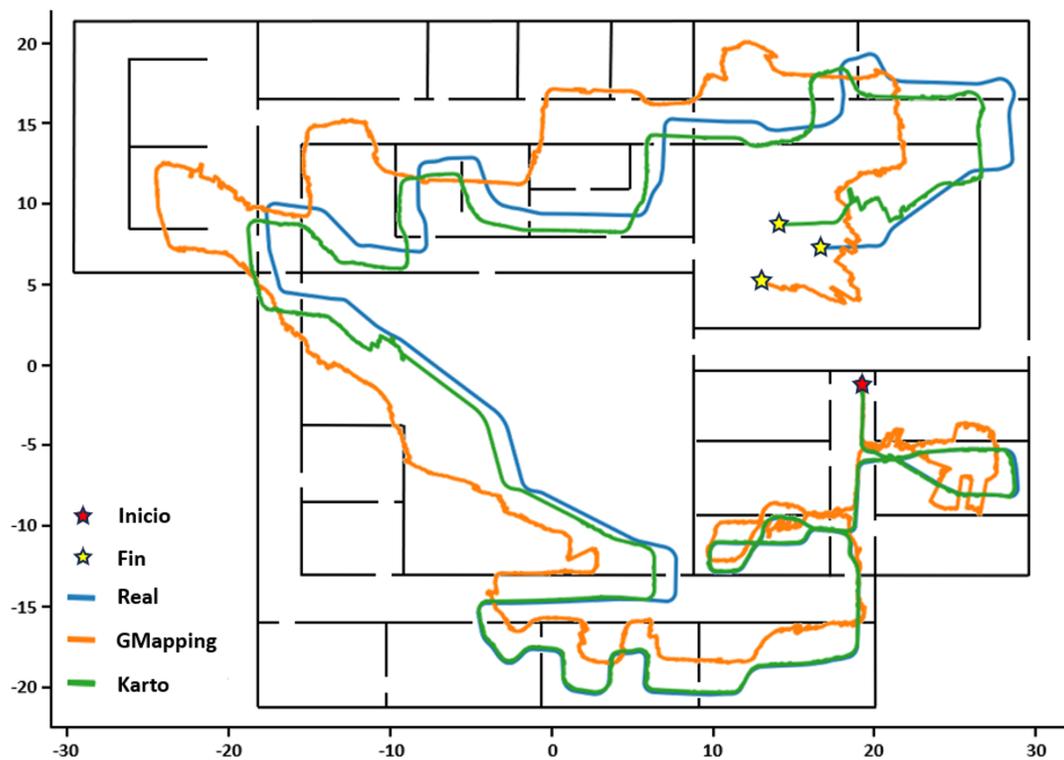


Figura 42. Recorrido de prueba #2, trayectoria real y estimadas por algoritmos de SLAM

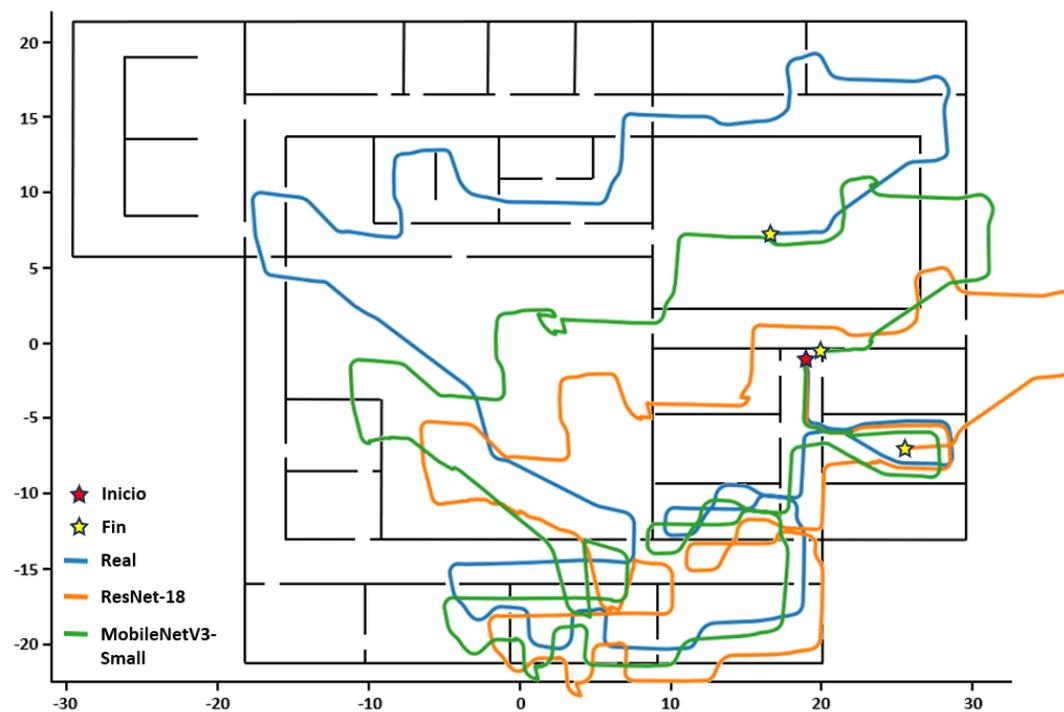


Figura 43. Recorrido de prueba #2, trayectoria real y estimadas por modelos entrenados

Como se observa en las tablas y figuras mencionadas, el desempeño de los modelos entrenados en el ambiente con obstáculos no fue tan bueno como en los recorridos de prueba en el espacio libre. Por ejemplo, ResNet-18 presentó un error RMSE máximo de 2.5739 para todos los recorridos de prueba en el espacio libre, mientras que en el ambiente con obstáculos el error RMSE mínimo fue de 4.0318. Conjeturamos que esto puede deberse a que, a pesar de que independientemente del tipo de ambiente los únicos datos utilizados son inerciales, en un ambiente con obstáculos los movimientos del robot tienen a ser más abruptos, presentando mayores cambios y con más frecuencia en la velocidad para evitar los obstáculos, cosa que probablemente no estuvo presente en los datos utilizados para el entrenamiento.

Por lo tanto, para mejorar el desempeño de los modelos, se creó un segundo ambiente con obstáculos, en el cual se realizaron 10 recorridos diferentes para aumentar la base de datos de entrenamiento, e incluir en esta los cambios abruptos de velocidad mencionados. Este ambiente se observa en la figura 44. Como resultado, se generaron 4.61 horas de nuevos datos que, al sumarlos a los datos existentes, se obtuvieron en total 15.93 horas de datos, o 229,314 ejemplos de entrenamiento.

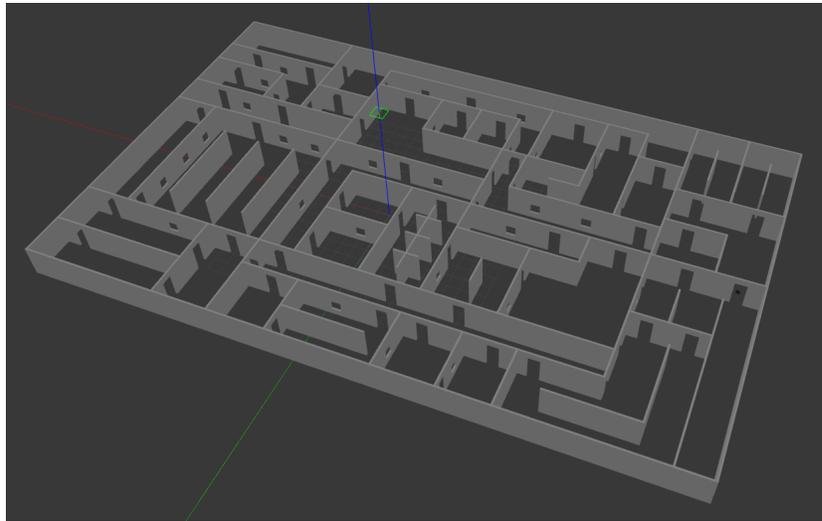


Figura 44. Ambiente con obstáculos utilizado para generar datos de entrenamiento adicionales.

Con estos datos, se entrenaron nuevamente los modelos comparados con los métodos de SLAM, y se realizaron pruebas en los dos recorridos de prueba #1 y #2. Los resultados de estos nuevos modelos se muestran en la tabla 16, y las trayectorias reconstruidas por los mismos en las figuras 45 y 46.

Tabla 16. Resultados para comparación con modelos mejorados en recorridos #1 y #2

	Prueba SLAM #1				Prueba SLAM #2			
	RMSE	RTE	MAEx	MAEy	RMSE	RTE	MAEx	MAEy
ResNet-18	0.3794	0.1743	0.2063	0.2611	0.5347	0.1883	0.2279	0.3251
MobileNetV3-Small	0.5389	0.2367	0.3755	0.2254	0.7691	0.2966	0.5934	0.2233

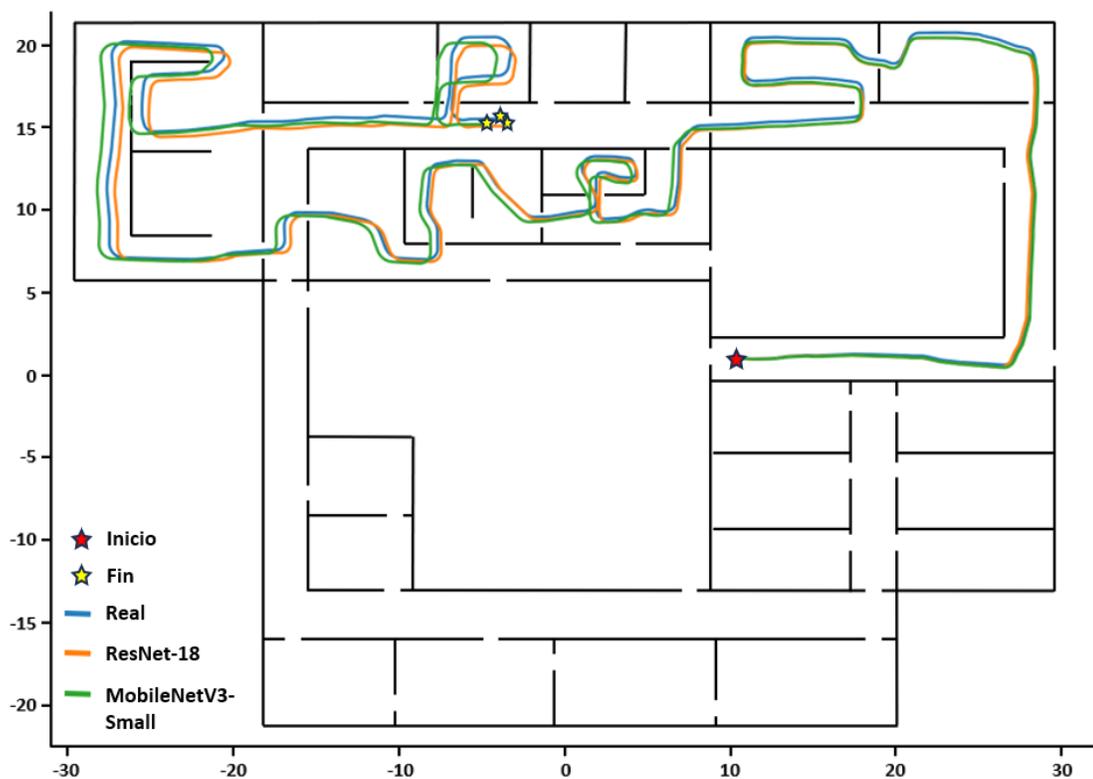


Figura 45. Recorrido de prueba #1, trayectoria real y estimadas por modelos entrenados con datos adicionales

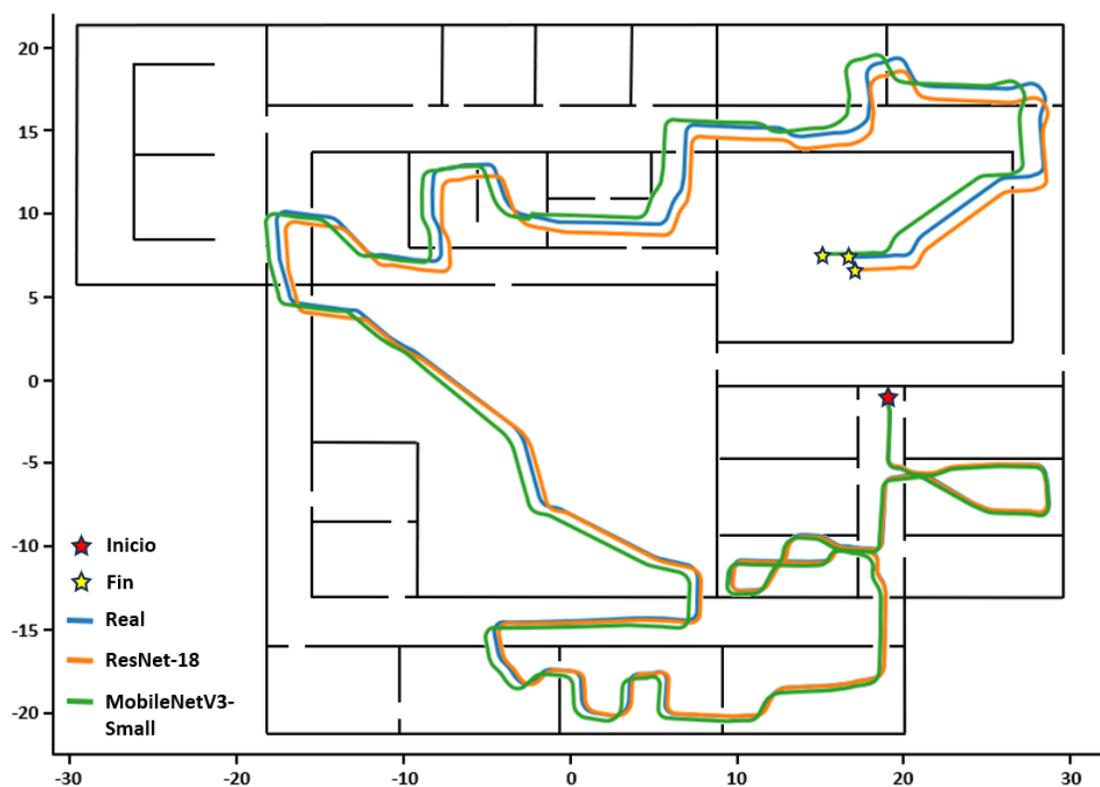


Figura 46. Recorrido de prueba #2, trayectoria real y estimadas por modelos entrenados con datos adicionales

De la tabla y figuras mencionadas en el párrafo anterior se puede concluir que entrenar los modelos nuevamente con datos adicionales generados en un ambiente con obstáculos mejoró considerablemente su desempeño, superando incluso a ambos métodos de SLAM. Para ResNet-18, hubo una reducción de 90.5898 % en el error RMS para el primer recorrido, y de 95.3626 % para el segundo recorrido. En el caso de MobileNetV3-Small, el error RMS se redujo en 82.4222 % para el primer recorrido y en 89.2109 % para el segundo recorrido. Al final, ResNet-18 tuvo el mejor desempeño en todas las métricas, con excepción del error absoluto en Y (MAEy), donde MobileNetV3-Small mostró mejores resultados. Comparando con los métodos de SLAM, *Karto* fue el que mostró inicialmente un mejor desempeño que los dos modelos probados en ambos recorridos. Después de entrenar los modelos nuevamente, ResNet-18 presentó un error RMS 72.8980 % menor que el de *Karto* para el recorrido de prueba #1 y 56.5461 % menor para el recorrido de prueba #2. Por su parte, MobileNetV3-Small mostró un error RMS 61.5043 % menor que *Karto* para el recorrido de prueba #1, y 37.4969 % menor para el recorrido de prueba #2.

5.3. Discusión

Durante las pruebas de este capítulo se logró un error RMS mínimo de 0.38 para los experimentos con obstáculos utilizando ResNet-18, como se observó en la sección 5.2, mostrando un mejor desempeño que los dos algoritmos de SLAM evaluados en los mismos recorridos. Este es un resultado notable, dado que los resultados obtenidos con los modelos entrenados utilizan datos de un solo sensor, el IMU montado sobre el robot, mientras que los algoritmos de SLAM utilizan como mínimo dos tipos de sensores diferentes, ya sean inerciales, encoders o sensores de rango láser, aunque existen también algoritmos de SLAM más modernos que hacen uso de cámaras los cuales pueden ser más precisos que los probados en este trabajo. Es importante notar que este resultado se obtuvo después de entrenar los modelos con datos adicionales capturados en un ambiente simulado con obstáculos, a pesar de que se pensaría que las aceleraciones y velocidades angulares del robot se comportarían de la misma manera en un ambiente con o sin obstáculos. Esto resalta la importancia de realizar la captura de datos para el entrenamiento del modelo en escenarios lo más cercanos posibles a los que se enfrentará el robot durante su funcionamiento normal.

El utilizar solamente los datos inerciales del robot tiene como ventaja sobre otro tipo de datos que es aplicable para cualquier tipo de ambiente, mientras que algoritmos que utilizan cámaras, por ejemplo, pueden fallar cuando no hay una iluminación adecuada, o los que utilizan sensores de rango láser pueden fallar cuando se encuentran con grandes espacios abiertos u obstáculos transparentes.

Los modelos de localización entrenados en este trabajo son aplicables solamente para el robot utilizado

durante la investigación. Si se quisiese utilizar con otro tipo de robot, deberá de ser entrenado con datos inerciales capturados de ese robot en específico, dado que cada robot tiene características de movimiento, aceleración y desaceleración diferentes. Una problema de investigación interesante consiste en evaluar si las arquitecturas analizadas tienen un desempeño similar con otros tipos de robot, por ejemplo, un robot tipo carro.

Otro de los puntos importantes a recalcar, es que todos los experimentos y los resultados fueron obtenidos en simulación. En un ambiente real pueden influir en el error de localización diferentes factores que no se tomaron en cuenta en la simulación, por ejemplo, las vibraciones debido a los motores del robot o la textura del suelo por el que se mueve, y los efectos que estas tendrían sobre los datos de aceleración y la estimación en la localización.

Capítulo 6. Conclusiones y trabajo futuro

Durante este trabajo se exploraron dos técnicas generales para resolver el problema de localización de robots móviles en interiores: empleando la intensidad de las señales Wi-Fi en el ambiente, y usando los datos inerciales del robot. Para el caso de localización utilizando señales Wi-Fi se exploraron tres métodos, *fingerprinting*, multilateración y multilateración con agrupamiento de K-medias, de los cuales *fingerprinting* mostró el mejor desempeño, utilizando KNN como modelo de localización. Sin embargo, los resultados no fueron lo suficientemente buenos para poder justificar el uso de un algoritmo que emplee únicamente la intensidad de las señales Wi-Fi del ambiente para realizar la localización.

Por su parte, se entrenaron siete modelos basados en redes neuronales profundas para estimar la localización del robot utilizando datos inerciales, mostrando en su mayoría un mejor desempeño que los métodos de localización con Wi-Fi. De todos los modelos entrenados, ResNet-18 y MobileNetV3-Small se seleccionaron para compararlos con dos algoritmos de SLAM, *GMapping* y *Karto*. Después de entrenar ambos modelos con datos adicionales generados en un ambiente con obstáculos, estos mostraron mejores resultados que los algoritmos de SLAM evaluados.

Al principio de la investigación se propuso el emplear un método de aprendizaje de máquina para fusionar los dos tipos de datos utilizados, intensidad de señales Wi-Fi y datos inerciales, basándose en que es posible corregir o reducir el error de deriva presente en la odometría inercial con la ayuda de un segundo sensor. Sin embargo, los resultados obtenidos por los métodos de localización Wi-Fi probados mostraron no ser lo suficientemente precisos para lograr esto. De haber fusionado ambos tipos de datos, el error presente de las señales Wi-Fi causaría que el error resultante en la localización fuera más grande que al utilizar solamente datos inerciales por sí solos.

En general, el utilizar únicamente datos inerciales para estimar la localización de un robot móvil en un ambiente interior resultó ser una buena solución, mostrando mejores resultados que los algoritmos de SLAM incluidos como parte de la plataforma robótica *TurtleBot3*. Además, un sensor inercial es aplicable para cualquier tipo de ambiente y cualquier tipo de robot.

6.1. Trabajo futuro

Dado que todos los experimentos presentados en este trabajo fueron realizados totalmente en simulación, como trabajo futuro se buscaría implementar los modelos y métodos aquí presentados, y cualquier mejora que pudiera desarrollarse, en un robot y ambientes reales, comparando los resultados obtenidos con los mostrados en este trabajo para experimentos en simulación.

Para el caso de la localización con señales Wi-Fi, podría explorarse el uso de datos adicionales de las señales además de la intensidad, como el tiempo de vuelo o el ángulo de llegada, los cuales podrían proporcionar información complementaria que ayude a mejorar la estimación en la posición del robot.

Además de las arquitecturas de aprendizaje profundo aquí mostradas, podrían analizarse otro tipo de redes neuronales, por ejemplo las redes *transformer*, utilizadas inicialmente para procesamiento del lenguaje natural. Nuevas variantes de las mismas tienen la posibilidad de procesar datos provenientes de series de tiempo, como el caso de los proporcionados por unidades de medición inercial.

Otro de los aspectos que no se estudió en esta investigación es el efecto que tiene la velocidad a la que se mueve el robot en el error de localización. Para esto, podría utilizarse otro tipo de robot que pueda moverse con mayor velocidad, ya que el robot empleado en este trabajo tiene una velocidad máxima de solamente 0.26 m/s.

Una de las formas de mejorar la estimación en la localización utilizando datos inerciales es realizando fusión con otro tipo de sensores. En general, se observó que a medida que el robot recorre el ambiente y el tiempo avanza, el error en la posición estimada va aumentando. Este error se podría corregir con la ayuda de un sensor más preciso, que típicamente tiene una frecuencia de muestreo mucho menor que la utilizada por el sensor inercial. Este sensor más preciso puede ser una cámara o un sensor de rango láser, y aunque estos fallen en situaciones específicas, la localización estimada utilizando solamente los datos inerciales es lo suficientemente buena como para ignorar los datos de estos sensores en los escenarios en los que tienden a fallar.

Literatura citada

- Abolfazli Esfahani, M., Wang, H., Wu, K., & Yuan, S. (2020). AbolDeepIO: A Novel Deep Inertial Odometry Network for Autonomous Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 21(5), 1941–1950. <https://doi.org/10.1109/TITS.2019.2909064>.
- Arun, A., Ayyalasomayajula, R., Hunter, W., & Bharadia, D. (2022). P2SLAM: Bearing Based WiFi SLAM for Indoor Robots. *IEEE Robotics and Automation Letters*, 7(2), 3326–3333. <https://doi.org/10.1109/LRA.2022.3144796>.
- Asraf, O., Shama, F., & Klein, I. (2022). PDRNet: A Deep-Learning Pedestrian Dead Reckoning Framework. *IEEE Sensors Journal*, 22(6), 4932–4939. <https://doi.org/10.1109/JSEN.2021.3066840>.
- Atia, M. M., Korenberg, M., & Noureldin, A. (2012). A WiFi-aided reduced inertial sensors-based navigation system with fast embedded implementation of particle filtering. *2012 8th International Symposium on Mechatronics and its Applications, ISMA 2012*. <https://doi.org/10.1109/ISMA.2012.6215167>.
- Ayyalasomayajula, R., Arun, A., Wu, C., Sharma, S., Sethi, A. R., Vasisht, D., & Bharadia, D. (2020). Deep learning based wireless localization for indoor navigation. *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, 214–227. <https://doi.org/10.1145/3372224.3380894>.
- Azini, A. S., Kamarudin, M. R., & Jusoh, M. (2016). Transparent antenna for WiFi application: RSSI and throughput performances at ISM 2.4 GHz. *Telecommunication Systems*, 61(3), 569–577. <https://doi.org/10.1007/s11235-015-0013-x>.
- Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2), 157–166. <https://doi.org/10.1109/72.279181>.
- Brossard, M. & Bonnabel, S. (2019). Learning wheel odometry and imu errors for localization. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May, 291–297. <https://doi.org/10.1109/ICRA.2019.8794237>.
- Campos, C., Elvira, R., Rodriguez, J. J., Montiel, J. M., & Tardos, J. D. (2021). ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. *IEEE Transactions on Robotics*, 37(6), 1874–1890. <https://doi.org/10.1109/TR0.2021.3075644>.
- Chen, C., Lu, X., Markham, A., & Trigoni, N. (2018). IoNet: Learning to cure the curse of drift in inertial odometry. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 6468–6476. <https://doi.org/10.1609/aaai.v32i1.12102>.
- Cui, W., Liu, Q., Zhang, L., Wang, H., Lu, X., & Li, J. (2020). A robust mobile robot indoor positioning system based on Wi-Fi. *International Journal of Advanced Robotic Systems*, 17(1), 1–10. <https://doi.org/10.1177/1729881419896660>.
- D'avella, S., Unetti, M., & Tripicchio, P. (2022). RFID Gazebo-Based Simulator With RSSI and Phase Signals for UHF Tags Localization and Tracking. *IEEE Access*, 10(January), 22150–22160. <https://doi.org/10.1109/ACCESS.2022.3152199>.
- Dobrev, Y., Gulden, P., & Vossiek, M. (2018). An Indoor Positioning System Based on Wireless Range and Angle Measurements Assisted by Multi-Modal Sensor Fusion for Service Robot Applications. *IEEE Access*, 6, 69036–69052. <https://doi.org/10.1109/ACCESS.2018.2879029>.

- El Khaled, Z., Ajib, W., & McHeick, H. (2022). Log Distance Path Loss Model: Application and Improvement for Sub 5 GHz Rural Fixed Wireless Networks. *IEEE Access*, 10, 52020–52029. <https://doi.org/10.1109/ACCESS.2022.3166895>.
- Etter-Olguín, J., Duran-Faundez, C., Rohten, J., Seguel-Cárdenas, R., & Santana, I. (2019). Simulation of RSSI-based positioning algorithms for wireless networks using ns-3. *XVIII Simposio de Ingeniería Eléctrica SIE-2019*. https://www.researchgate.net/publication/338409586_Simulation_of_RSSI-based_positioning_algorithms_for_wireless_networks_using_ns-3.
- Gerkey, B. (2019). slam_karto. ROS. Recuperado el 12 de agosto de 2023 de https://wiki.ros.org/slam_karto.
- Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2019). *Fundamental Concepts of Convolutional Neural Network*, (pp. 519–567). Springer International Publishing, Cham.
- Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1), 34–46. <https://doi.org/10.1109/TR0.2006.889486>.
- Hashemifar, Z. S., Adhivarahan, C., Balakrishnan, A., & Dantu, K. (2019). Augmenting visual SLAM with Wi-Fi sensing for indoor applications. *Autonomous Robots*, 43(8), 2245–2260. <https://doi.org/10.1007/s10514-019-09874-z>.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem, 770–778. <https://doi.org/10.1109/CVPR.2016.90>.
- Herath, S., Yan, H., & Furukawa, Y. (2020). RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, New Methods. *Proceedings - IEEE International Conference on Robotics and Automation*, 3146–3152. <https://doi.org/10.1109/ICRA40945.2020.9196860>.
- Howard, A., Pang, R., Adam, H., Le, Q., Sandler, M., Chen, B., Wang, W., Chen, L.-C., Tan, M., Chu, G., Vasudevan, V., & Zhu, Y. (2019). Searching for mobilenetv3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1314–1324. <https://doi.org/10.1109/ICCV.2019.00140>.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. <https://doi.org/10.48550/arXiv.1704.04861>.
- Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7132–7141. <https://doi.org/10.1109/CVPR.2018.00745>.
- Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In Bach, F. & Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 2342–2350, Lille, France. PMLR. <https://proceedings.mlr.press/v37/jozefowicz15.html>.
- Labbé, Mathieu and Michaud, F. (2013). Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation. *IEEE Transactions on Robotics*, 29(3), 734–745. <https://doi.org/10.1109/TR0.2013.2242375>.
- Li, C., Wang, S., Zhuang, Y., & Yan, F. (2021). Deep Sensor Fusion between 2D Laser Scanner and IMU for Mobile Robot Localization. *IEEE Sensors Journal*, 21(6), 8501–8509. <https://doi.org/10.1109/JSEN.2019.2910826>.

- Lima, J. P. S. D. M., Uchiyama, H., & Taniguchi, R. I. (2019). End-to-end learning framework for IMU-based 6-DOF odometry. *Sensors (Switzerland)*, 19(17), 1–16. <https://doi.org/10.3390/s19173777>.
- Liu, W., Caruso, D., Ilg, E., Dong, J., Mourikis, A. I., Daniilidis, K., Kumar, V., & Engel, J. (2020). TLIO: Tight Learned Inertial Odometry. *IEEE Robotics and Automation Letters*, 5(4), 5653–5660. <https://doi.org/10.1109/LRA.2020.3007421>.
- Liu, Y., Shao, Z., Teng, Y., & Hoffmann, N. (2021). NAM: Normalization-based Attention Module. <https://doi.org/10.48550/arXiv.2111.12419>.
- Luo, Q., Yang, K., Yan, X., Li, J., Wang, C., & Zhou, Z. (2022). An Improved Trilateration Positioning Algorithm with Anchor Node Combination and K-Means Clustering. *Sensors*, 22(16). <https://doi.org/10.3390/s22166085>.
- Open Robotics (2023). About Gazebo. *Gazebo*. Recuperado el 12 de agosto de 2023 de <https://gazebo.org/>.
- Pfeiffer, M., Shukla, S., Turchetta, M., Cadena, C., Krause, A., Siegwart, R., & Nieto, J. (2018). Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Mapless Navigation by Leveraging Prior Demonstrations. *IEEE Robotics and Automation Letters*, 3(4), 4423–4430. <https://doi.org/10.1109/LRA.2018.2869644>.
- ROBOTIS (2021). Turtlebot3. *ROBOTIS e-manual*. Recuperado el 12 de agosto de 2023 de <https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/>.
- Rodriguez-Losada, D., San Segundo, P., Matia, F., Galan, R., Jiménez, A., & Pedraza, L. (2007). FastSLAM: A factored solution to the simultaneous localization and mapping problem. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 6(PART 1), 542–547. <https://doi.org/10.3182/20070903-3-fr-2921.00092>.
- Roy, P. & Chowdhury, C. (2021). A Survey of Machine Learning Techniques for Indoor Localization and Navigation Systems. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 101(3). <https://doi.org/10.1007/s10846-021-01327-z>.
- Ruan, X., Ren, D., Zhu, X., & Huang, J. (2019). Mobile Robot Navigation based on Deep Reinforcement Learning. *Proceedings of the 31st Chinese Control and Decision Conference, CCDC 2019*, 6174–6178. <https://doi.org/10.1109/CCDC.2019.8832393>.
- Russell, S. & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*, (3a ed.). Prentice Hall.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *IEEE Conference on Computer Vision and Pattern Recognition*. <https://doi.org/10.48550/arXiv.1801.04381>.
- Seeger, M. (2004). Gaussian processes for machine learning. *International journal of neural systems*, 14(2), 69–106. <https://doi.org/10.1142/S0129065704001899>.
- Thrun, S. & Montemerlo, M. (2006). The graph SLAM algorithm with applications to large-scale mapping of urban structures. *International Journal of Robotics Research*, 25(5-6), 403–429. <https://doi.org/10.1177/0278364906065387>.
- Wang, J. & Park, J. G. (2021). An enhanced indoor positioning algorithm based on fingerprint using fine-grained csi and rssi measurements of ieee 802.11n wlan. *Sensors*, 21(8). <https://doi.org/10.3390/s21082769>.

- Wang, L., Shang, S., & Wu, Z. (2023a). Research on Indoor 3D Positioning Algorithm Based on WiFi Fingerprint. *Sensors*, 23(1). <https://doi.org/10.3390/s23010153>.
- Wang, Y., Kuang, J., Niu, X., & Liu, J. (2023b). LLIO: Lightweight Learned Inertial Odometer. *IEEE Internet of Things Journal*, 10(3), 2508–2518. <https://doi.org/10.1109/JIOT.2022.3214087>.
- Wen, S., Zhao, Y., Yuan, X., Wang, Z., Zhang, D., & Manfredi, L. (2020). Path planning for active SLAM based on deep reinforcement learning under unknown environments. *Intelligent Service Robotics*, 13(2), 263–272. <https://doi.org/10.1007/s11370-019-00310-w>.
- Woo, S., Park, J., Lee, J. Y., & Kweon, I. S. (2018). CBAM: Convolutional block attention module. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11211 LNCS, 3–19. https://doi.org/10.1007/978-3-030-01234-2_1.
- Xiao, Y., Ou, Y., & Feng, W. (2018). Localization of indoor robot based on particle filter with EKF proposal distribution. *2017 IEEE International Conference on Cybernetics and Intelligent Systems, CIS 2017 and IEEE Conference on Robotics, Automation and Mechatronics, RAM 2017 - Proceedings, 2018-Janua*, 568–571. <https://doi.org/10.1109/ICCIS.2017.8274839>.
- Yuan, Y. & Shaw, M. J. (1995). Induction of fuzzy decision trees. *Fuzzy Sets and Systems*, 69(2), 125–139. [https://doi.org/10.1016/0165-0114\(94\)00229-Z](https://doi.org/10.1016/0165-0114(94)00229-Z).
- Zhang, L., Chen, Z., Cui, W., Li, B., Chen, C., Cao, Z., & Gao, K. (2020). WiFi-Based Indoor Robot Positioning Using Deep Fuzzy Forests. *IEEE Internet of Things Journal*, 7(11), 10773–10781. <https://doi.org/10.1109/JIOT.2020.2986685>.
- Zhu, Y., Zhang, J., Zhu, Y., Zhang, B., & Ma, W. (2023). RBCN-Net: A Data-Driven Inertial Navigation Algorithm for Pedestrians. *Applied Sciences (Switzerland)*, 13(5). <https://doi.org/10.3390/app13052969>.
- Zou, H., Chen, C. L., Li, M., Yang, J., Zhou, Y., Xie, L., & Spanos, C. J. (2020). Adversarial Learning-Enabled Automatic WiFi Indoor Radio Map Construction and Adaptation with Mobile Robot. *IEEE Internet of Things Journal*, 7(8), 6946–6954. <https://doi.org/10.1109/JIOT.2020.2979413>.