Universidad Internacional de La Rioja (UNIR)

Escuela de Ingeniería

Máster Universitario en Inteligencia Artificial

Yielding Associations Network: Un sistema para mejorar la localización semántica

Trabajo Fin de Máster

Presentado por: Bello Méndez, Yan

Director: Dr. Cobos Guzman, Salvador

Ciudad: Madrid, España
Fecha: 27 de febrero de 2019

Contenido

Índice de Ecuaciones

Índice de Figuras

Índice de Tablas

Resumen

En el ámbito de la *Internet of Things* (IoT), el *Edge Computing* aporta un enfoque para la computación localizada que permite optimizar el almacenamiento de los datos y agilizar la toma de decisiones. Este trabajo tiene como objetivo estudiar el desempeño de varios algoritmos de Inteligencia Artificial para la localización de un robot móvil. La metodología se ha basado en un prototipo experimental para recoger datos de sensores acoplados a una Raspberry Pi 3B+. Con estos datos se han desarrollado y probado varios modelos de aprendizaje automático que han sido entrenados en la nube usando Google Colab. Los modelos se han validado desplegándolos en el prototipo desarrollado. Los resultados incluyen la comparativa de estos modelos y una propuesta novedosa de "*Yielding Associations Network*" como un sistema para mejorar la localización semántica. Las conclusiones destacan el gran valor de la aplicación de la Inteligencia Artificial para robótica móvil y *Edge Computing*.

Palabras Clave: aprendizaje automático, Edge Computing, localización semántica, redes neuronales artificiales, robótica móvil, Support Vector Machines

Abstract

In the field of the Internet of Things (IoT), Edge Computing provides an approach to localized computing that optimizes data storage and streamlines decision making. This work aims to study the performance of several artificial intelligence algorithms used in the automatic localization of a mobile robot. The methodology has been based on an experimental prototype, with which data from sensors coupled to a Raspberry Pi 3B+ have been collected. With this data, several machine learning models that have been trained in the cloud using Google Colab have been developed and tested. The operation of these models has been validated by deploying them in the developed prototype. The results include the comparison of the algorithms and a novel proposal of "Yielding Associations Network" as a system that allows to improve semantic localization. The conclusions highlight the great value of the application of Artificial Intelligence for mobile robotics and Edge Computing.

Keywords: artificial neural network, Edge Computing, machine learning, mobile robotics, semantic localization, Support Vector Machines

Capítulo 1. Introducción

Con el auge de los dispositivos conectados y la Internet de las Cosas (IoT, *Internet of Things*), el volumen de datos generados y almacenados en servidores en la nube crece a un ritmo imparable. Esta situación presenta varios retos técnicos y de gestión, tanto para aplicaciones industriales – como las enmarcadas en la llamada Industria 4.0 o la Industrial IoT (IioT) –, como para soluciones corporativas, así como para aplicaciones de servicios o productos de uso doméstico. Una de las tecnologías que ha emergido en los últimos años y que pretende afrontar este reto es la llamada computación en el borde (en inglés: *Edge Computing*) que aporta un enfoque para el procesamiento local de los datos, de forma más cercana o próxima a los dispositivos que generan y capturan dichos datos, permitiendo optimizar el uso de estos datos, agilizando la toma de decisiones y permitiendo reducir el volumen de datos enviados a la nube para su almacenamiento centralizado, si así se desea.

Actualmente existen un gran número de dispositivos que se conectan y que pueden enviar datos a servidores en la nube: desde relojes y teléfonos inteligentes, cascos sensorizados e inteligentes (ej. Livall BH55M), patinetes eléctricos (ej. Segway Loomo), a robots domésticos (ej. iRobot Roomba), así como diversos dispositivos (*"gadgets"*) de asistencia personal en el hogar u oficina (ej. Amazon Alexa, Google Home, Amazon Dash Button, etc.).

Figura 1. Productos y servicios innovadores recientes basados en robótica móvil.
Fuentes: https://blog.aboutamazon.com/transportation/meet-scout y https://www.segwayrobotics.com/

De los millones de dispositivos conectados hoy en día, el reto de la gestión de datos es especialmente relevante y de gran interés en el caso de los agentes móviles (robots móviles), sean de uso industrial, doméstico o de servicio. Este interés se debe especialmente al gran volumen de datos que se pueden obtener y recoger a través de los diversos sensores que estos agentes incorporan y que resultan importantes para poder desempeñar varias funciones básicas como la navegación, localización, reconocimiento del entorno, entre otras. Cabe resaltar que algunos de los productos y servicios más innovadores lanzados al mercado en los últimos años incorporan agentes robóticos móviles y prestaciones basadas en técnicas de Inteligencia Artificial como parte de sus propuestas de valor. Ejemplos destacables recientes

incluyen Segway Loomo – solución de movilidad personal con modo robótico – y Amazon Scout – el robot presentado hace apenas unos días por Amazon como una promesa de futuro en el reparto de paquetes. Ver Figura 1.

Otro reto relacionado es la capacidad de cálculo o cómputo que pueda disponer o requerir este tipo de sistemas, ya sea centralizada, distribuida o mixta. Es decir, las implicaciones resultantes de realizar los cálculos y procesamiento de datos de forma cercana o remota al dispositivo donde se usan los resultados de dicha computación. En estos escenarios de uso, dónde se realizan los cálculos, o a tal efecto la proximidad a los servidores donde se realicen también es relevante por razones de latencia, o por una eventual falta de conexión a Internet que imposibilite el funcionamiento normal del sistema o dispositivo en el borde.

Diversos trabajos han estudiado retos, oportunidades y varias casuísticas de uso del *Edge Computing* planteando propuestas de carácter general como por ejemplo una gestión oportunista de los recursos conectados y de los servidores cercanos-próximos (Olaniyan, Fadahunsi, Maheswaran y Zhani, 2018), o aplicando técnicas de Inteligencia Artificial como el uso de aprendizaje profundo (*Deep Learning*) para la gestión y transferencia de tareas costosas de computación de dispositivos conectados a servidores locales distribuidos (Huang, Feng, Feng, Huang y Qian, 2018), así como el uso de redes neuronales recurrentes LSTM (*Long Short Term Memory*) para la detección y gestión de fallos en redes de Edge Computing (Park, Kim, An y Jung, 2018).

Por otra parte, el auge de los ordenadores de placa única (SBC, *Single Board Computer*) como los varios modelos y versiones de las Raspberry Pi, han traspasado en los últimos años sus fines iniciales de carácter educativo y promoción de la enseñanza de las ciencias de la computación para abrir nuevas vías de experimentación, desarrollo de prototipos e incluso productos finales con un destacable bajo coste de adquisición y desarrollo. Como explican Johnston y Cox (2017), la popularidad y la gran demanda de SBC's para aplicaciones de sistemas embebidos (*embedded systems*) y el desarrollo de la IoT han beneficiado a la Raspberry Pi, hasta el punto de convertirla en una opción viable para desplegar computación en el borde o en la niebla (*Fog Computing*).

No obstante, el uso de este tipo de SPC no está exento de retos y problemas, como pueden ser escoger o decidir qué sistema operativo y qué configuración de periféricos utilizar, cómo conectar con sensores y otros dispositivos relevantes, así como el impacto que pueden tener las limitaciones técnicas de capacidad de cómputo, de memoria o almacenamiento de los sistemas basados en estas SBC en las soluciones que se diseñen, desarrollen y desplieguen usando estos dispositivos.

Esta problemática es aún más relevante si consideramos que precisamente una característica común del entrenamiento de algunos algoritmos de Inteligencia Artificial es la alta demanda de capacidad de cómputo, y que son precisamente estas técnicas y algoritmos de Inteligencia Artificial las que están llamadas a aportar gran valor en estas soluciones por su capacidad para automatizar diversas tareas con soluciones inteligentes (ej. reconocimiento y procesamiento de imágenes, voz, clasificación, navegación, etc.).

Por tanto, el problema que se plantea y que se pretende tratar en este trabajo requiere la integración efectiva de todos estos múltiples componentes (ordenadores de placa única, sensores láser, procesos de recogida y tratamiento de datos – "próximos", en el borde/Edge y en la nube -, algoritmos de aprendizaje automático, etc.) para el desarrollo de un sistema de localización de un agente robótico móvil basado en Inteligencia Artificial. En particular, se busca explorar y entender cómo aplicar varias técnicas o herramientas de Inteligencia Artificial en el contexto de los recursos limitados de un sistema de *Edge Computing* para resolver problemas prácticos como la localización semántica de un agente robótico móvil en un entorno interior, usando exclusivamente de datos de distancia obtenidos a través de un sensor láser de detección de rangos (Laser Range Finder, LRF).

Aunque el problema de localización de un robot móvil se ha abordado desde hace tiempo con diversas técnicas, incluyendo el uso de filtros de Kalman y de partículas en algoritmos SLAM (*Simultaneous Localization And Mapping*) (Piardi, Lima y Costa, 2018), así como técnicas de aprendizaje profundo *(Deep Learning*) con imágenes como puntos de referencias (*landmarks*) para la localización semántica (Atanasov, Zhu, Daniilidis, y Pappas, 2016; Fu, Atanasov, Topcu y Pappas, 2016), sigue siendo un campo activo de investigación y desarrollo de aplicaciones. En el contexto del *Edge Computing* y de los entornos con recursos limitados, la localización de un agente robótico presenta una problemática adicional, acentuada por las limitaciones explicadas anteriormente (capacidad de cómputo, almacenamiento, etc.).

Una posible solución o enfoque para tratar estos retos es aplicar las técnicas y algoritmos de Inteligencia Artificial de aprendizaje automático para entrenar y desplegar un clasificador que utilice la "percepción" del agente robótico de manera óptima. Por ejemplo, haciendo un uso óptimo de los datos de un número limitado de sensores. Si, además, se considera y aborda adecuadamente el ciclo de vida completo de la gestión de los datos, desde la definición de cómo debe realizarse su captura inicial y limpieza, hasta su procesamiento, modelado o incluso cómo se llevará a cabo su explotación en la toma de decisiones y su eventual almacenamiento, se podrá plantear un proyecto con un enfoque integral-sistémico que no sólo contemple la solución técnica, sino que también contemple el contexto de despliegue de dicha solución.

Desde este punto de vista contextual, la localización semántica también puede aportar gran valor facilitando una comunicación más humanista en la interacción entre las personas y las máquinas (agentes robóticos). Por ejemplo, aun cuando sea infinitamente más precisa la ubicación física de un robot de servicio doméstico, indicada con exactitud milimétrica en términos de sus coordenadas (x, y), seguramente para una persona se encuentre en esta vivienda sería suficiente, y mucho más informativo, que el robot comunique su ubicación indicando "…*estoy en la habitación de las niñas*" o "*… en el salón, justo al lado del sofá*".

En cualquier caso, la solución planteada en este trabajo no pretende ser una propuesta excluyente a otras soluciones o técnicas. Más bien, puede ser considerada complementaria a otros enfoques como, por ejemplo, los que utilizan imágenes o información de odometría. Así mismo, pueden plantearse casos de uso, donde la aplicación de un sensor láser puede evitar problemas éticos o percepción de intrusión, si en su lugar se usase un sistema basado en imágenes. Por ejemplo, para muchas personas representaría una amenaza o riesgos a sus derechos de privacidad, si un sistema robótico que efectúe tareas de limpieza y sanitarias en espacios o baños públicos basase su localización en la captura y uso de videos e imágenes.

Este trabajo tiene como objetivo principal realizar un estudio comparativo de varios algoritmos de Inteligencia Artificial para la localización semántica de un agente robótico móvil, mediante técnicas de aprendizaje automático supervisado considerando el ciclo de vida completo de la gestión de datos en un entorno de recursos limitados de *Edge Computing*. Como resultado de este estudio se ha podido plantear una propuesta novedosa mediante el uso de un grafo topológico semántico que mejora el desempeño de los algoritmos estudiados. En el desarrollo de este trabajo ha jugado un papel esencial el enfoque frecuentemente utilizado en proyectos basados en Inteligencia Artificial donde se exploran, parametrizan, optimizan y comparan múltiples modelos para aportar una solución lo más adecuada posible al problema planteado.

Por otra parte, cada vez es más frecuente que los sistemas y soluciones planteados usando técnicas de Inteligencia Artificial no tengan un único componente central o algoritmo basado en estas técnicas. Por lo que existen numerosas propuestas actuales (Wang et al., 2018; Bello, 2019) que integran varios modelos de Inteligencia Artificial dentro de una misma solución, incorporando lo que ha sido considerado como meta-algoritmos (Smolyakov, 2017) de "*ensemble learning*" o aprendizaje combinado porque literalmente combinan estos distintos modelos en busca de una mejora en el desempeño de las tareas de aprendizaje automático. En este sentido, el trabajo desarrollado y presentado a continuación, también se asoma al prometedor ámbito de las sinergias entre técnicas y modelos, proponiendo un sistema que mejora la localización de un agente robótico móvil, mediante el uso combinado de

clasificadores automáticos (ej. una Máquina de Vector de Soporte) y mapas o grafos topológico-semánticos.

Aunque en este trabajo no se pretende aportar con esta propuesta “la solución” (única o idónea, si existiese) para el problema planteado, el enfoque sistémico con el que se ha realizado ejemplifica cómo las técnicas de Inteligencia Artificial y la disciplina de la ciencia del dato (*Data Science*) pueden ayudar y aportar gran valor a problemas reales como los retos planteados de *Edge Computing* y robótica móvil. De esta forma, se ha demostrado de manera experimental la implementación de varios algoritmos de Inteligencia Artificial en un dispositivo real como prueba de concepto del sistema computacional diseñado para el agente robótico.

Las contribuciones y resultados principales de este trabajo son:

1. Propuesta de diseño de “prueba de concepto” (prototipo): Integrando hardware (Raspberry PI y sensor LRF Hokuyo) y software (algoritmos de IA en Python), así como el flujo de trabajo de captura de datos, modelado, entrenamientos-pruebas y despliegue de modelos de Inteligencia Artificial.
2. Comparativa y análisis de los resultados del uso de varios modelos basados en Inteligencia Artificial en varios experimentos para el problema de localización semántica de un agente robótico móvil usando exclusivamente datos de un sensor láser (LRF) y algoritmos de aprendizaje automático supervisado.
3. Propuesta de un sistema de localización semántica con un Filtro de Contexto. Este filtro funciona como un operador que usa el conocimiento del entorno que tiene el agente robótico móvil representado como un grafo topológico-semántico o una matriz, y permite mejorar la predicción de localización semántica del agente haciendo uso exclusivo de los datos de un sensor láser y la ubicación anterior del agente.
4. Guía de implementación del enfoque/método sistémico propuesto: consistente en una lista de comprobación tipo *checklist* de aspectos que se deben tener en cuenta en cada una de las fases de un proyecto de este tipo.

El trabajo se organiza con la siguiente estructura. En el Capítulo 2 se presenta el estado del arte donde se describen trabajos previos relevantes e investigaciones relacionadas, en el Capítulo 3 se presentan los objetivos que se desea alcanzar y la metodología a utilizar, en el Capítulo 4 se incluye el desarrollo de las contribuciones, destacando una propuesta novedosa de sistema de localización semántica, el uso del prototipo creado y los resultados de los numerosos experimentos realizados con los algoritmos y modelos basados en técnicas de Inteligencia Artificial. Finalmente, en el Capítulo 5 se debaten las conclusiones, lecciones aprendidas y se exploran posibles líneas de investigación futura.

Capítulo 2. Estado del arte y situación actual

En este capítulo se presentan los resultados de una revisión detallada y exhaustiva de trabajos previos, referencias e investigaciones que consideramos que tienen especial relevancia para el trabajo realizado.

2.1. Retos y oportunidades de la Internet de las Cosas (IoT, *Internet of Things*)

Desde hace años, tanto la comunidad científica como profesionales de la industria y los servicios mantienen un foco de atención en temas relacionados con el impacto, oportunidades y retos que presenta el auge de los dispositivos conectados a Internet. La experiencia profesional del autor en este ámbito no hace más que confirmar el interés y vigencia de estos temas. Como él pudo comprobar durante su participación en los congresos Advanced Factories 2018 y Global Robot Expo 2018 celebrados en Barcelona y Madrid respectivamente, entre los temas tratados con mayor recurrencia y protagonismo destacaron precisamente el impacto de la IoT en la robótica, la Inteligencia Artificial, la gestión de datos (en la nube y en los dispositivos) y las novedades del Edge Computing, entre otras. La previsión de agenda para estos eventos en 2019 es similar (Advanced Factories, 2019; Global Robot Expo 2019).

Los retos y oportunidades de la IoT han sido tratados desde distintas perspectivas y disciplinas, incluyendo la gestión de los datos. En el Simposio sobre "*Colossal Data Analysis and Networking*", Gupta y Gupta (2016) destacaban la importancia de los avances alcanzados y los retos pendientes que se muestran en la Tabla 1.

Tabla 1. Retos y avances en IoT

Avances destacados:	Retos principales:
• Tecnología de sensores • **Robótica e Inteligencia Artificial** • Comunicaciones • Computación en la nube y *Machine Intelligence* • Información ubicua • Percepción del contexto	• Sensores • Conexiones a redes • Privacidad y seguridad • Almacenamiento en la web y recuperación de datos • **Independencia de los humanos** • Escalabilidad e identidad de objetos

Fuente: adaptado de Gupta y Gupta (2016)

Actualmente, varios de estos retos motivan áreas de investigación activa e interés de aplicación en la industria. En este contexto, el uso de técnicas de Inteligencia Artificial para

dotar a agentes robóticos de mayor autonomía en la toma de decisiones y generar comportamientos más adaptativos es uno de los aspectos clave del presente trabajo.

2.2. Recursos para una investigación "low-cost" de altas prestaciones

Como se indicaba en la introducción, en los últimos años algunos recursos como las Raspberry Pi, han pasado de ser una herramienta de apoyo a la educación en tecnologías de la información, para convertirse en elementos centrales de una potente plataforma de experimentación (Johnston y Cox, 2017).

De forma paralela, otros dispositivos y sensores relacionados se han popularizado por el aumento en la demanda de aplicaciones de IoT, un hecho que se ha acelerado con la bajada de costes de muchos de estos recursos. Por ejemplo, el mismo sensor utilizado en el presente trabajo, el Hokuyo URG-04LX, con un coste de mercado en 2018-2019 de alrededor de 1.000 euros (RobotShop, 2019), ha sido objeto de estudio detallado y propuesto como elemento sensorial central para un sistema low-cost de localización de referencia ("*Ground Truth*") para robots móviles con ruedas (Piardi et al., 2018). Piardi et al. (2018) observan que "*la tecnología de los escáneres basados en láser ha aportado la habilidad de desarrollar aplicaciones para la medición sin contacto, precisa y rápida en un amplio rango de aplicaciones*". En dicho trabajo, se propone el uso del sensor LRF Hokuyo para detectar formas geométricas circulares en dos situaciones: en primer lugar, como un objeto estático – que pudiera servir como punto de referencia, y en segundo lugar como un objeto móvil – que pudiera ser el propio robot percibido por un LRF desde una posición fija en una estancia, habitación o espacio determinado.

Si bien la contribución de Piardi et al. (2018) se centra más en aportar una estimación detallada de referencia *"ground truth"* de la ubicación de dichas formas geométricas para facilitar la localización del agente robótico móvil; y nuestro trabajo, en cambio, está más focalizado en la localización semántica, el uso del mismo sensor y los resultados por ellos obtenidos han servido también como confirmación de estar utilizando en este trabajo recursos que son relevantes y de aplicación para otras investigaciones y trabajos.

Este y otros trabajos debatidos a continuación permiten confirmar que, aunque existen diversas técnicas y algoritmos para la localización de un agente robótico móvil en un espacio interior, el uso de sensores láser (LRF) ha ganado popularidad y aceptación como opción económicamente viable y de buenas prestaciones técnicas.

Por otra parte, el uso de la Raspberry Pi también se ha adoptado en proyectos de investigación sobre sistemas de control para vehículos autónomos (Berntorp, Hoang, Quirynen y Di Cairano, 2018). Como explican Berntorp et al. (2018), los prototipos que han usado para probar sus modelos de control se han basado en la plataforma Hamster, un vehículo robótico que incorpora dos Raspberry Pi 3B+ y otros sensores a escala, que tienen entre otras ventajas un menor coste que hacer dichas pruebas y experimentos en prototipos y modelos a tamaño real.

2.3. Sobre SLAM y localización semántica de robots móviles

El problema general de localización y mapeo simultáneo (SLAM, *Simultaneous Localization And Mapping*) es de crucial importancia para numerosas aplicaciones de robótica móvil y especialmente relevante, si se desea dotar de autonomía a un agente robótico móvil para que pueda navegar en el entorno. La localización en este contexto puede ser planteada como la capacidad de un agente o robot de determinar su posición en un entorno usando información de diversa naturaleza (ej. sensores, mapas, puntos de interés, etc.). Así, existen numerosos trabajos previos que han desarrollado y planteado alternativas para dar respuesta a diversos retos relacionados con la localización, haciendo uso e integrando diversas técnicas (matrices de ocupación, redes neuronales, etc.) y atendiendo a distintas asunciones de trabajo: entorno simplificado/discreto en 2D, entorno percibido en 3D con video, etc. A continuación, se relatan varios trabajos previos, que destacan por el uso novedoso de técnicas de Inteligencia Artificial.

Förster, Graves y Schmidhuber (2007) plantean una propuesta que consiste en realizar una localización eficiente de un robot usando redes neuronales recurrentes (RNN) de tipo LSTM (*Long Short Term Memory*). En dicho trabajo, los autores utilizan una red neuronal, entrenada con datos de sensores láser de distancia 2D, para aprender – de forma supervisada – un conjunto de etiquetas topológicas definidas como significativas en un entorno doméstico que simula el espacio en el que se encuentra el robot. Cabe destacar que el método propuesto por Förster et al. (2007) no requiere la elaboración de un mapa explícito del entorno ni el cálculo de las coordenadas euclidianas.

En base a los resultados de los experimentos realizados como parte de dicho trabajo, sus autores afirman que el método utilizado es robusto al ruido en los datos de los sensores y al problema del “secuestro” (situación en la que el robot es retirado de su posición actual y trasladado a otra posición, sin aportarle ninguna información sobre este cambio al robot) (Förster et al., 2007). Los experimentos que ellos realizaron tuvieron lugar en un entorno simulado y en un número limitado. No obstante, las sugerencias de trabajos futuros de Förster et al. (2007), en parte han servido de inspiración para el presente trabajo. En particular, las posibles ventajas que ofrecen los métodos basados en mapas-topológicos en cuanto a

eficiencia en el uso de memoria y de interpretación semántica de los resultados que resaltan Förster et al. (2007), se ven alineadas con las características y limitaciones del entorno de *Edge Computing* para robótica móvil.

Cabe destacar que el uso de Redes Neuronales Recurrentes (RNN) de tipo LSTM también ha encontrado gran acogida en el procesamiento de otros tipos de datos de sensores, no necesariamente acoplados a agentes robóticos, sino también para estudiar y reconocer la actividad humana (Ordóñez y Roggen, 2016) como sistemas de detección de caídas (*Fall Detection Systems*) (Bello, 2019). Esto se debe a la buena capacidad de las redes RNN/LSTM para aprender y modelar el comportamiento de fenómenos descritos como series temporales (Malhotra, Vig, Shroff y Agarwal, 2015; O'Shea, Clancy y McGwier, 2016).

Las redes RNN también han sido utilizadas para reducir el error sistemático en modelos basados en el EKF (*Extended Kalman Filter*) (Kotov, Maltsev y Sobolev, 2013), que los autores han aplicado a tareas de SLAM en 2D en las que un robot hace el seguimiento de un punto objetivo (target) y reconstruye la trayectoria real usando datos de sensores de distancia. Es interesante notar que dichos autores usaron datos de simulación obtenidos con un robot móvil e-puck, concebido originalmente para uso educacional.

2.4. Localización semántica avanzada: sistemas visuales con *Deep Learning*

También existe un amplio número de trabajos en los que la localización incorpora imágenes, videos y/o información de sensores adicionales a los *Laser Range Finders* de 2D, como por ejemplo cámaras (Wang et al., 2018), sensores 3D, entre otros.

Algunos trabajos han extendido e integrado enfoques y métodos ya existentes para abordar problemas que enriquecen las capacidades de navegación y movilidad de agentes robóticos autónomos en un entorno modelado tridimensionalmente. Por ejemplo, Kumar, Bhandarkar y Mukta (2018) proponen el sistema DepthNet con una arquitectura de RNN (*Recurrent Neural Networks*) para plantear un modelo de Deep Learning que combina redes convolucionales y LSTM (ConvLSTM) con técnicas de SFM (*Structure From Motion*) para realizar predicciones sobre un mapa de profundidad de una escena en base a secuencias de video monocular. Como indican los autores en dicho trabajo, "*aprender a predecir la profundidad, incluso aunque sólo sea de manera aproximada, proporciona una oportunidad para inyectar información valiosa en los procedimientos de reconstrucción 3D, estimación e inferencia de la posición tridimensional en SLAM*" (Kumar et al., 2018). Estos resultados son especialmente interesantes ya que como hemos relatado en la introducción cada vez existen más soluciones

basadas en robots que navegan en un entorno real (no simulado, ni de laboratorio), que evidentemente es tridimensional.

Por otra parte, Sünderhauf y Protzel (2011) proponen un sistema de reconocimiento de lugares – "basado en apariencias" que incorpora BRIEF-Gist, una mejora a los descriptores de tipo BRIEF (*Binary Robust Independent Elementary Features*) para poder aplicar mapas semánticos con imágenes a tareas de reconocimiento y localización. Como indican Sünderhauf y Protzel (2011), los sistemas modernos de SLAM de estas características suelen estar divididos en dos partes: *front-end* y *back-end*. Donde el front-end construye un conjunto de restricciones del entorno y puntos de interés (*landmarks*) y el *back-end* contiene un optimizador que construye y mantiene un mapa del entorno. Los autores de dicho trabajo utilizan un sistema con la arquitectura anterior y aplican descriptores BRIEF-Gist a muestras de imágenes completas, extraídas de videos para realizar la identificación de lugares conocidos y resolver problemas de falsos "*closed loops*" (recorridos en bucles) en la trayectoria de un vehículo autónomo. El trabajo también destaca por el uso de datos de lugares reales para realizar tareas SLAM sobre 66km de espacios urbanos del St. Lucía *dataset* (Sünderhauf y Protzel, 2011).

Otros trabajos han impulsado el avance en la investigación y aplicación práctica de la localización y la realización de tareas SLAM a partir de observaciones semánticas (Atanasov et al., 2016; Fu et al., 2016). Atanasov et al. (2016) aportan una formulación formal del problema de localización semántica para un robot móvil cuya dinámica de movimiento está gobernada por un modelo que incorpora información sobre la ubicación actual del robot, su orientación, su velocidad, la aceleración, así como el posible ruido en los datos causado por el movimiento. En dicho planteamiento del problema, la localización semántica se describe como el cálculo de la probabilidad a posteriori de obtener un conjunto finito de observaciones semánticas después de aplicar una función de control al robot, dado un mapa semántico y unas probabilidades a priori de observación de puntos de interés ("*landmarks*").

Para dar solución a dicho problema Atanasov et al. (2016) plantean varias proposiciones que describen aspectos computacionales sobre cómo implementar filtros de partículas para la localización semántica. Otro enunciado formal de Atanasov et al. (2016) plantea el problema de la localización semántica *activa* como la posibilidad de que un observador pueda escoger (proactivamente) una trayectoria para la cual se optimiza el desempeño de la localización semántica. Además de la contribución conceptual, también hay que destacar el trabajo experimental realizado por los autores, incluyendo simulaciones y experimentos reales con una plataforma robótica móvil, equipada con ruedas "*encoder*", una unidad de medición inercial (IMU), una cámara Kinect RGB-D Nyko Zoom con lentes gran angular y un sensor

láser Hokuyo UTM-30LX, aunque ellos afirman que para las observaciones semánticas sólo utilizaron imágenes RGB.

Como se ha visto en los trabajos comentados en este capítulo y ya se mencionaba en la introducción, actualmente existe un gran número de propuestas de soluciones basadas en técnicas de Inteligencia Artificial que integran varios modelos funcionando de manera conjunta como partes de un mismo sistema. El modelo de navegación semántica visual basada en *Deep Learning* para robots móviles en espacios interiores propuesto por Wang et al. (2018) es un claro ejemplo de esta tendencia. En dicho trabajo, los autores proponen un marco ("*framework*") de percepción formado por tres capas: PRM (Place Recognition Model), 3RM (Rotation Region Recognition Model) y SRM ("Side" Recognition Model) (Wang et al., 2018).

Dicho sistema utiliza técnicas de *transfer learning* para mejorar las capacidades de percepción y reconocimiento semántico del entorno de los robots. Con este sistema se realizaron diversos experimentos, usando una plataforma robótica con un Kinect Sensor para capturar imágenes RGB y una tarjeta gráfica Leadtek Quadro K4200 para entrenar los modelos. Wang et al. (2018) reportan en sus resultados el desempeño del sistema explorando varias trayectorias para ir de habitaciones a habitaciones o del pasillo a las habitaciones, haciendo un uso intensivo de imágenes con los tres modelos para el reconocimiento de los lugares, el reconocimiento de regiones de rotación y el reconocimiento lateral, respectivamente.

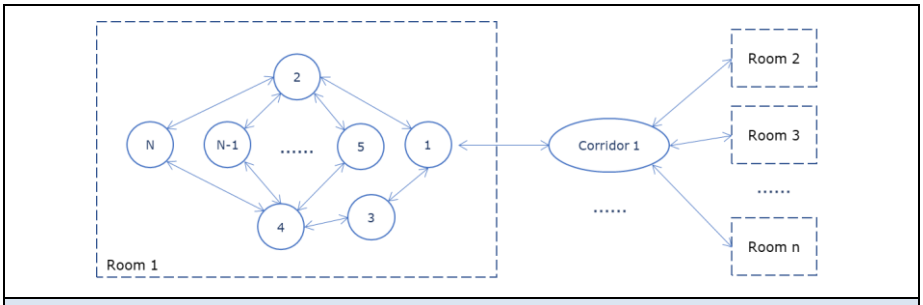


Figura 2. Diagrama de relaciones topológicas semánticas.
Fuente: Wang et al. (2018)

Otro aspecto destacable del trabajo anterior es el uso de un grafo dirigido de relaciones topológicas semánticas (Figura 2), donde las regiones semánticas se representan en nodos y los lados representan regiones reconocidas para la rotación (Wang et al., 2018).

El estudio de los trabajos previos, considerando la importancia de abordar estos proyectos con un enfoque sistémico, ha ayudado a plantear nuestra investigación y experimentos para explorar la localización semántica como una vía "económica" y efectiva que puede aportar conocimiento del entorno y dotar de mayor capacidad de decisión a un agente robótico móvil.

Capítulo 3. Objetivos y metodología

Considerando el estado del arte y los trabajos anteriores, el presente trabajo se ha planteado alcanzar los objetivos y utilizar la metodología que se describen a continuación.

3.1. Objetivos generales

Con este trabajo se pretende aportar un sistema que sirva de modelo para el desarrollo ágil de proyectos que incorporan técnicas de Inteligencia Artificial en un entorno de *Edge Computing* con agentes de robótica móvil. Por tanto, se desea alcanzar estos objetivos:

1. Desarrollar un prototipo como prueba de concepto de un sistema de recogida de datos y localización semántica de un agente robótico móvil usando técnicas de Inteligencia Artificial, basado en una *Single Board Computer* (SBC) Raspberry Pi y un sensor Laser Range-Finder (LRF) Hokuyo URG-04LX.
2. Realizar experimentos para estudiar la aplicación de varias técnicas de Inteligencia Artificial, usando el prototipo-prueba de concepto y el sistema propuesto.
3. Analizar y comparar el desempeño de las técnicas de Inteligencia Artificial usadas, explorando posibles mejoras para el sistema de localización semántica.
4. Desarrollar una guía (Checklist) en base a la propuesta de sistema-integrado para la implementación ágil de técnicas de Inteligencia Artificial en un entorno de computación en el borde (*Edge computing*) con un agente de robótica móvil.

3.2. Objetivos específicos y detallados

Para alcanzar los objetivos anteriores se han establecido las siguientes metas específicas:

- Guía (“Checklist”) del método de implementación ágil de técnicas de IA:
  - Realizar una reflexión sobre cuestiones clave de las fases de la metodología, fundamentada en la experiencia profesional del autor, así como el trabajo y aprendizajes conseguidos en el Máster de Inteligencia Artificial.
  - Elaborar preguntas clave de reflexión a plantearse en cada fase del proyecto.
- Desarrollar prototipo de sistema de localización semántica para un agente robótico móvil:
  - Realizar una definición conceptual-esquemática del todo el sistema, incluyendo una descripción sus componentes, así como los flujos de datos e información.
  - Señalar las partes del sistema con implementación “local” (en el borde/Edge) y las que tienen lugar o implementación en la nube.

  - Considerar como realizar el despliegue y explotación de los algoritmos de Inteligencia Artificial en el sistema propuesto.

- Realizar experimentos:
  - Configurar y programar los componentes del sistema para la recogida de datos.
  - Realizar la recogida de datos (Edge).
  - Subir los datos a la nube (Cloud)
  - Crear modelos usando algoritmos de Inteligencia Artificial y entrenarlos usando los datos recogidos: SVM, KNN, ANN, etc.
  - Explorar alternativas de uso de los datos recogidos y de algoritmos como apoyo a la toma de decisiones del agente robótico móvil.
  - Desplegar los modelos entrenados en el agente robótico móvil y comprobar su funcionamiento en “tiempo real”.
  - Validar el funcionamiento del prototipo de sistema de localización.

- Analizar y comparar el desempeño de los modelos de Inteligencia Artificial:
  - Obtener y comparar métricas de desempeño de los modelos desarrollados.
  - Identificar ventajas/desventajas y consideraciones relevantes en la implementación de tareas clave de aprendizaje automático.
  - Explorar formas de mejorar el desempeño de los modelos propuestos.
  - Elaborar estimaciones orientativas para comparar el almacenaje en la nube de datos primarios (ej. distancias) VS. datos calculados (ej. estadísticas, modelos).
  - Obtener conclusiones específicas en base a los resultados del trabajo.

3.3. Metodología

La metodología utilizada se ha basado en un enfoque de trabajo ágil e iterativo. De esta forma se ha buscado poder desarrollar el proyecto de forma incremental y completarlo de manera satisfactoria, considerando las características singulares de un alcance híbrido (incluyendo métodos, hardware, software, experimentos, gestión de datos, etc.) y los plazos de entrega del trabajo. La Figura 3 muestra un esquema de la metodología, donde se puede apreciar que, si bien varias de las fases y actividades principales pudieran considerarse de forma lineal y secuencial, en la práctica se han desarrollado de forma solapada. Estos solapes han permitido la retroalimentación entre varias de dichas actividades que a su vez ha facilitado la adaptación continua del trabajo a los retos encontrados y a las soluciones, observaciones y conclusiones preliminares que se iban obteniendo. A continuación, se describen las fases de la metodología utilizada en el trabajo.

3.3.1. Fase 1. Desarrollo metodológico

En esta fase se han abordado las consideraciones metodológicas para poder realizar un proyecto de estas características. En primer lugar, se realizó un planteamiento inicial consistente en grandes líneas en las mismas fases aquí descritas. Si bien en una primera instancia no estaba claro o decidido cuántos experimentos se iban a realizar, durante el desarrollo del proyecto se consideró necesario extender-ampliar los experimentos y mediciones realizadas para incluir otras viviendas adicionales a las inicialmente previstas y poder obtener más datos con los que trabajar.

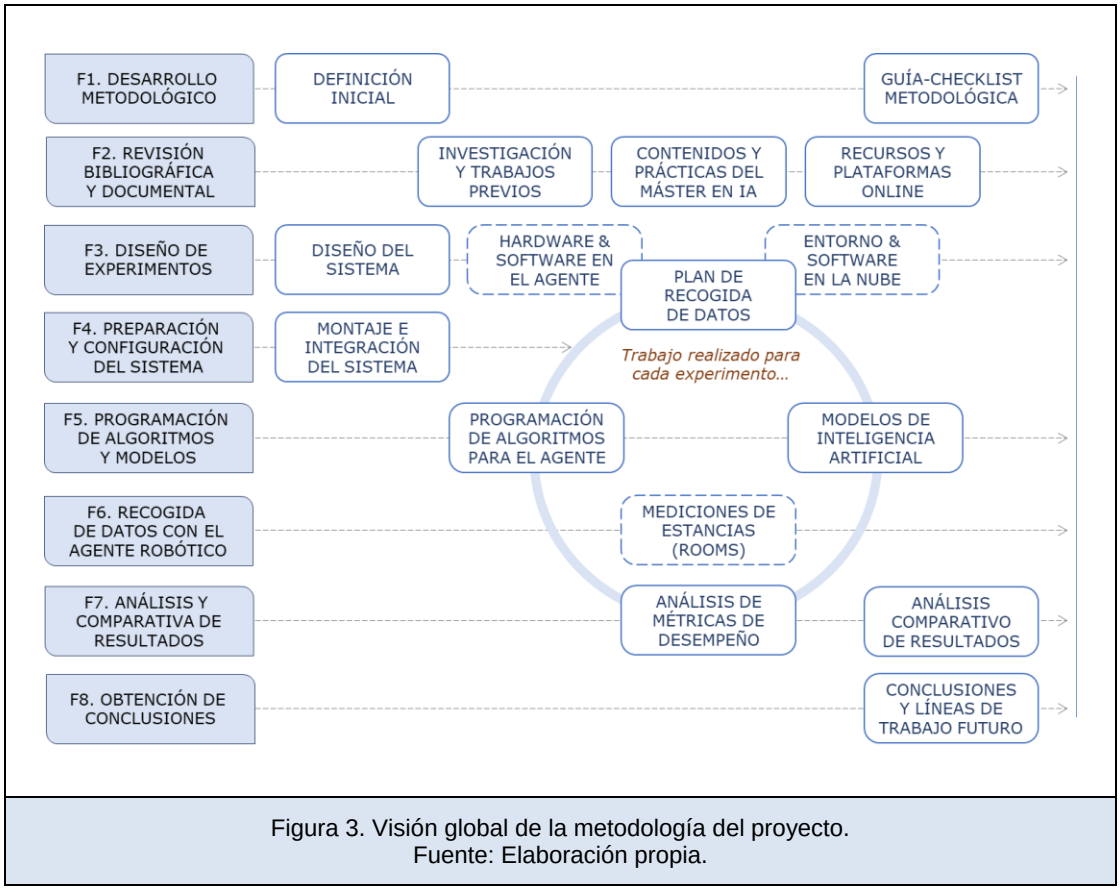


Figura 3. Visión global de la metodología del proyecto.
Fuente: Elaboración propia.

Esta decisión fue soportada con facilidad gracias al carácter iterativo del método usado en el proyecto. En la etapa final del proyecto, se realizó una reflexión global sobre los aspectos más destacados de cada fase y las decisiones que se tuvieron que tomar, así como consideraciones críticas. Estas reflexiones se han recogido como posibles lecciones aprendidas y se han documentado en el Anexo 4 “Guía-Checklist Metodológica” como un resultado adicional del trabajo realizado.

3.3.2. Fase 2. Revisión bibliográfica y documental

La fase de investigación bibliográfica y revisión documental ha tenido lugar durante prácticamente toda la duración del trabajo. Dada la relevancia y multiplicidad de perspectivas

del problema tratado (robótica móvil, *Edge Computing*, localización semántica, aprendizaje automático supervisado, etc.), se ha estudiado una amplia diversidad de trabajos, incluyendo publicaciones científicas, libros, información divulgativa sobre tendencias, blogs y foros afines, así como webs y material publicitario de algunos productos innovadores relacionados con los objetivos de la investigación del proyecto.

Además, el uso de plataformas online como Google Colab y Google Drive, lenguajes de programación como Python, el ROS (Robot Operating System) y varias librerías software (scikit-learn, TensorFlow/Keras, etc.) para desarrollar la implementación de los algoritmos seleccionados ha requerido un estudio detallado de la documentación de referencia de estas fuentes, la consulta de blogs especializados, así como visitar oportunamente algunas comunidades online de desarrolladores (ej. ROS *Discussion Forums*, stackoverflow.com, etc.). También se han consultado los manuales técnicos y de referencia del sensor láser utilizado (Hokuyo URG-04LX), de la Raspberry Pi 3B+ y de otros periféricos como la guía de montaje de la pantalla táctil SmartiPi Touch usada en el prototipo desarrollado.

3.3.3. Fase 3. Diseño de experimentos

En la fase del diseño experimental, se han tenido en cuenta las características propias y las restricciones más relevantes del dominio del problema para plantear las preguntas, hipótesis y asunciones relevantes que se deben explorar, validar o rechazar. Tratándose de la localización semántica de un agente robótico móvil mediante el uso exclusivo de datos de sensores (entorno *Edge Computing*), se han planteado cuestiones e hipótesis de trabajo relativas a la selección, recogida y etiquetado de los datos, se han considerado y gestionado situaciones que podían haber generado sesgos, incorporando en el diseño y realización de los experimentos variables y factores para garantizar una diversidad adecuada de los datos.

Las preguntas de exploración que han dirigido la indagación y los trabajos realizados con los experimentos incluyen:

- ¿Cómo realizar un diseño de prototipo como prueba de concepto para un proyecto de recogida ágil de datos, que sirva de entrada a sistemas de localización usando técnicas de Inteligencia Artificial con un ordenador de placa única (Raspberry Pi) y un sensor láser como una solución de bajo coste de *Edge Computing*?
- ¿Qué modelos y algoritmos de Inteligencia Artificial se pueden entrenar y/o desplegar en el sistema de *Edge Computing* anterior (basado en una Raspberry Pi y un sensor LRF Hokuyo) para apoyar la toma de decisiones en una aplicación de robótica móvil?
- ¿Qué implicaciones tiene la elección y uso de uno u otro modelo de Inteligencia Artificial en términos de cantidad de datos, precisión de los resultados, escalabilidad, etc.?

- ¿Es posible realizar una localización “semántica” de un agente robótico móvil usando exclusivamente datos de un sensor láser LRF? Es decir, sin necesidad de usar o recurrir a un mapa o representación SLAM (*Simultaneous Localization And Mapping*). En caso afirmativo, ¿con qué niveles de precisión puede llevarse a cabo dicha localización?
- ¿Qué otros escenarios de uso y aplicaciones de los datos del sensor LRF se pueden abordar en el marco del prototipo y experimentos planteados?
- ¿Cómo pueden ayudar las técnicas de *“ensemble learning”*? ¿Se puede “enriquecer” el clasificador para mejorar las predicciones de localización del agente robótico móvil usando información o conocimiento del entorno? ¿Puede un mapa o grafo topológico ayudar a “ajustar” y mejorar las clasificaciones y la localización del agente?
- Desde un punto de vista metodológico, ¿Qué cuestiones o aspectos han de considerarse de forma oportuna en el diseño, ejecución e implementación de un proyecto de esta naturaleza?

Completando el enfoque práctico del diseño de los experimentos, se ha preparado un plan de recogida de datos. Este plan se ha planteado de forma incremental para ir respondiendo a las cuestiones anteriores aplicando las técnicas de Inteligencia Artificial al problema de la localización semántica del agente robótico móvil en varias estancias o habitaciones de viviendas diferentes. El diseño de los experimentos también tuvo en cuenta las limitaciones de procesamiento de la arquitectura seleccionada, por lo que parte del procesamiento de los datos se planteó como un trabajo a realizar en la nube, usando los servicios de la plataforma de Google Colab.

3.3.4. Fase 4. Preparación y configuración del sistema

Para realizar el trabajo se ha preparado y configurado una prueba de concepto (“*proof-of-concept*”) de un sistema hardware que ha sido utilizado como prototipo funcional del subsistema de recogida de las mediciones de distancia, emulando lo que pudiera ser un dispositivo para tal fin, usando una Raspberry Pi 3B+ y un sensor láser HOKUYO como elementos centrales de este componente del agente robótico móvil.

Además, para facilitar y poder realizar una recogida de datos repetible y consistente, se preparó como parte de dicha prueba de concepto, una plataforma móvil para ser usada como base de trabajo en los experimentos, instalándola en una maleta con fijaciones de cámaras profesionales GoPro. De esta forma, se ha podido instalar y operar el sensor láser HOKUYO, conectado directamente vía USB a la Raspberry Pi, y conseguir mediante una fuente de

alimentación portátil una movilidad y altura similar a la que podría tener un sensor de este tipo en un robot de servicio o doméstico. Esta solución ha permitido asegurar que los datos del sensor utilizado se han capturado desde aproximadamente una misma altura relativa al suelo por donde se ha realizado el trayecto de recogida de datos con el concepto de prueba del agente robótico. Teniendo en cuenta que los espacios incluidos en el estudio han sido distintos tipos de habitaciones de viviendas ubicadas en edificios diferentes, garantizar esta consistencia en el proceso de recogida de las mediciones ha permitido hacer comparaciones entre los algoritmos implementados con los datos de dichas viviendas/estancias sin riesgo de introducir variables no-deseadas en el estudio.

Así mismo, se ha tenido en cuenta que para la tarea de reconocimiento y clasificación del "patrón" de los datos del sensor dada una determinada posición, la altura desde la cual se capturan dichos datos puede tener consecuencias e impacto en la calidad o riqueza de dichos datos. Por ejemplo, es muy probable que en un espacio doméstico (por ejemplo, en un salón o dormitorio) existan obstáculos semipermanentes como muebles o equipos electrodomésticos cuya altura no sea superior a un metro. Si consideramos que los datos se toman en un plano horizontal (paralelo al suelo), a la altura del sensor, ciertamente el perfil de la "huella" de dichos datos en una posición puede ser bien diferente para distintas alturas.

Siendo conscientes de esta situación y considerando los objetivos planteados para este trabajo se ha aceptado la asunción de que esta decisión de diseño del prototipo no influye de manera significativa en la calidad de los datos para explorar las preguntas-hipótesis y objetivos de los experimentos y del propio trabajo. Aun así, también se consideraron como parte del diseño, otros escenarios para la recogida de datos: desde distintas alturas (por ejemplo, con un robot móvil de pequeñas dimensiones) o incorporando un sensor láser de 360º (en lugar al utilizado de rango 240º). Finalmente, para facilitar la implementación en el tiempo disponible se optó por el prototipo que se describe en el Capítulo 4.

Para poder acceder a los datos capturados por el sensor láser desde la Raspberry Pi, ha sido necesario instalar y configurar los paquetes ROS que permiten la comunicación con el sensor láser LRF y varias librerías software requeridas como dependencias críticas de dichos paquetes ROS. Al tratarse de un entorno Open Source, existen abundantes fuentes de información sobre estos procesos de configuración e instalación. No obstante, con cierta frecuencia es posible que no sea fácil encontrar una explicación concisa, detallada o bien orientada que aplique a la configuración de hardware-software que se desea utilizar. Por tanto, esta parte del trabajo de diseño y preparación del sistema ha tenido que ser abordada de manera iterativa hasta conseguir una configuración funcional adecuada.

Complementariamente al mencionado “subsistema” SBC (Raspberry Pi) + sensor láser LRF, se han usado ordenadores personales-portátiles y los servicios de almacenamiento en la nube como Google Drive, Microsoft OneDrive, así como la plataforma Google Colab para realizar desarrollos y programación en Python. El ecosistema de recursos técnicos para desarrollar el sistema usado en los experimentos ha sido completado con varios periféricos como teclados inalámbricos, pantalla LCD/táctil, monitor/televisor, así como los imprescindibles servicios de conexión a Internet a través de WIFI o desde el móvil 4G.

3.3.5. Fase 5. Programación de algoritmos y modelos

En esta fase se han desarrollado varios programas, funciones, scripts y algoritmos usando técnicas de Inteligencia Artificial y de la ciencia del dato (“*Data Science*”) para:

- Capturar datos del sensor láser, proceder a su etiquetado y registrarlos de manera utilizable para los modelos que se desean desarrollar. Estos desarrollos han sido realizados en Python, desplegando una interfaz efectiva a través de componentes ROS *Packages* para el sensor láser. Estos algoritmos y funciones se programaron directamente en la Raspberry Pi 3B+ y fueron ejecutados sobre el sistema operativo Linux Ubuntu (16.04) instalado en dicha Raspberry Pi.

- Crear, entrenar y validar varios modelos basados en técnicas de Inteligencia Artificial, incluyendo Máquinas de Vector de Soporte (Support Vector Machines, SVM), *K-Nearest Neighbors* y Redes Neuronales Artificiales (Perceptrón Multicapas, MLP). Estos modelos han sido programados y entrenados en Python, usando la plataforma de Google Colab. Su ejecución y validación ha tenido lugar en la nube (Google Colab) y también en el borde (Edge Computing). Ver punto siguiente.

- Desplegar los modelos de Inteligencia Artificial en el borde (Edge Computing). Una vez entrenados en Google Colab, se han desplegado y ejecutado en tiempo real en la Raspberry Pi, realizando la captura de datos de los sensores “en vivo” y clasificando, “prediciendo” en qué estancia, espacio o ubicación se encontraba el agente robótico.

3.3.6. Fase 6. Recogida de datos con el agente robótico móvil

En esta fase se han usado las funciones y los algoritmos antes mencionados, y se ha seguido el Plan de recogida de datos del diseño experimental para realizar varias sesiones de recogida de datos. Estas sesiones de recogida de datos con el agente robótico móvil han tenido lugar en varias viviendas ubicadas en la comunidad de Madrid, por lo que el autor ha desplazado hasta éstas el prototipo “prueba de concepto” y lo ha configurado para que la recogida de

datos ocurra con etiquetado simultáneo a la captura de las mediciones. Los datos recogidos de esta forma han sido almacenados localmente en la tarjeta de memoria de la Raspberry Pi. Posteriormente, los datos han sido “subidos” (enviados) a la nube, a un servidor de Google Drive para ser almacenados de forma segura y procesados para limpiarlos y prepararlos para el modelado como parte de la siguiente fase.

La recogida de datos se realizó en primer lugar en una vivienda (código: 1SY). Estos datos fueron usados para ejecutar y validar varios modelos que fueron posteriormente desplegados en la Raspberry Pi, validando así que todo el flujo de trabajo se podía completar correctamente. Posteriormente, se procedió a recoger datos en otra vivienda (2CA) repitiendo el proceso anterior, y finalmente se recogieron los datos de las otras dos viviendas (3ME y 4TR) en sesiones realizadas en dos días consecutivos.

3.3.7. Fase 7. Análisis y comparativa de resultados

La fase de análisis y comparativa de los modelos ha sido muy enriquecedora y productiva. En esta fase se han obtenido diversas métricas del desempeño de los modelos implementados, tanto a través de conjuntos de datos de validación y pruebas, como con datos capturados en directo, obtenidos tras el despliegue de los modelos en el prototipo.

Training & Cross-testing de los modelos		Probado con datos de las VIVIENDAS				
		1SY	2CA	3ME	4TR	TODAS
Entrenado con datos de las VIVIENDAS	1SY	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	
	2CA	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	
	3ME	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	
	4TR	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	
	TODAS					1, 2, 3, 4

MODELOS INCLUIDOS:
1) SVM: Support Vector Machines (LinearSVC)
2) SVM-RBF: Radial Basis Function
3) KNN: K-Nearest Neighbors
4) ANN-MLP: Artificial Neural Networks - Multilayer Perceptron
5) *Yielding Associations Network* (Propuesta de este TFM):
usando SVM (kernel=linear) con el "Filtro de Contexto"

Figura 4. Enfoque de entrenamiento y pruebas cruzadas de los modelos.
Fuente: Elaboración propia.

Usando los resultados de estas métricas de desempeño de los distintos modelos se han realizado comparaciones diversas. Para maximizar la riqueza de esta parte de los experimentos se ha definido y utilizado un enfoque de entrenamiento y *cross-testing*, según el cual varios de los modelos desarrollados se han entrenado con datos de una vivienda y

probado con datos de todas las viviendas. Así mismo, también se han desarrollado y entrenado varios modelos usando los datos de todas las viviendas. Estos modelos "integrados" también han sido probados con datos de todas las viviendas. La Figura 4 resume este enfoque de pruebas cruzadas que ha permitido realizar de manera sistemática más de 68 pruebas-experimentos con los modelos entrenados.

En esta fase también se realizó un análisis y búsqueda de oportunidades de mejora del desempeño de los modelos desarrollados. Realizando un estudio detallado de los resultados de varios de los clasificadores desarrollados, se descubrió que algunas de las clasificaciones incorrectas se pueden corregir usando información sobre el contexto en el que se encuentra el agente robótico móvil. Esto permitió definir un operador que se ha denominado Filtro de Contexto. Usando el Filtro de Contexto propuesto se realizaron varias pruebas y análisis de los resultados.

Este análisis permitió confirmar las mejoras de desempeño que se esperaban y, además, observar que existe otra forma de aplicar el conocimiento del entorno. Esta segunda forma o variante del Filtro de Contexto se definió usando las probabilidades de pertenencia a cada clase que devuelve el clasificador. Partiendo de la definición de estas dos variantes de Filtro de Contexto se entrenaron cinco modelos diferentes para comparar el desempeño de los filtros. Los resultados se analizaron como parte de esta fase, permitiendo considerar y proponer varias opciones para usar estos filtros en situaciones reales.

Por último, se han usado los datos del tamaño real en disco de los archivos de *datasets* y modelos creados para realizar un análisis y una estimación comparativa del espacio requerido si se desea guardar en la nube un histórico de las ubicaciones en las que ha estado el agente robótico móvil. Así, a través de un ejercicio hipotético-ilustrativo se ha podido apreciar el valor de las técnicas de Inteligencia Artificial en un contexto de Edge Computing.

3.3.8. Fase 8. Obtención de conclusiones

En base a los resultados de los experimentos, las mediciones y los análisis realizados, se han extraído varias conclusiones destacadas del trabajo, resaltando también varios aprendizajes obtenidos con el proyecto. La obtención de conclusiones también se ha enriquecido a través de la elaboración de la guía metodológica que se ha creado como parte de una reflexión global sobre el desarrollo de este proyecto de investigación. Además, se han explorado posibles líneas futuras de investigación y/o de aplicación a oportunidades de negocio que pueden dar continuidad al trabajo de investigación realizado.

Capítulo 4. Desarrollo y contribuciones del trabajo

4.1. Descripción general de las contribuciones

En este capítulo se presentan los resultados del proyecto. Estas contribuciones incluyen:

- Consideraciones para el diseño de una solución sistémica y de los experimentos
- Retos y decisiones sobre la implementación de algoritmos de Inteligencia Artificial
- Demostración del despliegue y uso en el entorno Edge Computing
- Análisis comparativo de métricas de desempeño y posibles mejoras
- Propuestas novedosas de modelos y algoritmos para la localización semántica

4.2. Consideraciones para el diseño de la solución y de los experimentos

Aun tratándose de una prueba de concepto, se ha utilizado una situación de aplicación real, consistente en realizar la localización semántica de un robot móvil usando sólo datos de distancia obtenidos con un sensor láser. Las características de este problema han ayudado a definir los casos de uso y diseñar la solución considerando los siguientes elementos:

- El movimiento del agente robótico móvil ha sido simulado con una plataforma instalada en una maleta con ruedas (Figura 5), desde donde se han recogido las mediciones del sensor láser, mientras la maleta era desplazada por el espacio que se quería medir.

Sensor láser LRF: Hokuyo URG-04LX
Raspberry Pi 3B+
SmartiPi Touch
Teclado Bluetooth
Batería portátil

Figura 5. Prototipo “prueba de concepto” para los experimentos y la recogida de datos. Fotos: Elaboración propia.

- Las mediciones obtenidas corresponden a seis tipos de habitaciones diferentes. Los códigos utilizados para las habitaciones han sido los siguientes: R01S-Salón, R02C-Cocina, R03H-habitación/dormitorio, R04P-pasillo, RP5D-despacho y R06B-baño.

- Se han recogido y guardado datos de mediciones de los seis tipos de habitaciones anteriores en cuatro viviendas diferentes de Madrid capital, todas de uso familiar. Las viviendas han sido codificadas como “1SY”, “2RJ”, “3ME” y “4TR”. Como ilustra la Figura 6, se han obtenido, por tanto, datos de veinticuatro espacios distintos.

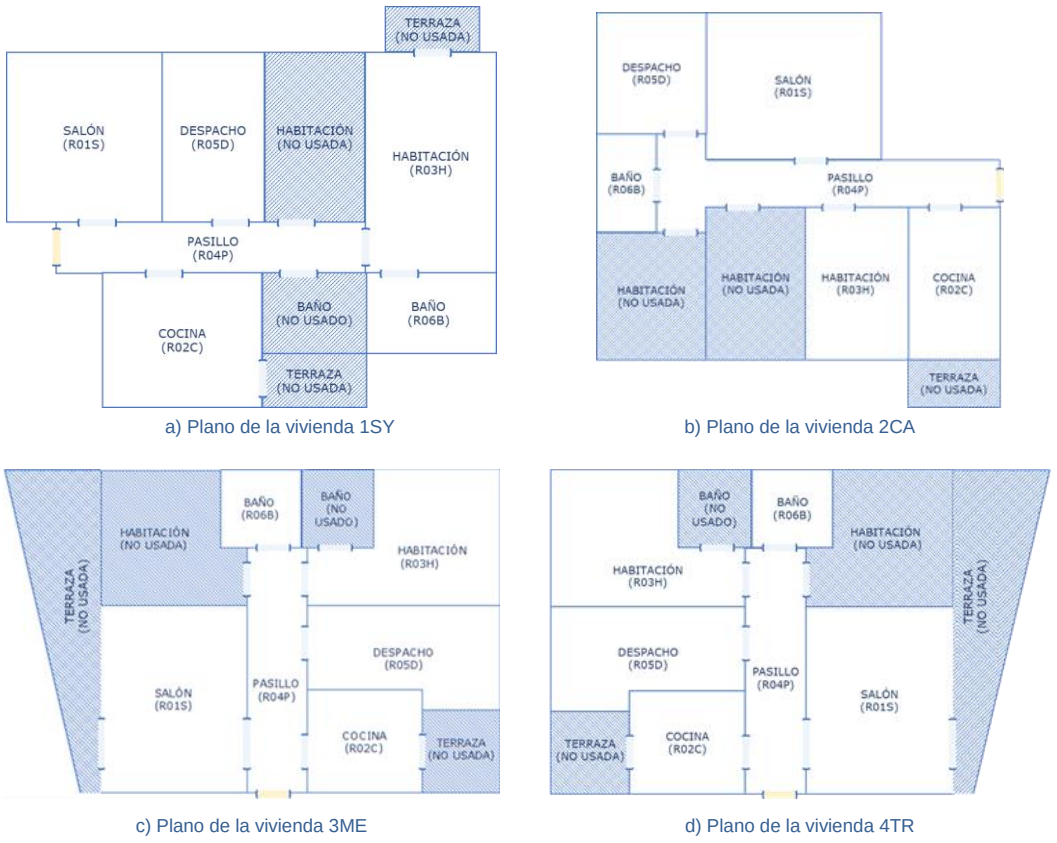


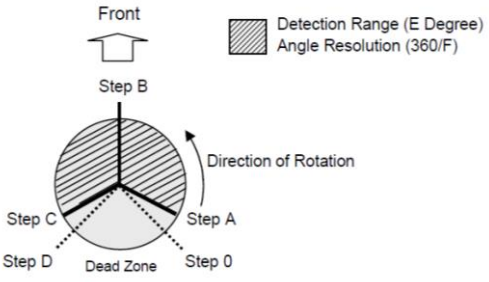
Figura 6. Planos de las cuatro viviendas y los espacios donde se recogieron los datos.
Fuente: Elaboración propia.

En un proyecto de estas características existen múltiples consideraciones que se deben tener en cuenta en la toma de decisiones sobre el diseño de una solución. En primer lugar, es necesario basar el trabajo de definición y diseño de la solución en un análisis que permita alinearse y satisfacer los objetivos que se quieren conseguir. En este caso, para poder dar respuesta al problema/reto de localización de un robot móvil usando los datos de un sensor láser, hay que destacar el importante papel que juegan los datos para la implementación de algoritmos y técnicas de Inteligencia Artificial. Por tanto, desde el inicio del proyecto se ha trabajado en entender, conocer y decidir sobre las siguientes cuestiones:

- **¿Cuáles, cómo y cuántos datos se pueden capturar con el sensor láser que vamos a utilizar (HOKUYO LRF)? ¿En qué formato se obtienen estos datos? ¿Con qué precisión y frecuencia de muestreo se deben recoger, guardar, etc.?**

  Como resultado de esta indagación inicial sobre los parámetros de medición y características técnicas del sensor, se pudo tener en cuenta en el diseño de los experimentos y en la programación de los algoritmos que aunque el sensor utilizado permite recoger datos “oficialmente” en un rango angular de detección de 240º, realmente el vector de mediciones que se obtiene mediante la interfaz de ROS y que el autor ha procesado en Python, incluye también datos de la zona “muerta” o “fronteriza” que está entre los ángulos de los pasos Step 0 al Step A y Step C al Step D (Figura 7).

  Estos datos corresponden a las mediciones obtenidas en las direcciones de los ángulos menores a -20º y mayores a 210º. Pero estos valores no son precisos o fiables para la detección de objetos u obstáculos, y en consecuencia no sirven para medir las distancias deseadas. Por tanto, fue necesario descartar dichos datos de la parte inicial y final de la ventana de medición para quedarnos solamente con la ventana de detección de 240º. Con los datos de la ventana real de detección se generaron los vectores compuestos por 681 mediciones de distancia.



		URG-04LX
Step 0	First Measurement Point	0
Step A	Initial measurement Step of Detection Range	44
Step B	Sensor Front Step	384
Step C	End point of Detection Range	725
Step D	Last Measurement Point	768
E	Detection Range	239.77
F	Slit Division	1024

Figura 7. Especificaciones y parámetros del sensor LRF Hokuyo URG-04LX.
Fuente: Kawata (2006).

- **¿Cómo realizar el etiquetado de los datos de manera automática**?

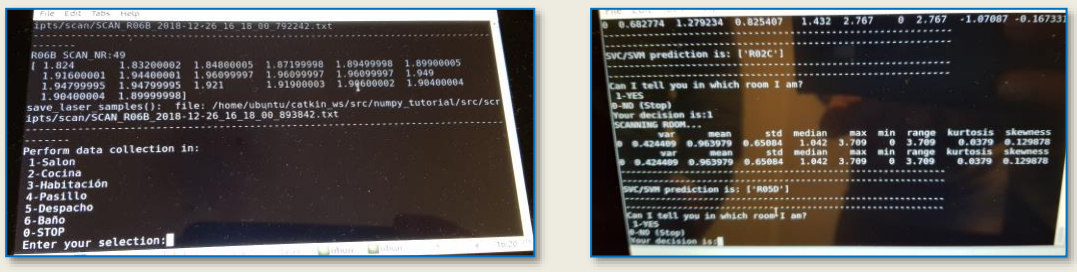
  Otro aspecto que afecta el planteamiento, desarrollo e implementación de un sistema basado en técnicas de Inteligencia Artificial de aprendizaje automático supervisado es la disponibilidad y calidad del etiquetado de los datos de entrenamiento. Se trata de un factor que atendido adecuadamente puede facilitar enormemente el trabajo de preparación de las fuentes de datos (*datasets*) que se van a utilizar y agilizar, por tanto, el desarrollo del proyecto en las etapas de definición y entrenamiento de los modelos de

Inteligencia Artificial. Por esta razón, se ha creado un código-identificador para cada habitación o estancia donde se recogen datos y así, se puede usar en el desarrollo de los algoritmos y en la ejecución de estas funciones para recoger y guardar los datos, realizando el etiquetado simultáneamente a su captura. Ver Figura 8.

Figura 8. Visión global de la creación de los *datasets* para los experimentos.
Fuente: Elaboración propia.

4.2.1. Creación de una interfaz básica con un "menú" de opciones

Como consecuencia de la necesidad de etiquetar los datos de forma simultánea a su recogida, se ha programado un diálogo-menú en Python para permitir la elección de la habitación en la que se va a realizar la recogida de datos (Figura 9a). Otra interfaz similar, sencilla pero efectiva, ha sido implementada para usar los modelos entrenados y desplegados en el agente robótico para obtener la clasificación (ubicación) más probable en base a los datos que el propio agente percibe de su entorno a través del sensor láser LRF.



a) Etiquetando los datos recogidos en cada estancia b) Predicción/clasificación de la ubicación del robot

Figura 9. Interfaces para la captura de datos, el etiquetado y la clasificación.
Fuente: Elaboración propia.

El uso de *datasets* “propios” (recogidos con el prototipo) ha permitido un entendimiento profundo del valor de los datos en el desarrollo de los modelos de Inteligencia Artificial.

4.3. Implementación de modelos y técnicas de Inteligencia Artificial

En esta sección se describe el trabajo realizado para el diseño y entrenamiento de varios modelos de Inteligencia Artificial para abordar el problema de la localización semántica del agente robótico móvil. En primer lugar, hay que destacar que este problema se ha planteado como un problema de clasificación a resolver con técnicas de aprendizaje automático supervisado. Por tanto, para poder reconocer el espacio o habitación donde se encuentra el agente robótico móvil, se entrena un modelo usando datos etiquetados previamente, que indican las posiciones correctas que se espera que aprenda el clasificador. Por otra parte, es importante destacar el papel de las variables de entrada, características predictoras o *features*, tal y como se describe a continuación.

4.3.1. Recogida de los datos: limpieza y preparación de los *datasets*

En cada una de las habitaciones, se ha realizado el mismo procedimiento de recogida de datos de la siguiente manera. Se ha iniciado el programa de captura de datos y se ha indicado el código de la habitación. A partir de esta elección, se ha comenzado la recogida de datos, guardando en cada instante de tiempo (según las especificaciones del sensor LRF Hokuyo un vector de mediciones de 240º se recoge en 100ms) el resultado de los pasos de medición del sensor láser en un vector de 769 componentes. La plataforma móvil se ha desplazado por la habitación siguiendo un recorrido lo más amplio posible dirigido por el autor.

Este proceso se ha realizado hasta que el programa de captura de datos ha registrado 300 vectores en cada habitación. La captura de datos en una habitación ha durado entre dos y tres minutos. En total se han recogido 6.400 registros, que una vez “limpiados” son vectores de 681 elementos cada uno. Toda la recogida de datos se ha realizado de manera autónoma en la plataforma móvil (Raspberry Pi + sensor láser Hokuyo).

La Figura 10 muestra los resultados de uno de los algoritmos que se han desarrollado como parte de este trabajo para visualizar las mediciones de distancias recogidas en cada tipo de habitación. Esta visualización es importante porque permite apreciar las diferencias en el perfil de los datos que se obtienen con el sensor láser en distintas posiciones o direcciones, así como en ubicaciones singulares como las zonas próximas a puertas o a otras habitaciones.

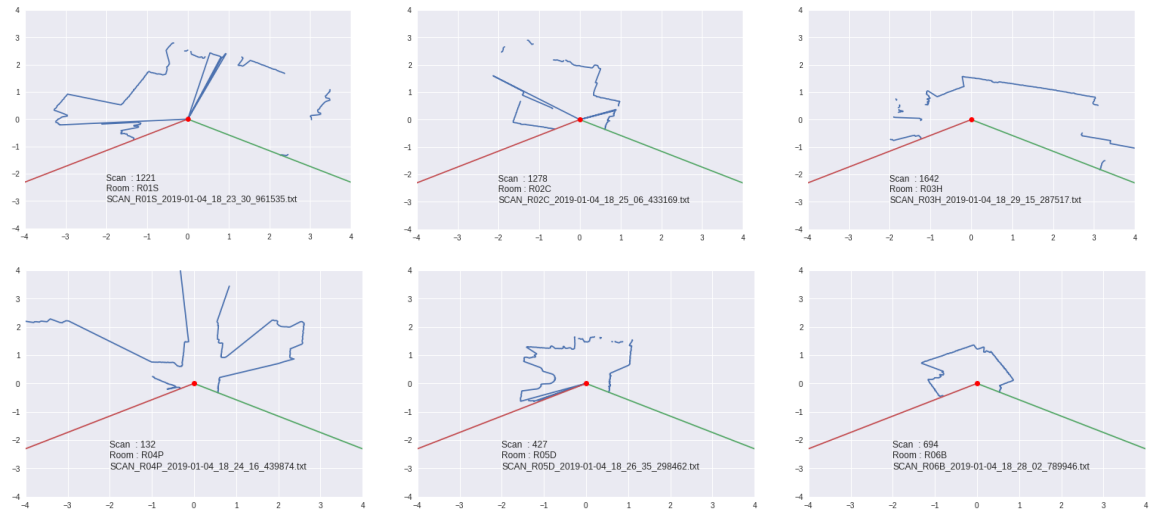


Figura 10. Visualizando el resultado de las mediciones con el sensor láser.
Fuente: Elaboración propia.

Posteriormente, el *dataset* completo de cada vivienda – compuesto por 1800 vectores – ha sido guardado/enviado a la nube (Google Drive) para realizar el preprocesamiento de los datos. Ya en la nube, se ha procedido entonces a la limpieza y preparación de los datos, descartando los 44 primeros y los 44 últimos componentes de los vectores anteriormente recogidos, para quedarnos sólo con las mediciones correspondientes a los 240º del rango real de detección del sensor láser. Además, se han agregado todos los vectores para crear un único archivo con el *dataset* resultante completo. Estos algoritmos y funciones de preprocesamiento se han programado en Python usando Google Colab.

Selección de "features" para el diseño de los modelos de Inteligencia Artificial

Un aspecto importante en la preparación de los datos para ser usados en un modelo de clasificación basado en técnicas de Inteligencia Artificial es la elección o definición de las variables de entrada, características o atributos (*"features"*).

Los experimentos desarrollados como parte de este trabajo se han basado en el uso de dos tipos de "*features"* para la localización del agente robótico móvil:

- Para los modelos de Máquinas de Vector de Soporte y K-Nearest Neighbors se ha realizado una ingeniería de factores ("*features engineering*") para obtener variables de tipo estadístico a partir de los datos del vector de distancias medidas por el sensor láser. Estas características o atributos estadísticos han sido: Media, Mediana, Mínimo, Máximo, Rango, Varianza, Desviación estándar, Curtosis y *Skewness* (como medida de asimetría).
- Para los modelos basados en redes neuronales se han usado: 1) los valores de distancia medidos por el sensor láser y 2) las "*features*" estadísticas anteriores (normalizadas).

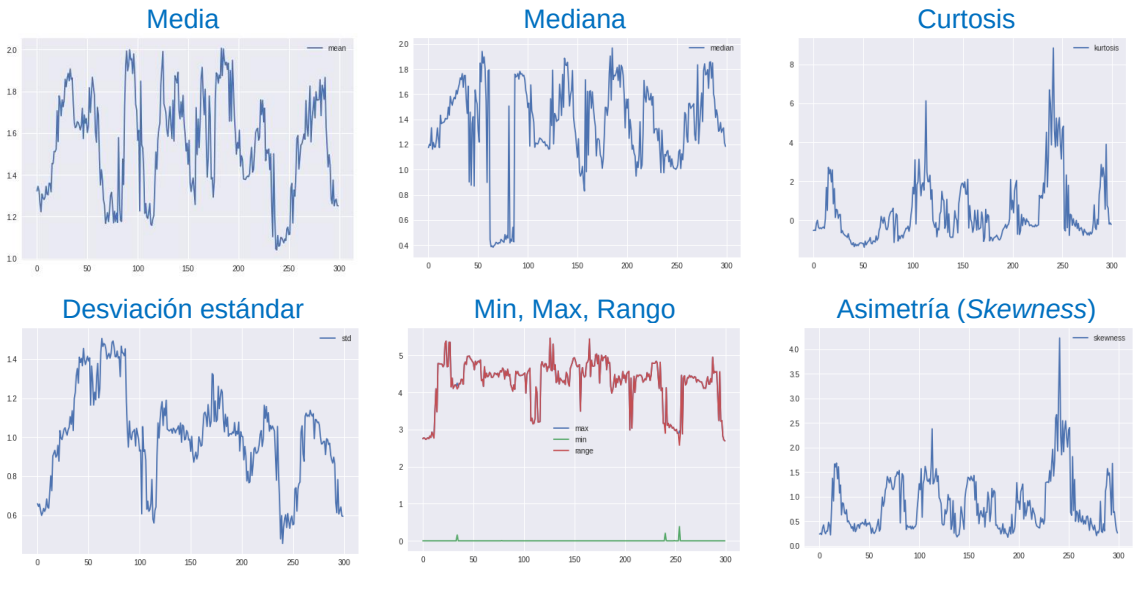


Figura 11. Evolución de las "*features*" para la habitación R03H (vivienda 2CA).
Fuente: Elaboración propia.

Las Figuras 11 y 12 muestran los valores de las "*features*" de dos estancias diferentes. La evolución temporal de los valores corresponde a una transformación de lo que "percibe" el agente robótico móvil mientras se desplaza por la estancia (habitación, pasillo, etc.). Visualizar el comportamiento de las variables de entrada que se desean usar en nuestros modelos, ayuda a entender y a poder escoger aquellas que mejor describan el fenómeno en estudio.

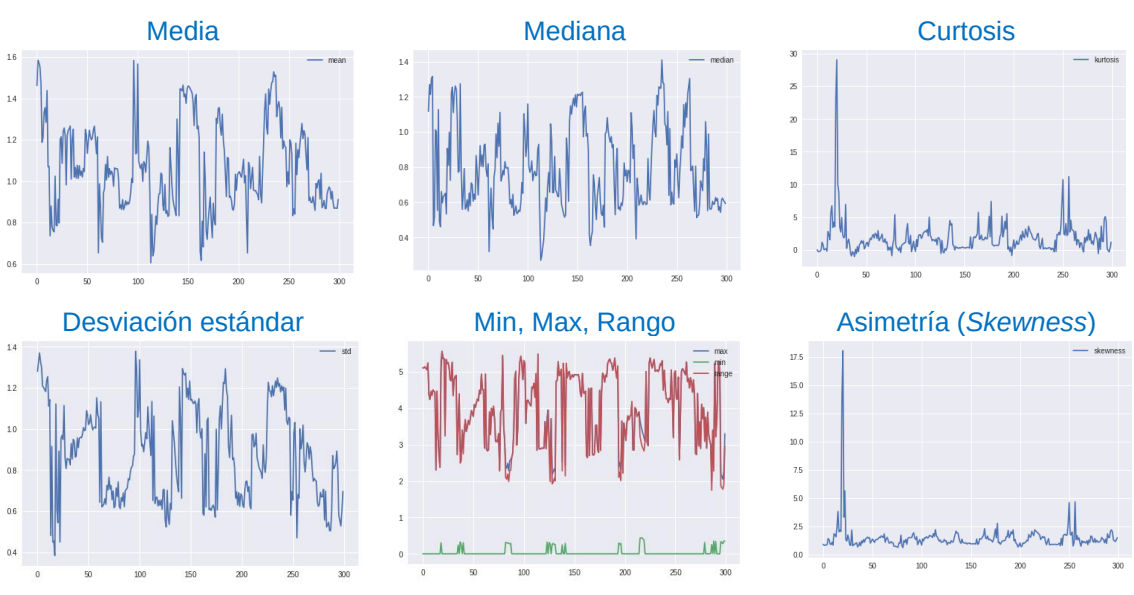


Figura 12. Evolución de las "*features*" para el pasillo R04P (vivienda 4TR).
Fuente: Elaboración propia.

En las Figuras 11 y 12, se pueden apreciar notables diferencias entre el "perfil" de los datos de Curtosis/*Skewness* (Asimetría) entre una habitación/dormitorio (R03H) y un pasillo (R04P). Aunque esta observación pueda parecer prematura y ciertamente aun no es posible determinar si estas variables van a ser suficientemente expresivas como para poder distinguir y reconocer o diferenciar las estancias de las viviendas, lo que sí hubiese sido un indicador poco prometedor es que las variables candidatas tuviesen un comportamiento casi exacto o muy similar en las habitaciones o espacios que se quieren "aprender" con el clasificador.

Sobre normalización y data-augmentation

Una técnica usada frecuentemente para mejorar el desempeño de los algoritmos de aprendizaje automático es la normalización de los valores de las variables de entrada. En nuestro caso, hemos realizado la normalización de las "*features"* estadísticas para el entrenamiento de las redes neuronales, pero no para los otros modelos. El resultado de la normalización de seis vectores se ilustra en la Figura 13.

a) Antes de la normalización b) Después de normalizar las variables

Figura 13. Perfil de las "*features*" antes y después de normalizar.
Para ilustrar las diferencias se han incluido datos de un registro de cada habitación de la vivienda 1SY. Fuente: Elaboración propia.

En este trabajo, también se contempló la posible aplicación de técnicas de *data-augmentation* para generar más ejemplos o registros partiendo de las mismas mediciones obtenidas inicialmente. Se trata de una técnica que ha sido aplicada en otros trabajos previos, por ejemplo, rotando los datos (Förster et al., 2007). No obstante, teniendo en cuenta los objetivos de los experimentos planteados y la cantidad de datos obtenidos a través de medición directa con el sensor láser LRF no se consideró necesaria su aplicación en esta fase del trabajo. Posteriormente, en la sección final de este capítulo, se retoma la idea del incremento del número o cantidad de datos desde el punto de vista de validar el sistema allí propuesto.

4.3.2. Modelar y entrenar varios modelos de Inteligencia Artificial

En esta sección se presentan los resultados y el trabajo realizado para abordar el problema de localización del agente robótico móvil usando un clasificador basado en técnicas de Inteligencia Artificial. Se han realizado diversos experimentos y pruebas incluyendo modelos como Máquinas de Vector de Soporte, K-Nearest Neighbors y Redes Neuronales Artificiales (Perceptrón Multicapas, MLP). Todos estos algoritmos y modelos han sido programados, diseñados y entrenados en Python por el autor usando la plataforma de Google Colab.

Support Vector Machines (SVM) – Máquinas de Vector de Soporte

Las Máquinas de Vector de Soporte (SVM: Support Vector Machines) – también conocidas como Redes de Vector de Soporte (SVN) o Clasificadores de Vector de Soporte (SVC) – han destacado desde sus orígenes por su buena capacidad de generalización (Cortes y Vapnik, 1995). El método de aprendizaje que sustenta las Máquinas de Vector de Soporte se basa en la búsqueda de hiperplanos que separen los distintos conjuntos de datos representativos de las clases involucradas, usando para ello lo que se conoce como un espacio de propiedades o atributos (*"feature space"*). En este trabajo, se han utilizado como variables predictoras o *"features"* del modelo las nueve características estadísticas (*"engineered features"*) antes mencionadas. En la Figura 14, se pueden ver valores de estas *"features"* (sin ser normalizados) para cinco ejemplos de las mediciones obtenidas en el salón (R01S) de la vivienda 1SY. Estos valores se han usado para entrenar los modelos de SVM.

	room_ID	house_ID	var	mean	std	median	max	min	range	kurtosis	skewness
0	R01S	1SY	1.280102	1.727656	1.129888	2.177	4.305	0.0	4.305	-0.706080	-0.508396
1	R01S	1SY	1.424719	1.659540	1.191901	2.147	4.327	0.0	4.327	-0.924417	-0.316680
2	R01S	1SY	1.420910	1.645136	1.190304	2.115	4.316	0.0	4.316	-0.976682	-0.320166
3	R01S	1SY	1.554625	1.462976	1.245024	1.966	4.339	0.0	4.339	-1.183259	-0.013393
4	R01S	1SY	1.528801	1.566290	1.234636	2.003	4.332	0.0	4.332	-1.078383	-0.137667

Figura 14. Ejemplos del *"dataset"* de *"features"* estadísticas (vivienda 1SY).
Fuente: Elaboración propia.

Los *datasets* con estas *"features"* calculadas en base a las mediciones obtenidas en cada vivienda se han dividido de forma aleatoria para designar un 70% - 30% de los registros como datos de training y test, respectivamente. Al realizar este *split*, se ha asegurado que se mantiene balanceada la representación de cada tipo de estancia en los datos (Tabla 2) de manera que nuestros datos no sesguen el clasificador que vamos a entrenar.

Tabla 2. Representatividad de los datos de training y test

Room ID	% del total	% test	% training
R01S	16,67%	17,59%	16,27%
R02C	16,67%	18,33%	15,95%
R03H	16,67%	15,00%	17,38%
R04P	16,67%	18,52%	15,87%
R05D	16,67%	14,07%	17,78%
R06B	16,67%	16,48%	16,75%

Representatividad de los datos de training y test para las estancias de las viviendas durante uno de los experimentos.
Fuente: Elaboración propia.

Haciendo uso de los datos anteriores se ha entrenado un modelo de Máquina de Vector de Soporte (SVM) usando el módulo LinearSVC de la librería “Scikit Learn” de aprendizaje automático en Python. El modelo implementado ha permitido obtener un desempeño elevado en la tarea de clasificación que pretende informar sobre la localización del agente robótico móvil, siendo evaluado con datos de prueba “no vistos” por el modelo durante el entrenamiento. En este experimento los datos usados tanto para el entrenamiento como para las pruebas pertenecían a la misma vivienda. Los resultados de uno de estos experimentos se muestran en la Figura 15, alcanzando una precisión (*accuracy*) de 0,6518 (65,18%) en los datos de prueba y de 0,6674 (66,74%) en datos de entrenamiento.

```
Confusion Matrix:
-----------------
['R01S' 'R02C' 'R03H' 'R04P' 'R05D' 'R06B']
[[78  1 15  5  0  0]
 [ 3 25  7 27 23  5]
 [ 9  0 63  5  4  0]
 [ 0  8 15 61  6  6]
 [ 0 35  0  5 44  3]
 [ 0  0  1  3  2 81]]
-----------------

Confusion Matrix (Normalized):
------------------------------
[[0.78787879 0.01010101 0.15151515 0.05050505 0.         0.        ]
 [0.03333333 0.27777778 0.07777778 0.3        0.25555556 0.05555556]
 [0.11111111 0.         0.77777778 0.0617284  0.04938272 0.        ]
 [0.         0.08333333 0.15625    0.63541667 0.0625     0.0625    ]
 [0.         0.40229885 0.         0.05747126 0.50574713 0.03448276]
 [0.         0.         0.01149425 0.03448276 0.02298851 0.93103448]]
------------------------------
```

Precisión (*accuracy*) del modelo en datos de prueba: 0,6518 (65,18%)

Figura 15. Matriz de confusión de resultados del modelo LinearSVC (1SY).
Entrenado y probado con datos de la vivienda 1SY. Fuente: Elaboración propia.

En un segundo experimento se entrenó el modelo con datos de una vivienda y se probó con los datos de otras viviendas. Es decir, se probó el agente usando datos que no habían sido vistos durante el entrenamiento para comprobar la capacidad de generalización a otras estancias de uso similar (habitaciones, pasillos, etc.) en otras viviendas. Estos experimentos produjeron peores resultados. La Figura 16 muestra la matriz de confusión y la precisión (*accuracy*) para un modelo entrenado en la vivienda 3ME y probado en la vivienda 4TR.

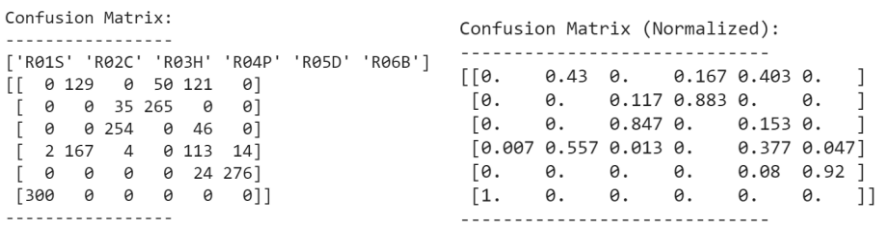


Figura 16. Matriz de confusión del modelo LinearSVC (3ME/4TR).
Entrenado con datos de la vivienda 3ME y probado con datos de la vivienda 4TR.
Fuente: Elaboración propia.

Modelo SVM (Linear SVC) entrenado con todos los registros

En una tercera ronda de experimentos se entrenó un modelo de Máquina de Vector de Soporte (SVM) usando un subconjunto balanceado de datos de todas las estancias de todas las viviendas. Este experimento resultaba ser importante para explorar la capacidad de estos modelos de asociar de manera efectiva atributos de carácter físico a descriptores basados en un conocimiento semántico. En este caso, vuelve a ser especialmente importante garantizar que los datos de entrenamiento no están sesgados y, por tanto, incluyen una muestra representativa y diversa de cada tipo de habitación que se quiere modelar, aun cuando en este experimento se trata de datos provenientes de todas las viviendas.

Tabla 3. División de los datasets para el modelo integrado

% ejemplos por estancia	Ejemplos totales: 7.200	Ejemplos para training: 5.040	Ejemplos para test: 2.160
Room ID	% del total	% training	% test
R01S	16,67%	16,23%	17,69%
R02C	16,67%	17,06%	15,74%
R03H	16,67%	16,35%	17,41%
R04P	16,67%	16,59%	16,85%
R05D	16,67%	16,85%	16,25%
R06B	16,67%	16,92%	16,06%

Porcentajes de representatividad de las seis estancias en los datos de training/test para el modelo “integrado” con todas las viviendas y habitaciones. Fuente: Elaboración propia.

La Figura 17 muestra la matriz de confusión con los resultados de las pruebas realizadas con este clasificador, implementado con LinearSVC y entrenado con datos de todas las viviendas y todas las habitaciones. La precisión del modelo en datos de pruebas es de 0,517 (51.7%).

		CLASE PREDICCIÓN						
		R01S	R02C	R03H	R04P	R05D	R06B	TOTAL
CLASE REAL	R01S	251	16	72	29	6	8	382
	R02C	6	91	42	77	65	59	340
	R03H	72	6	216	60	15	7	376
	R04P	33	48	48	167	52	16	364
	R05D	12	60	31	62	128	58	351
	R06B	13	43	2	6	16	267	347
	TOTAL	387	264	411	401	282	415	2160

Precisión (*accuracy*) del modelo en datos de prueba: 0,517 (51,70%)

Figura 17. Matriz de confusión del modelo "integrado" (LinearSVC). Resultados del modelo entrenado con datos de todas las viviendas/habitaciones en datos de prueba. Fuente: Elaboración propia.

Los resultados de estos experimentos se han comparado con los de los experimentos anteriores, realizados con modelos entrenados con datos de cada vivienda y probados de forma cruzada con datos de otras viviendas. Esta comparación se muestra en la sección 4.4.

SVM-RBF (Usando una función de base radial "RBF")

Una de las características distintivas de las Máquinas de Vector de Soporte es el uso de funciones de núcleo (*kernel*) que permiten transformar las variables de entrada a un espacio de mayor dimensionalidad en el que estas pueden ser separadas cuando dicha separación no es efectiva con un hiperplano lineal. Así, hemos experimentado con el uso de un kernel de Función de Base Radial, y ciertamente los resultados muestran que mejora considerablemente el desempeño del clasificador integrado. Llegando a alcanzar, como se muestra en la Figura 18, un valor de precisión del 75,28% en los datos de prueba.

		CLASE PREDICCIÓN						
		R01S	R02C	R03H	R04P	R05D	R06B	TOTAL
CLASE REAL	R01S	284	20	34	37	5	2	382
	R02C	10	237	4	38	30	21	340
	R03H	18	14	301	17	18	8	376
	R04P	21	33	5	267	27	11	364
	R05D	1	52	13	19	247	19	351
	R06B	2	19	7	14	15	290	347
	TOTAL	336	375	364	392	342	351	2160

Precisión (*accuracy*) del modelo en datos de prueba: 0,7528 (75,28%)

Figura 18. Matriz de confusión del modelo "integrado" (SVM-RBF) Modelo SVM (con kernel RBF) entrenado con datos de todas las viviendas/habitaciones. Fuente: Elaboración propia.

Los K vecinos más próximos (K-Nearest Neighbors)

Un método de clasificación que destaca por su sencillez de comprensión es el de los K vecinos más próximos (en inglés: *"K-Nearest Neighbors", KNN*). Este método se basa en clasificar un ejemplo o muestra desconocida usando la distancia mínima de dicho ejemplo o patrón a K elementos de cada clase. La clase que incluya el subconjunto de K elementos más próximos al nuevo elemento será la que se use como resultado de la clasificación.

		CLASE PREDICCIÓN						
		R01S	R02C	R03H	R04P	R05D	R06B	TOTAL
CLASE REAL	R01S	297	15	32	26	3	9	382
	R02C	12	235	4	43	23	23	340
	R03H	15	12	316	12	17	4	376
	R04P	24	33	10	261	25	11	364
	R05D	8	38	11	17	265	12	351
	R06B	3	15	6	16	11	296	347
	TOTAL	359	348	379	375	344	355	2160

Precisión (*accuracy*) del modelo KNN en datos de prueba: 0,7731 (77,31%)

Figura 19. Matriz de confusión del modelo "integrado" (KNN, K=1) Modelo KNN entrenado usando datos de todas las viviendas y habitaciones. (Con K=1). Fuente: Elaboración propia.

La Figura 19 muestra los resultados de uno de los modelos KNN desarrollados. Otros experimentos con modelos KNN también reportaron buenos resultados (Tabla 4).

Tabla 4. Precisión del modelo KNN para varios valores de K.

Clasificador KNN	K=12	K=4	K=1
Precisión (*Accuracy*)	0.6837	0.7125	0.7731

Fuente: Elaboración propia.

Clasificador basado en Redes Neuronales (Perceptrón Multicapa)

Para el entrenamiento de modelos basados en redes neuronales, se utilizó el SGD "*Stochastic Gradient Descent*". Para trabajar con estos modelos, se dividieron los datos en tres conjuntos.

Tabla 5. Dimensionado de los *datasets* de training, validación y pruebas.

Dataset →	Total	Training	Validación	Prueba
% de los datos	100%	70%	20%	10%
Número de registros	7.200	5.040	1.440	720

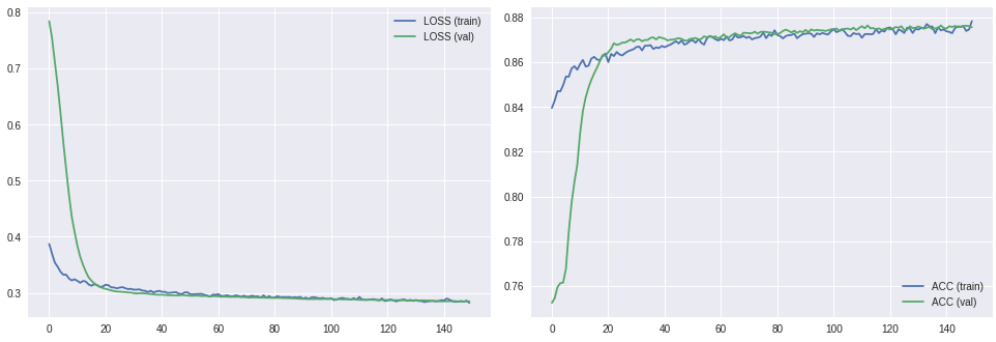
Fuente: Elaboración propia.

Con estos datos, se han diseñado y entrenado varios modelos de redes neuronales, usando varias arquitecturas e hiperparámetros. Los resultados varían notablemente en función de estas decisiones. Por ejemplo, la red que se presenta a continuación ha sido entrenada usando como variables o "*features"* las características estadísticas antes expuestas.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               1280
_________________________________________________________________
batch_normalization_v1 (Batc (None, 128)               512
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               66048
_________________________________________________________________
batch_normalization_v1_1 (Ba (None, 512)               2048
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                32832
_________________________________________________________________
batch_normalization_v1_2 (Ba (None, 64)                256
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 390
=================================================================
Total params: 103,366
Trainable params: 101,958
Non-trainable params: 1,408
```

Figura 20. Arquitectura de la red neuronal Perceptrón Multicapa (MLP). Modelo entrenado y utilizado en varios de los experimentos. Fuente: Elaboración propia.

Aunque aquí sólo se muestran los resultados de dos redes neuronales y de dos de sus entrenamientos, en los experimentos se realizaron numerosos entrenamientos usando distintos parámetros y adaptando la arquitectura de las redes con técnicas de regularización para minimizar la tendencia a caer en *overfitting*. Por ejemplo, el uso de *Batch Normalization* y *Dropout* se debe a estas razones. Cabe resaltar que los datos de entrada de esta red han sido las *"features"* estadísticas normalizadas como se describió en la Figura 13.



a) Evolución del LOSS (pérdida) en Training/Validación b) Evolución del ACCURACY (precisión)

Figura 21. Historial del entrenamiento del clasificador Perceptrón Multicapa (MLP). Entrenamiento del modelo de la Figura 20 durante 150 *epochs*, usando características (*"features"*) estadísticas normalizadas. Fuente: Elaboración propia.

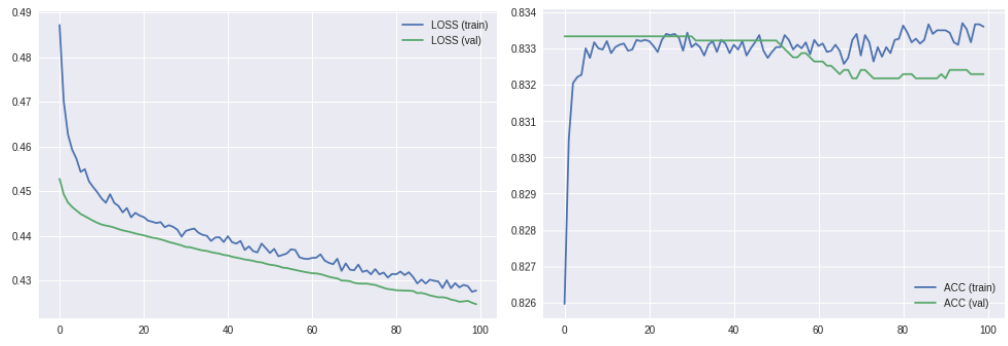
Se puede apreciar en la Figura 21, que los valores de la métrica de precisión (*accuracy*) en el conjunto de entrenamiento y de validación son respectivamente 0,878 y 0,876. No obstante, esta red cae en *overfitting* (memorizando los datos), hecho que se puede apreciar en el valor de precisión sobre los datos de prueba que es: 0,590. Los resultados de la clasificación en los datos de prueba se presentan en la matriz de confusión de la Figura 22.

		CLASE PREDICCIÓN						
		R01S	R02C	R03H	R04P	R05D	R06B	TOTAL
CLASE REAL	R01S	66	11	16	7	5	0	105
	R02C	0	52	8	16	24	13	113
	R03H	10	7	82	14	17	1	131
	R04P	11	20	3	61	13	5	113
	R05D	1	25	16	14	71	10	137
	R06B	1	11	0	5	11	93	121
	TOTAL	89	126	125	117	141	122	720

Precisión (*accuracy*) del modelo en datos de prueba: 0,590 (59%)

Figura 22. Matriz de confusión del clasificador Perceptrón Multicapa. Resultados del uso de la red neuronal de la Figura 20. Fuente: Elaboración propia.

El problema de *overfitting* se ve aún más acentuado con una red de neuronas similar a la anterior, pero usando las mediciones de distancias directamente como datos de entrada, en lugar de las *"features"* estadísticas usadas anteriormente. Esta red – usada a modo de comparación – tiene la misma arquitectura, excepto en la capa de entrada que incluye 681 neuronas para recibir el vector de mediciones del sensor láser. Aunque el entrenamiento de esta red converge en menos de 100 *epochs*, su desempeño/precisión ha sido de 0,832 y 0,833 respectivamente en validación y training, mientras que en los datos de prueba ha sido de tan solo 0,375. En este caso, el *dataset* también ha sido dividido en la proporción 70%-20%-10% para entrenamiento-validación-pruebas, respectivamente.



a) Evolución del LOSS (pérdida) en Training/Validación b) Evolución del ACCURACY (precisión)

Figura 23. Historial del entrenamiento del MLP usando mediciones de distancia. Perceptrón Multicapa similar al clasificador de la Figura 20, entrenado con datos de mediciones de distancia durante 100 *epochs*. Fuente: Elaboración propia.

4.4. Resultados cuantitativos: comparativa del desempeño de los modelos

En este apartado se muestra una comparativa de los resultados de más de 65 pruebas cruzadas, realizadas con modelos entrenados con los datos de una vivienda y probados con los datos de todas las viviendas. Estos experimentos y pruebas se han realizado con todos los modelos, incluyendo sus variantes (SVM con LinearSVC, SVM-RBF, KNN, ANN, etc.).

Modelo LinearSVC		Probado con datos de las VIVIENDAS				
		1SY	2CA	3ME	4TR	TODAS
Entrenado con datos de las VIVIENDAS	1SY	0,652	0,547	0,439	0,285	
	2CA	0,532	0,712	0,463	0,308	
	3ME	0,530	0,432	0,586	0,395	
	4TR	0,315	0,292	0,371	0,544	
	TODAS					0,518

Modelo SVM-RBF		Probado con datos de las VIVIENDAS				
		1SY	2CA	3ME	4TR	TODAS
Entrenado con datos de las VIVIENDAS	1SY	0.867	0,502	0,377	0,292	
	2CA	0,438	0,846	0,389	0,262	
	3ME	0,359	0,378	0,848	0,433	
	4TR	0,302	0,322	0,480	0,767	
	TODAS					0,752

Modelo KNN, K=4		Probado con datos de las VIVIENDAS				
		1SY	2CA	3ME	4TR	TODAS
Entrenado con datos de las VIVIENDAS	1SY	0,855	0,492	0,385	0,290	
	2CA	0,466	0,809	0,376	0,252	
	3ME	0,363	0,369	0,867	0,407	
	4TR	0,347	0,305	0,460	0,785	
	TODAS					0,773

Figura 24. Comparativa del desempeño de varios modelos (+50 pruebas cruzadas). Métrica usada: precisión ("*accuracy*"), calculada como la ratio entre aciertos y el total. Fuente: Elaboración propia.

En base a estos resultados, se pueden observar varios hechos destacables:

- Los mejores resultados se obtienen con los modelos SVM-RBF y KNN, K=4.
- En general, todos los modelos entrenados con datos de una sola vivienda, "sufren" una pérdida considerable de precisión al ser probados ("puestos en producción") con los datos de las otras viviendas. Este hecho invita a reflexionar sobre la capacidad de estos modelos (con estos datos y *"features"*) para generalizar el aprendizaje de etiquetas semánticas en base a propiedades físicas de las habitaciones de las viviendas.
- En relación con el punto anterior, aunque es de esperar que un modelo entrenado con los datos de todas las viviendas tenga un mejor desempeño, en la práctica lo que se ha comprobado con los experimentos es que los modelos "integrados" tienen aproximadamente un desempeño similar al "peor" de los modelos de este tipo. Ver los

valores de las esquinas (celdas inferiores a la derecha) de las tablas de la Figura 24 y comparar estos valores con la diagonal de cada tabla, respectivamente.

- También son destacables las diferencias entre los resultados de las viviendas 3ME y 4TR. A pesar de ser prácticamente iguales en términos de tamaños y distribución física de sus habitaciones (ver planos en la Figura 6), los resultados cruzados entre estas dos viviendas empeoran igual que en el resto de los casos. La explicación que proponemos se debe a diferencias en el mobiliario y objetos personales diferentes que tiene cada familia en sus distintas estancias.

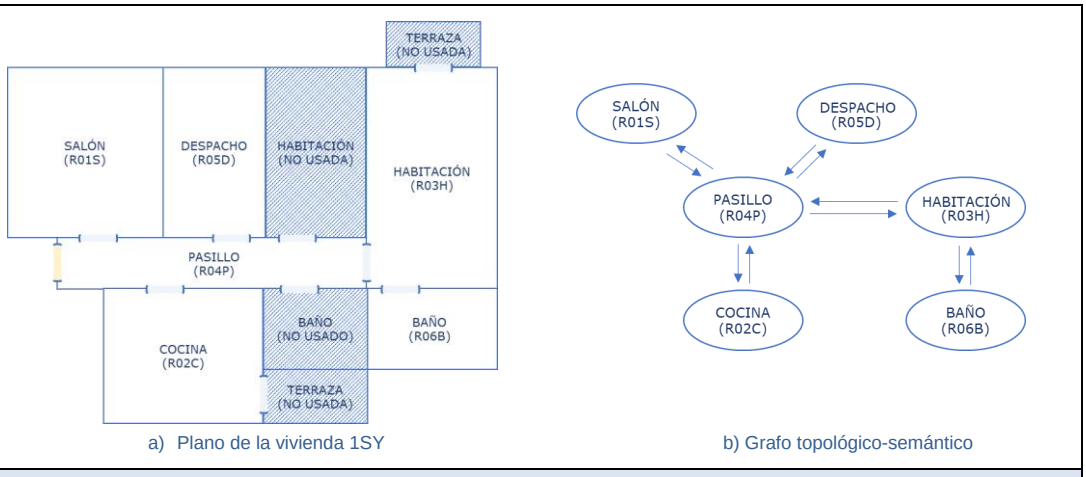
Si bien algunos de los resultados anteriores pudieran considerarse razonablemente buenos, por ejemplo, para el modelo KNN con valores de precisión superiores al 85%, en esta fase del trabajo realizado no nos hemos dado por satisfechos, y hemos explorado otras ideas y opciones para mejorar el desempeño de estos modelos. Si bien estas actividades y reflexiones de mejora han surgido del análisis comparativo de los datos, los resultados que se han derivado de ellas tienen suficiente entidad para ser explicados y documentados de forma más detallada en los siguientes apartados.

4.5. Mejora del desempeño del clasificador con un grafo topológico-semántico

En este apartado, se explican varias ideas con las que se ha experimentado para mejorar el desempeño de los clasificadores y así tener un mejor resultado en la localización semántica del agente robótico móvil. Si bien la propuesta que se describe a continuación es aplicable a todos los modelos anteriores se presenta usando un clasificador SVM para simplificar la exposición. El enfoque propuesto utiliza el conocimiento del entorno del robot, extraído de la topología semántica de dicho entorno, para ajustar y mejorar la predicción de su ubicación.

Es decir, teniendo en consideración las restricciones que impone la arquitectura del entorno (en estos experimentos: las viviendas y las conexiones reales entre habitaciones, pasillos, etc.), en la cual se encuentra y por donde realiza su recorrido el agente robótico móvil, podemos corregir una parte de las predicciones realizadas erróneamente por el clasificador utilizado, en este caso la Máquina de Vector de Soporte (SVM, con *kernel* linear).

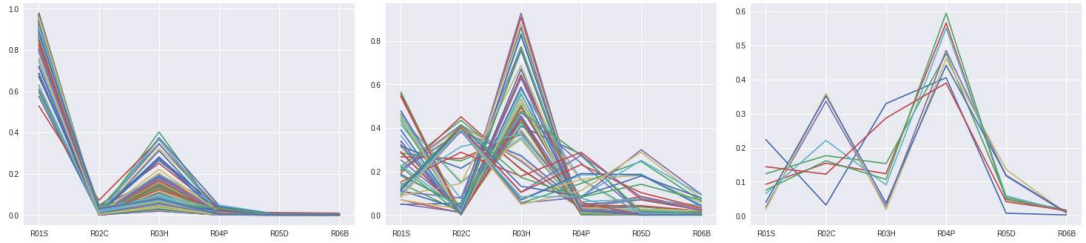
La idea para realizar esta corrección se basa en que, si el agente está en una estancia determinada, desde dicha ubicación sólo sería posible permanecer en la misma estancia o sería posible el desplazamiento a algunas, pero no a todas las ubicaciones de la vivienda. Para visualizar dichas opciones se propone usar un grafo topológico-semántico (Figura 25), que como se explica más adelante se puede generar automáticamente durante la recogida de datos en el entorno o alternativamente, se puede diseñar manualmente.



a) Plano de la vivienda 1SY b) Grafo topológico-semántico

Figura 25. Representaciones del entorno (vivienda 1SY).
a) plano físico, b) grafo topológico-semántico. Fuente: Elaboración propia.

Por ejemplo, considerando el entorno de la vivienda 1SY (Figura 25), si sabemos que la última posición conocida del agente robótico móvil ha sido el salón (R01S), entonces la siguiente posición sólo podría ser el propio salón o el pasillo (R04P). Cualquier otra predicción que dé el clasificador en esa situación puede ser considerada como un error y pudiera ser reclasificada. Por ejemplo, ante un error así, se pudiera tomar la decisión de reclasificar la predicción como salón – asumiendo que el agente continúa estando en el salón. Si bien es posible que también sea justamente el momento en el que el agente se haya movido al pasillo (R04P), lo más probable es que si así ha sido en los siguientes momentos la "huella" (vector de *"features"*) del entorno provoque la clasificación correcta de estar en el pasillo.



a) Número de clasificaciones correctas (iguales a R01S): 235 b) Número de clasificaciones incorrectas (diferentes a R01S y R04P): 56 c) Número de clasificaciones incorrectas pero factibles (iguales a R04P): 9

Figura 26. Probabilidades obtenidas con SVM en R01S (vivienda 1SY).
Resultados del clasificador Support Vector Machine. Fuente: Elaboración propia.

En la Figura 26 se muestran las probabilidades de cada clase/estancia que da el clasificador sobre todas las muestras tomadas en el salón (R01S) de la vivienda 1SY. En la Figura 27 se muestran los resultados de las posibles correcciones considerando todos los datos del entorno de la vivienda 1SY. Como se deriva de estos datos, el uso del conocimiento del entorno a través de este grafo semántico permite aumentar la precisión del clasificador del 73,05% al 90,22%. Como se puede observar la mejora del desempeño es diferente para cada estancia.

VIVIENDA 1SY							
SVM (kernel = linear)	**R01S**	**R02C**	**R03H**	**R04P**	**R05D**	**R06B**	**TODAS**
Predicción correcta del SVM	235	103	269	208	232	268	1315
Pred. incorrecta, pero factible (ej. R04P)	9	76	6	68	17	0	176
Pred. incorrecta, pero imposible	56	121	25	24	51	32	309
Número TOTAL de muestras	300	300	300	300	300	300	1800
% de error descartable	18,67%	40,33%	8,33%	8,00%	17,00%	10,67%	17,17%
% de precisión (sólo SVM)	78,33%	34,33%	89,67%	69,33%	77,33%	89,33%	73,06%
% precisión (SVM+Grafo semántico)	97,00%	74,67%	98,00%	77,33%	94,33%	100,00%	90,22%

Figura 27. Mejora del clasificador usando un grafo topológico-semántico (1SY).
Comparativa de resultados del clasificador SVM en la vivienda 1SY. Fuente: Elaboración propia.

4.6. *Yielding Associations Network*: sistema mejorado de localización semántica

En base a las observaciones que se han obtenido con la experimentación realizada, se propone considerar la operación de mejora o revisión de la predicción del clasificador como una función que ajusta los resultados de dicha clasificación, sometiéndolos al escrutinio del conocimiento semántico del entorno físico. En este sentido, es como si el sistema propuesto considerara la asociación "inicial" que resulta del clasificador como una "propuesta" sujeta a revisión. Tomándose la libertad de cambiar dicha asociación inicial si es contradictoria con el conocimiento del entorno. En este caso, el sistema propuesto genera una nueva asociación.

Así, podemos definir formalmente este sistema como una red formada por varias partes interrelacionadas que denominamos "*Yielding Associations Network*" (Red Generadora de Asociaciones). Esta red incluye un primer componente que tiene la función de hacer una "propuesta" de clasificación-asociación, y un segundo componente que funciona como un operador de "Filtro de Contexto". Este Filtro de Contexto es una función que tiene como entradas el grafo semántico del entorno, la última ubicación conocida del agente robótico móvil y la predicción del clasificador para la ubicación actual, y tiene como salida la localización semántica "revisada". A continuación, se define formalmente este Filtro de Contexto.

Ecuación 1. Filtro de Contexto Of_C ("lazy")

$$\text{a)}\ Of_C(y_i, u_{i-1}, G_{TS}) \rightarrow y_i^* \quad \begin{cases} y^*_i = y_i & \text{si } y_i = u_{i-1} \\ y^*_i = y_i & \text{si } y_i \in G_{TS}(u_{i-1}) \\ y^*_i = u_{i-1} & \text{si } y_i \notin G_{TS}(u_{i-1}) \end{cases} \qquad (1)$$

$$\text{b)}\ u_i \leftarrow y_i^*$$

En la Ecuación 1, y_i es la predicción del clasificador, u_{i-1} es la ubicación anterior (en el momento i-1), G_{TS} es el grafo topológico-semántico y $G_{TS}(u_i)$ representa el conjunto de nodos (estancias/habitaciones) conectados directamente con u_i. El resultado de este operador es la ubicación revisada o ajustada y_i^* (posiblemente mejorada) que se obtiene en la Ecuación 1a.

Esta ubicación ajustada y_i^* se guarda (Ecuación 1b) como nueva ubicación conocida en la "memoria" del sistema, usándose de forma recursiva en pasos posteriores. La Figura 28 ilustra el flujo de información entre las partes del sistema de localización semántica propuesto.

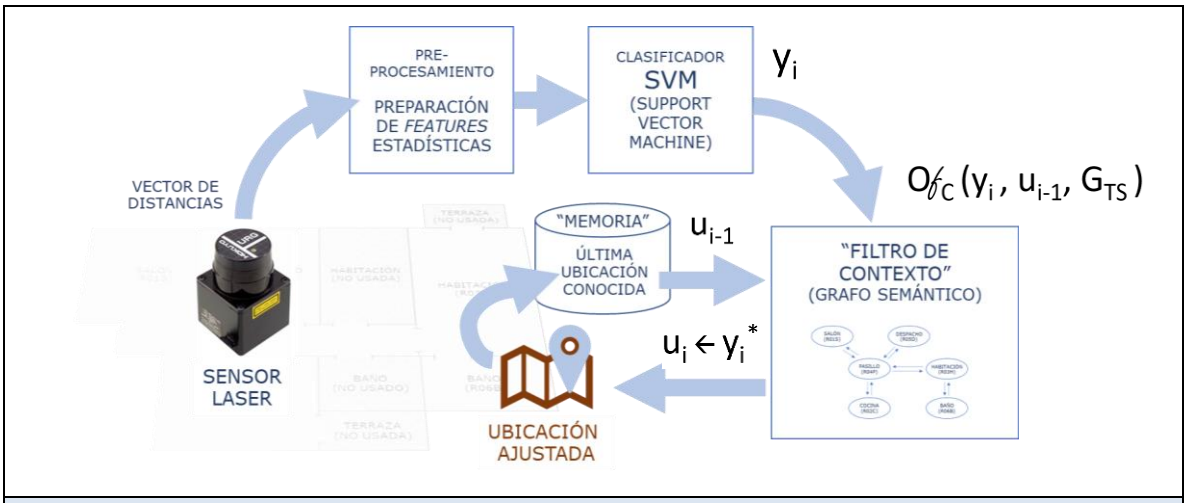
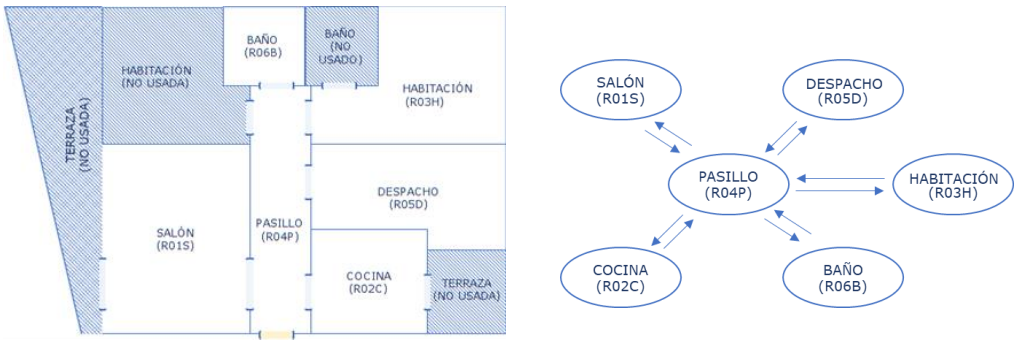


Figura 28. Sistema de localización semántica: *Yielding Associations Network.*
Usando un clasificador SVM y un operador de "Filtro de Contexto". Fuente: Elaboración propia.

Como es de esperar las relaciones o conexiones entre las estancias del entorno (vivienda) afectan al número de predicciones erróneas que se pueden corregir usando el operador de Filtro de Contexto propuesto. Por ejemplo, si una estancia está conectada con todas las demás, como suele ocurrir en ocasiones con los pasillos, ninguna predicción del clasificador que corresponda a dicha estancia podrá ser ajustada o corregida por el operador. De hecho, varias de las viviendas incluidas en los experimentos tienen este tipo de pasillos. En la Figura 29 se muestra el plano de una de ellas y el correspondiente grafo topológico-semántico.

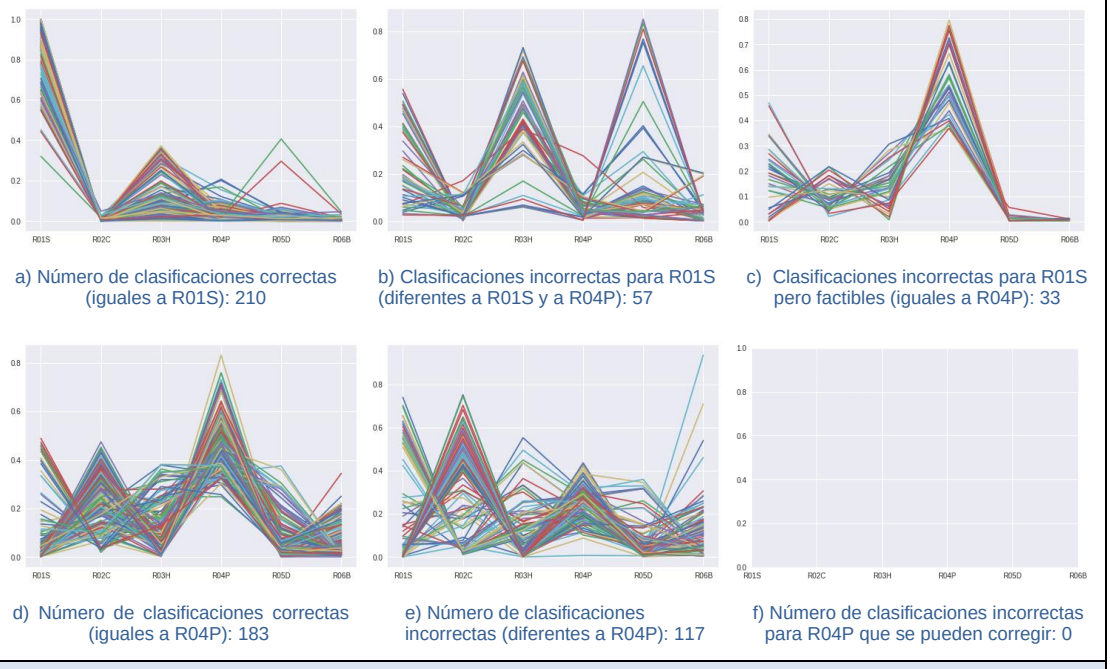


a) Plano de la vivienda 3ME b) Grafo topológico-semántico

Figura 29. Representaciones del entorno (vivienda 3ME)
a) plano físico, b) grafo topológico-semántico. Fuente: Elaboración propia.

La Figura 30 (d, e, f) muestra los resultados del clasificador SVM, agrupados como paso previo a la aplicación del Filtro de Contexto a las predicciones del pasillo (R04P) de la vivienda 3ME. Como se aprecia, no se pueden mejorar las predicciones realizadas en el pasillo. Esto se debe

a que el pasillo está conectado con todas las otras estancias de la vivienda y cualquier predicción del clasificador SVM sería un resultado factible en este entorno.



a) Número de clasificaciones correctas (iguales a R01S): 210

b) Clasificaciones incorrectas para R01S (diferentes a R01S y a R04P): 57

c) Clasificaciones incorrectas para R01S pero factibles (iguales a R04P): 33

d) Número de clasificaciones correctas (iguales a R04P): 183

e) Número de clasificaciones incorrectas (diferentes a R04P): 117

f) Número de clasificaciones incorrectas para R04P que se pueden corregir: 0

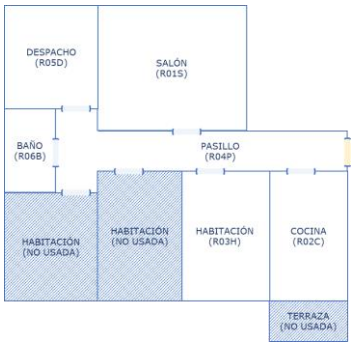
Figura 30. Resultados del clasificador SVM en R01S y R04P (vivienda 3ME).
Fuente: Elaboración propia.

A pesar de la imposibilidad de reducir los posibles fallos en el pasillo, como muestra la Figura 31, los resultados de aplicar el Filtro de Contexto en la vivienda 3ME ofrece una mejora global que permite aumentar la precisión en un 13%, pasando en total del 68,33% a 81,33%.

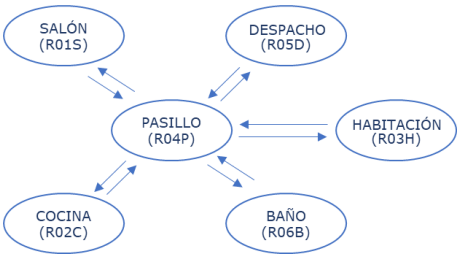
VIVIENDA 3ME							
SVM(kernel=linear)	**R01S**	**R02C**	**R03H**	**R04P**	**R05D**	**R06B**	**TODAS**
Predicción correcta del SVM	210	131	237	183	221	248	1230
Pred. incorrecta, pero factible (ej. R04P)	33	97	32	117	37	20	336
Pred. incorrecta, pero imposible	57	72	31	0	42	32	234
Número TOTAL de muestras	300	300	300	300	300	300	1800
% de error descartable	19,00%	24,00%	10,33%	0,00%	14,00%	10,67%	13,00%
% de precisión (sólo SVM)	70,00%	43,67%	79,00%	61,00%	73,67%	82,67%	68,33%
% precisión (SVM+Grafo semántico)	89,00%	67,67%	89,33%	61,00%	87,67%	93,33%	81,33%

Figura 31. Mejora del clasificador SVM con el grafo topológico-semántico (3ME).
Resultados en la vivienda 3ME. Fuente: Elaboración propia

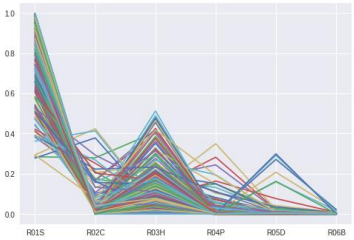
La topología de las habitaciones de las otras dos viviendas incluidas en los experimentos es similar a la de la vivienda anterior. Estas viviendas tienen un pasillo que se conecta con todas las habitaciones y, por tanto, para éstos no es posible mejorar los resultados del clasificador. No obstante, como en el caso anterior los resultados de mejora globales son buenos, llegando a estar esta mejora entre el 17% y hasta casi un 27%, respectivamente. Las Figuras 32 y 33 presentan estos resultados resaltando ejemplos de estancias donde las mejoras son significativas.



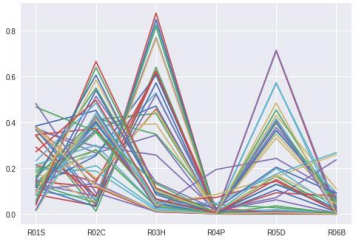
a) Plano de la vivienda 2CA



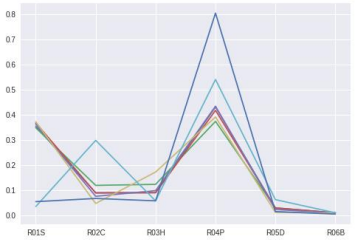
b) Grafo topológico-semántico



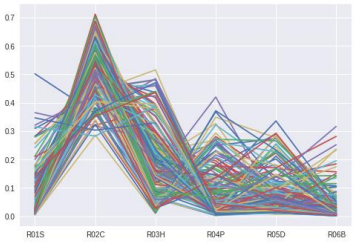
c) Número de clasificaciones correctas (iguales a R01S): 240



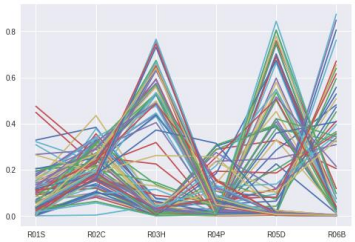
d) Clasificaciones incorrectas para R01S (diferentes a R01S y a R04P): 53



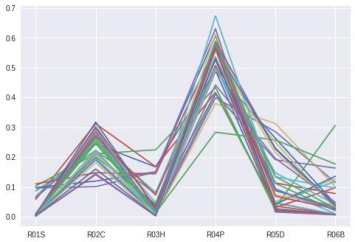
e) Clasificaciones incorrectas para R01S pero factibles (iguales a R04P): 7



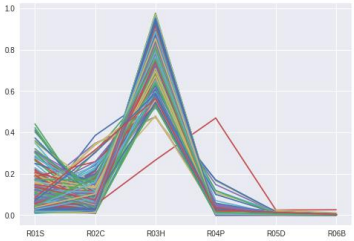
f) Número de clasificaciones correctas (iguales a R02C): 171



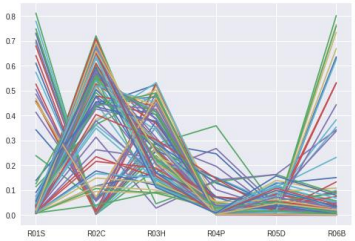
g) Clasificaciones incorrectas para R02C (diferentes a R02C y a R04P): 95



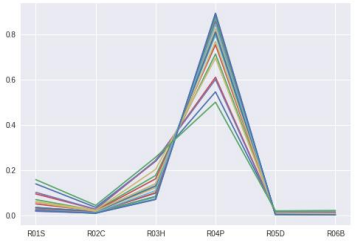
h) Clasificaciones incorrectas para R02C pero factibles (iguales a R04P): 34



i) Número de clasificaciones correctas (iguales a R03H): 177



j) Clasificaciones incorrectas para R03H (diferentes a R03H y a R04P): 103

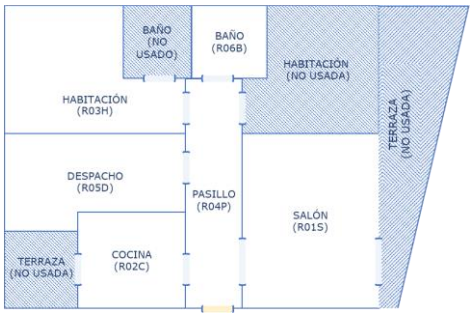


k) Clasificaciones incorrectas para R03H pero factibles (iguales a R04P): 20

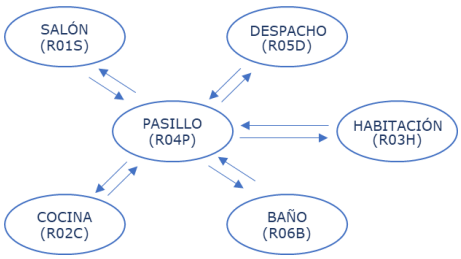
VIVIENDA 2CA							
SVM(kernel=linear)	**R01S**	**R02C**	**R03H**	**R04P**	**R05D**	**R06B**	**TODAS**
Predicción correcta del SVM	240	171	177	238	265	270	1361
Pred. incorrecta, pero factible (ej. R04P)	7	34	20	62	9	1	133
Pred. incorrecta, pero imposible	53	95	103	0	26	29	306
Número TOTAL de muestras	300	300	300	300	300	300	1800
% de error descartable	17,67%	31,67%	34,33%	0,00%	8,67%	9,67%	17,00%
% de precisión (sólo SVM)	80,00%	57,00%	59,00%	79,33%	88,33%	90,00%	75,61%
% precisión (SVM+Grafo semántico)	97,67%	88,67%	93,33%	79,33%	97,00%	99,67%	92,61%

f) Resumen de mejoras de la precisión del clasificador SVM, usando el Filtro de Contexto (vivienda 2CA).

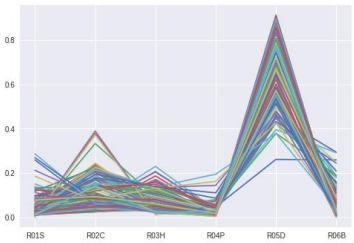
Figura 32. Mejora del clasificador SVM con el grafo topológico-semántico (2CA). Resultados obtenidos en la vivienda 2CA. Fuente: Elaboración propia.



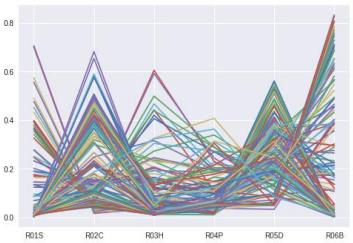
a) Plano de la vivienda 4TR



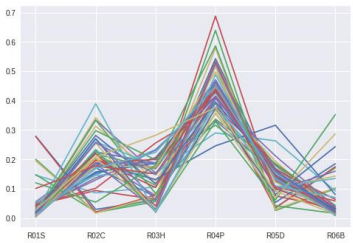
b) Grafo topológico-semántico



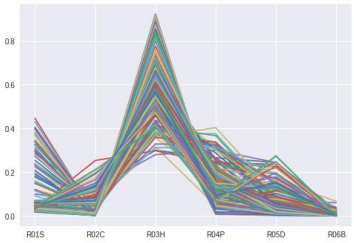
c) Número de clasificaciones correctas (iguales a R05D): 102



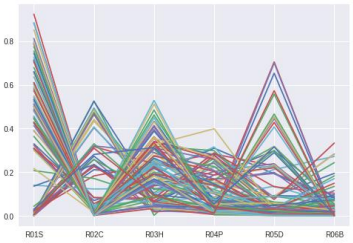
d) Clasificaciones incorrectas para R05D (diferentes a R05D y a R04P): 150



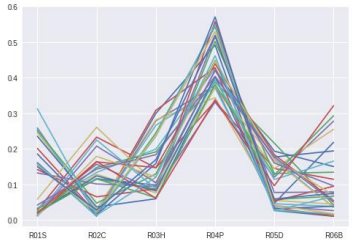
e) Clasificaciones incorrectas para R05D pero factibles (iguales a R04P): 48



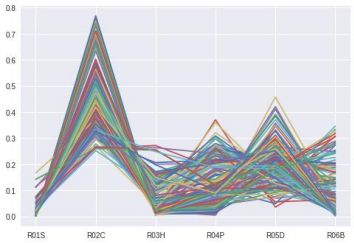
f) Número de clasificaciones correctas (iguales a R03H): 164



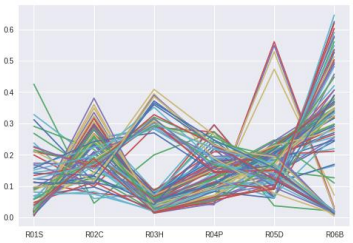
g) Clasificaciones incorrectas para R03H (diferentes a R03H y a R04P): 105



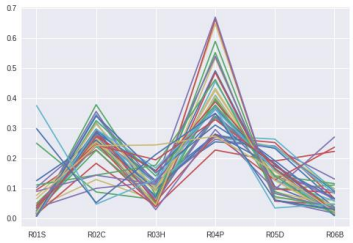
h) Clasificaciones incorrectas para R03H pero factibles (iguales a R04P): 31



i) Número de clasificaciones correctas (iguales a R02C): 158



j) Clasificaciones incorrectas para R02C (diferentes a R02C y a R04P): 99



k) Clasificaciones incorrectas para R02C pero factibles (iguales a R04P): 43

VIVIENDA 4TR							
SVM(kernel=linear)	**R01S**	**R02C**	**R03H**	**R04P**	**R05D**	**R06B**	**TODAS**
Predicción correcta del SVM	221	158	164	102	102	209	956
Pred. incorrecta, pero factible (ej. R04P)	24	43	31	198	48	17	361
Pred. incorrecta, pero imposible	55	99	105	0	150	74	483
Número TOTAL de muestras	300	300	300	300	300	300	1800
% de error descartable	18,33%	33,00%	35,00%	0,00%	50,00%	24,67%	26,83%
% de precisión (sólo SVM)	73,67%	52,67%	54,67%	34,00%	34,00%	69,67%	53,11%
% precisión (SVM+Grafo semántico)	92,00%	85,67%	89,67%	34,00%	84,00%	94,33%	79,94%

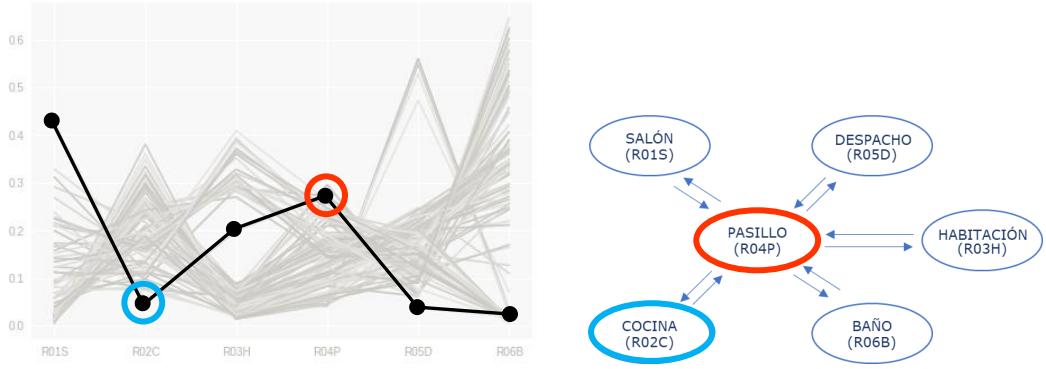
l) Resumen de mejoras de la precisión del clasificador SVM, usando el Filtro de Contexto (vivienda 4TR).

Figura 33. Mejora del clasificador SVM con el grafo topológico-semántico (4TR). Resultados obtenidos en la vivienda 4TR. Fuente: Elaboración propia.

4.7. Una variante “aventurera” del Filtro de Contexto

En la Ecuación 1, el operador de Filtro de Contexto se ha definido de forma tal que corrige cualquier predicción que sea incompatible con las conexiones de la última ubicación conocida en el grafo topológico-semántico, asignándole la ubicación anterior conocida. Esto pudiera entenderse como una variante “rezagada” o “perezosa” (“*lazy*”) del método de ajuste, ya que ante una predicción incorrecta se muestra “reticente” a moverse a otra posible posición. Sin embargo, también se pudiera plantear una variante que sea más “osada” o “aventurera” (“*gambler*”) de modo que asuma riesgos dándole una “segunda oportunidad” al clasificador en el caso de una predicción contradictoria. En este caso, se podría seleccionar la siguiente opción de nodo del grafo o ubicación “viable” que el clasificador considere más probable.

En la Figura 34a se muestra un ejemplo de instancia o registro en el que el sistema propuesto tendría un desempeño diferente dependiendo de la variante usada (*“lazy”* o *“gambler”*). Considerando que el agente robótico móvil estaba (venía) de una ubicación en la cocina (R02C), serían válidas las predicciones de continuar en la cocida o moverse al pasillo R04P.



a) Ejemplo de clasificación incorrecta para R02C (la predicción del SVM fue R01S).

b) Grafo topológico-semántico

Figura 34. Diferencias en la clasificación dependiendo del filtro (*Lazy* VS. *Gambler*). Clasificaciones incorrectas, pero mejorables con el Filtro de Contexto (vivienda 4TR). Fuente: Elaboración propia.

Los datos que se muestran en la Figura 34.a son los mismos que aparecen en la Figura 33.j y corresponden a las predicciones del clasificador SVM para entradas que deberían haber sido clasificadas como cocina (R02C), pero no ocurrió así. Como el clasificador SVM nos devuelve el vector con las probabilidades de que el ejemplo pertenezca a cada clase, la variante “aventurera” del ajuste se puede plantear como la habitación para la cual la probabilidad es máxima, sujeta a que sea una habitación válida en el grafo. Para el vector destacado en la Figura 34.a, con la variante “aventurera” obtendríamos la clasificación R04P, mientras que con la variante “rezagada” obtendríamos la clasificación R02C.

La variante “aventurera” del Filtro de Contexto se puede definir a través de la Ecuación 2.

Ecuación 2. Filtro de Contexto O^*f_C (“gambler”)

$$\begin{aligned} &a.)\ O^*f_C(P_i, u_{i-1}, M_{TS}) \rightarrow y_i^* \\ &b.)\ u_i \leftarrow y_i^* \end{aligned} \quad \Big\} = \text{argmax}(P_i \circ M_{TS}(u_{i-1})) \qquad (2)$$

Donde P_i es un vector N-dimensional con las probabilidades de predicción del clasificador para cada estancia, N es el número de habitaciones (en nuestro caso N=6), u_{i-1} es la ubicación anterior (en el momento i-1), M_{TS} es una matriz de N x N y componentes iguales a 0 o 1, indicando las conexiones directas entre estancias (Figuras 35.c y 35.d). $M_{TS}(u_i)$ es una fila de la matriz anterior (un vector) que contiene todas las estancias que están conectadas directamente con el espacio u_i. El operador $\circ$ es el producto de Hadamard que consiste en obtener un nuevo vector mediante la multiplicación elemento a elemento de dos vectores. Finalmente, *argmax* nos devuelve el índice o identificador de la habitación para la cual la componente del vector resultante del producto anterior alcanza su valor máximo.

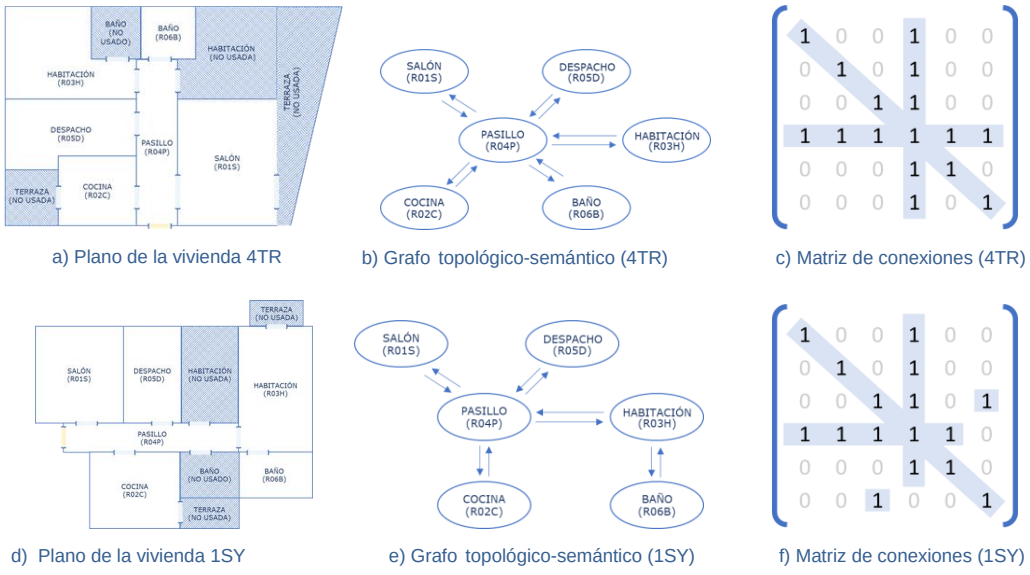


Figura 35. Representaciones del conocimiento del entorno.
Planos, grafos topológicos-semánticos y matrices de conexiones. Fuente: Elaboración propia.

El planteamiento de la variante “aventurera” del Filtro de Contexto como una operación con vectores/matrices, permite integrar fácilmente este filtro como última capa del componente predictor del sistema cognitivo propuesto. Recordamos que si bien hemos usado una Máquina de Vector de Soporte (SVM) para describir los resultados, el sistema propuesto se puede usar igualmente con cualquier clasificador. Por ejemplo, se puede usar con una red neuronal que incluya una capa de salida tipo *softmax* para obtener el vector de probabilidades de pertenencia a cada clase o con un modelo *ensemble* de clasificadores.

La Figura 36 muestra una comparación del desempeño de ambas variantes del Filtro de Contexto (“rezagado” VS. “aventurero”) en la habitación R03H de la vivienda 4TR.

Cinco modelos SVM entrenados en VIVIENDA 4TR y probados con datos de R03H						
Resultados usando Método LAZY	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	**% medio**
Predicción correcta del SVM	164	154	139	153	161	51,40%
Pred. incorrecta, pero factible (ej. R04P)	31	24	22	25	29	8,73%
Pred. incorrecta, corregidas como R03H	105	122	139	122	110	39,87%
Número TOTAL de muestras	300	300	300	300	300	**Promedio**
% de error descartable	35,00%	40,67%	46,33%	40,67%	36,67%	39,87%
% de precisión (sólo SVM)	54,67%	51,33%	46,33%	51,00%	53,67%	51,40%
% precisión (Método LAZY)	89,67%	92,00%	92,67%	91,67%	90,33%	91,27%
MEJORA (Método LAZY)	35,00%	40,67%	46,33%	40,67%	36,67%	39,87%

a) Resultados del uso de la variante “rezagada“ (“*lazy*”)

Cinco modelos SVM entrenados en VIVIENDA 4TR y probados con datos de R03H						
Resultados usando Método GAMBLER	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	**% medio**
Predicción correcta del SVM	164	154	139	153	161	51,40%
Pred. incorrecta, pero factible (ej. R04P)	31	24	22	25	29	8,73%
Pred. Incorrecta, corregidas como R03H	80	101	109	92	84	31,07%
Pred. Incorrecta, corregidas como R04P	25	21	30	30	26	8,80%
Número TOTAL de muestras	300	300	300	300	300	**Promedio**
% de error descartable	35,00%	40,67%	46,33%	40,67%	36,67%	39,87%
% de precisión (sólo SVM)	54,67%	51,33%	46,33%	51,00%	53,67%	51,40%
% precisión (Método GAMBLER)	81,33%	85,00%	82,67%	81,67%	81,67%	82,47%
MEJORA (Método GAMBLER)	26,67%	33,67%	36,33%	30,67%	28,00%	31,07%

b) Resultados del uso de la variante “aventurera“ (“*gambler*”)

Figura 36. Comparando el desempeño del Filtro de Contexto (*Lazy* VS. *Gambler*).
Fuente: Elaboración propia.

En los datos en la Figura 36 se puede observar que la variante “aventurera” tiene “peor” desempeño en términos porcentuales de la mejora que aporta su uso en los datos de la habitación R03H. No obstante, esta variante permite apreciar en promedio un 8% más de ejemplos en los que el clasificador “sugiere” que ya está en el pasillo. En este caso, se puede interpretar este hecho como que nos acercamos desde la habitación R03H al pasillo R04P. Algo similar ocurre, aunque a la inversa, con los datos de las mediciones del pasillo cuando nos acercamos a la habitación. Se generan “falsas” clasificaciones de localización en la habitación, cuando “oficialmente” estamos en el pasillo. Estos cambios en la predicción tienen mayor frecuencia usando la variante “aventurera” (“*gambler*”) del Filtro de Contexto.

Esta mayor frecuencia de “saltos” entre la predicción de que el agente robótico está en una estancia y la predicción de que está en una estancia vecina puede servir para diseñar un modelo o algoritmo que, usando un umbral convenientemente establecido y un intervalo de predicciones anteriores, detecte automáticamente que el robot se encuentra en una ubicación “de transición”, incluso llegando a diferenciar y generar un nuevo conocimiento semántico asociado a estas zonas de transición. Es decir, retomando el ejemplo que sugeríamos en el capítulo de “Introducción”, en lugar de decir el robot “*estoy en la habitación de las niñas*”, pudiera decir “*…estoy en la habitación, moviéndome hacia el pasillo*” o “*…estoy en la*

habitación de las niñas, pero ya percibo o 'veo' el pasillo", lo que puede aportar más información y una mayor riqueza semántica sobre la localización del agente robótico móvil.

El escenario anterior puede dar lugar a trabajos más específicos o detallados, por ejemplo, usando redes neuronales recurrentes de tipo LSTM (*Long Short Term Memory*) para tratar las secuencias temporales. Además, el grafo topológico-semántico que se utiliza en el método propuesto se puede generar automáticamente durante las sesiones de recogida de datos y ampliar durante sesiones de trabajo. También se pueden generar otros experimentos usando técnicas de *"data-augmentation"* para comparar el desempeño de las variantes del filtro en recorridos reales o simulados. En el Capítulo 5 se mencionan otros posibles trabajos futuros.

4.8. Sobre almacenamiento en la nube, en el dispositivo y Edge Computing

La Figura 37 muestra el espacio necesario para almacenar los datasets y modelos creados, destacando la gran diferencia entre los *datasets* de mediciones y de "*features*" calculadas.

Nº	Archivo	Tamaño en disco	%
1	Registro de 681 mediciones (Sin incluir TimeStamp en el archivo)	12.943 bytes	0,03%
2	Dataset: 4 viviendas = 7.200 registros x 681 mediciones (con Timestamp)	41.877.504 bytes	100,00%
3	Dataset: 4 viviendas = 7.200 registros x 9 *"features"* (con TimeStamp)	757.760 bytes	1,81%
4	Dataset: 4 viviendas = 7.200 registros x 1 Predicción (con TimeStamp)	110.592 bytes	0,26%
5	Modelo de clasificador entrenado (SVM-RBF.joblib)	462.848 bytes	1,11%
6	Código/algoritmo que carga y ejecuta el modelo (en Python)	4.096 bytes	0,01%
7	Dataset: 4 viviendas = 7.200 registros x 6 probabilidades (incl. TimeStamp)	565.248 bytes	1,35%
8	Grafo topológico-semántico (matriz de NxN), N=Nº de habitaciones (N=6)	78 bytes	0,00%

Figura 37. Comparando el espacio de almacenamiento para Edge Computing.
Fuente: Elaboración propia.

En base a estos datos, se puede reflexionar sobre las ventajas de utilizar estas técnicas de Inteligencia Artificial para reducir drásticamente la cantidad de datos, el tráfico en la red y el espacio de almacenaje en un entorno de Edge Computing. A modo ilustrativo, consideremos que necesitamos guardar en la nube un histórico ("*log*") de las ubicaciones en las que un robot ha estado durante una misión de vigilancia o seguridad para poder ser auditado posteriormente. Si se guarda su posición cada 12 segundos se obtendrían 7.200 registros al día. Extrapolando los valores de la Figura 37, podemos calcular que el espacio requerido para registrar 1 año de actividad sería de 0,02Gb guardando el *dataset* de 9 *features* y el clasificador entrenado (filas 3 y 5 de la Figura 37, respectivamente). En cambio, si se guarda el *dataset* de todas las mediciones (fila 2 de la Figura 37) se necesitan 1,17Gb por año. En el hipotético caso de tratarse de una fábrica o centro logístico con una flota de 100 robots y un requisito legal de almacenar durante tres años los datos históricos, la diferencia sería de 76,22Gb versus 4.212,16Gb. Este ejemplo ilustra una significativa reducción del 98%, atribuible al uso de las técnicas y modelos de Inteligencia Artificial que hemos desarrollado.

Capítulo 5. Conclusiones y trabajo futuro

5.1. Conclusiones generales

A continuación, se presentan las conclusiones y un resumen de los resultados destacados. En primer lugar, el sistema propuesto "*Yielding Associations Network*" aporta una mejora efectiva de la localización semántica de un agente robótico móvil, usando *"features"* estadísticas de las mediciones de un sensor láser y un grafo topológico-semántico. El Filtro de Contexto propuesto permite aplicar el conocimiento del entorno en la tarea de localización.

Con el presente trabajo se ha demostrado que es posible realizar prototipos funcionales de sistemas de localización semántica de bajo coste y alto rendimiento. Permitiendo una comparativa exhaustiva y detallada de varios modelos, variantes y configuraciones (parámetros, hiperparámetros, condiciones de contexto y training, etc.).

Se ha evaluado experimentalmente el impacto de la elección y uso de distintos tipos de variables de entrada, incluyendo atributos calculados (*"engineered features"*) y atributos físicos (distancias), demostrando el mejor desempeño de las "*features*" estadísticas en los modelos estudiados para tareas de aprendizaje automático con robótica móvil. El uso de entornos reales ha permitido demostrar el valor de aplicar íntegramente el ciclo de vida de la gestión de datos en un proyecto basado en Inteligencia Artificial.

El trabajo ha permitido apreciar el impacto de las decisiones, diseños y alternativas planteadas en el almacenamiento de los datos, así como en la efectividad de los modelos estudiados, demostrando el valor de diversas técnicas de Inteligencia Artificial aplicadas para afrontar los retos actuales de Edge Computing y robótica móvil.

5.2. Lecciones aprendidas

A continuación, se destacan aprendizajes resultantes del trabajo desarrollado.

- Importancia de poder realizar el trabajo de forma incremental e iterativa para adaptar las decisiones en base a los resultados que se van obteniendo. Por ejemplo: obtener más datos en siguientes sesiones de medición, proponer modelos innovadores, entre otros.
- Usar varias técnicas de visualización de forma temprana para complementar el entendimiento de los datos. Por ejemplo, la visualización del resultado del escaneo de una habitación, o la visualización de las probabilidades de las clasificaciones que permitieron especular sobre el uso de un operador e inventar el Filtro de Contexto.

- Considerar aplicaciones reales: de negocio y/o de utilidad o beneficio social para acercar el trabajo de investigación a una posible creación de valor posterior. Por ejemplo, considerando desde el inicio del proyecto que se pudiera aplicar el sistema de localización semántica para asistir a personas con discapacidad visual en espacios no conocidos por ellas, o en tareas robotizadas de limpieza, seguridad y vigilancia, etc.

5.3. Líneas futuras de investigación

El trabajo realizado abre varias posibles líneas para realizar futuros trabajos de investigación:

- Oportunidades de implementar o integrar el sistema propuesto con técnicas de aprendizaje por refuerzo ("*Reinforcement Learning*"). Por ejemplo, considerando la detección de novedad o la aplicación de enfoques de exploración y aprendizaje basados en "curiosidad" (Pathak, Agrawal, Efros y Darrell, 2017; Burda et al., 2018).
- Posible uso de un sensor 360º o uso de volumetría 3D. Por ejemplo, moviendo o inclinando el sensor LRF en un eje vertical, enriqueciendo los modelos y datasets creados teniendo en cuenta que se necesita recoger el ángulo de inclinación, de forma conjunta con las otras mediciones.
- Extender al trabajo a viviendas o espacios con topologías más complejas. Por ejemplo, incluyendo todas las estancias de la vivienda, habitaciones muy similares, o estancias con una única dirección de recorrido como ocurre en algunos espacios públicos.
- Uso del reconocimiento de objetos u otros puntos de interés ("*landmarks*"). Por ejemplo, el volumen o imagen de un objeto habitual en una estancia, como puede ser una TV en un salón o una cama en una habitación pueden ayudar a mejorar las predicciones.
- Uso combinado del modelo propuesto con otros sensores. Por ejemplo, un sensor de luz, de temperatura, o un giróscopo para obtener la orientación del agente, etc. pueden ayudar a diferenciar una habitación de otra por tener mayor incidencia de luz natural, temperatura ambiental o tener una orientación diferente (como es el caso de muchos pasillos que son perpendiculares a las habitaciones que éstos conectan).
- Aplicación de los Filtros de Contexto propuestos en modelos *ensembles*, como capas finales o integradoras de los resultados de los modelos individuales.
- Uso de los Filtros de Contexto propuestos para trabajar con secuencias temporales. Por ejemplo, variando la frecuencia de muestreo, usando no sólo la última ubicación conocida sino un intervalo de ubicaciones anteriores para explorar y/o generar nuevo conocimiento semántico, como el reconocimiento de zonas de transición entre espacios.

Referencias bibliográficas

Advanced Factories (2019). Expo & Congress. Barcelona, 9-11 abril 2019. Recuperado de: https://www.advancedfactories.com/

Atanasov, N., Zhu, M., Daniilidis, K., y Pappas, G. J. (2016). Localization from semantic observations via the matrix permanent. *The International Journal of Robotics Research*, *35*(1-3), 73-99.

Bello, Y. (2019). BIOBOT: A Fall Detection System (FDS) using Artificial Intelligence. [Trabajo de prácticas. Máster en Inteligencia Artificial, UNIR]. Recuperado de: http://www.spaceminds.com/wp/biobot-a-fall-detection-system-fds-using-artificial-intelligence/

Berntorp, K., Hoang, T., Quirynen, R. y Di Cairano, S. (2018, August). Control Architecture Design for Autonomous Vehicles. In 2018 IEEE Conference on Control Technology and Applications (CCTA) (pp. 404-411). IEEE.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T. y Efros, A. A. (2018). Large-scale study of curiosity-driven learning. arXiv preprint arXiv:1808.04355.

Cortes, C., y Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.

Förster, A., Graves, A. y Schmidhuber, J. (2007). RNN-based Learning of Compact Maps for Efficient Robot Localization. In ESANN (pp. 537-542).

Fu, J., Atanasov, N., Topcu, U. y Pappas, G. J. (2016, May). Optimal temporal logic planning in probabilistic semantic maps. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on* (pp. 3690-3697). IEEE.

Global Robot Expo (2019). World Congress & Exhibition. Recuperado de: https://www.globalrobotexpo.com

Gupta, R. y Gupta, R. (2016, March). ABC of Internet of Things: Advancements, benefits, challenges, enablers and facilities of IoT. In Colossal Data Analysis and Networking (CDAN), Symposium on (pp. 1-5). IEEE.

H. Kawata (2006). Communication Protocol Specification for SCIP2.0 Standard. Drawing number: C-42-03320B.

Huang, L., Feng, X., Feng, A., Huang, Y. y Qian, L. P. (2018). Distributed Deep Learning-based Offloading for Mobile Edge Computing Networks. Mobile Networks and Applications, 1-8.

Johnston, S. J. y Cox, S. J. (2017). The Raspberry Pi: A Technology Disrupter, and the Enabler of Dreams.

Kotov, K. Y., Maltsev, A. S. y Sobolev, M. A. (2013). Recurrent neural network and extended Kalman filter in SLAM problem. IFAC Proceedings Volumes, 46(20), 23-26.

Kumar, A. C., Bhandarkar, S. M. y Mukta, P. (2018, June). Depthnet: A recurrent neural network architecture for monocular depth prediction. In 1st International Workshop on Deep Learning for Visual SLAM. (CVPR) (Vol. 2, p. 2).

Malhotra, P., Vig, L., Shroff, G. y Agarwal, P. (2015, April). Long short-term memory networks for anomaly detection in time series. In Proceedings (p. 89). Presses universitaires de Louvain.

Olaniyan, R., Fadahunsi, O., Maheswaran, M. y Zhani, M. F. (2018). Opportunistic Edge Computing: Concepts, Opportunities and Research Challenges. arXiv preprint arXiv:1806.04311.

Ordóñez, F. J. y Roggen, D. (2016). Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. Sensors, 16(1), 115.

O'Shea, T. J., Clancy, T. C. y McGwier, R. W. (2016). Recurrent neural radio anomaly detection. arXiv preprint arXiv:1611.00301.

Park, D., Kim, S., An, Y. y Jung, J. Y. (2018). Lired: A light-weight real-time fault detection system for edge computing using LSTM recurrent neural networks. *Sensors*, *18*(7), 2110.

Pathak, D., Agrawal, P., Efros, A. A. y Darrell, T. (2017, May). Curiosity-driven exploration by self-supervised prediction. In International Conference on Machine Learning (ICML) (Vol. 2017).

Piardi, L., Lima, J. y Costa, P. (2018). Development of a Ground Truth Localization System for Wheeled Mobile Robots in Indoor Environments based on Laser Range-finder for Low-cost Systems.

RobotShop (2019). Hokuyo URG-04LX-UG01 Scanning Laser Rangefinder (EU). RobotShop Inc. Recuperado de: https://www.robotshop.com/eu/en/hokuyo-urg-04lx-ug01-scanning-laser-rangefinder-eu.html

Smolyakov, V. (22 de agosto de 2017). Ensemble Learning to Improve Machine Learning Results [Mensaje en un blog]. Stats and Bots. Recuperado de https://blog.statsbot.co/ensemble-learning-d1dcd548e936

Sünderhauf, N. y Protzel, P. (2011, September). Brief-gist-closing the loop by simple means. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*(pp. 1234-1241). IEEE.

Wang, L., Zhao, L., Huo, G., Li, R., Hou, Z., Luo, P., ... y Yang, C. (2018). Visual Semantic Navigation Based on Deep Learning for Indoor Mobile Robots. *Complexity*, *2018*.

Anexos

Anexo 1: Artículo científico

Anexo 2: Repositorio de código fuente

Anexo 3: Resumen de datos de los experimentos

Anexo 4: Guía-Checklist metodológica

Un sistema para mejorar la localización semántica

Yan Bello Méndez

Universidad Internacional de la Rioja, Logroño (España)

27 de febrero de 2019

Resumen

En el ámbito de la *Internet of Things* (IoT), el Edge Computing aporta un enfoque para la computación localizada que permite optimizar el almacenamiento de los datos y agilizar la toma de decisiones. Este trabajo tiene como objetivo estudiar el desempeño de varios algoritmos de inteligencia artificial usados en aplicaciones de localización automática de un robot móvil. La metodología se ha basado en un prototipo experimental, para recoger datos de sensores acoplados a una Raspberry Pi 3B+. Con estos datos se han desarrollado y probado varios modelos de aprendizaje automático usando Google Colab. El funcionamiento de estos modelos se ha validado desplegándolos en el prototipo desarrollado. Los resultados incluyen una comparativa exhaustiva de los algoritmos y una propuesta novedosa de un sistema con filtros de contexto que permiten mejorar la localización semántica. Las conclusiones destacan el gran valor de la Inteligencia Artificial para robótica móvil y el Edge Computing.

Palabras Clave

Edge Computing, Localización semántica, Redes neuronales artificiales, Robótica móvil, Support Vector Machines (SVM)

I. Introducción

Con el auge de los dispositivos conectados y la Internet de las Cosas (IoT, *Internet of Things*), el volumen de datos generados y almacenados en servidores en la nube crece a un ritmo imparable. Esta situación presenta retos técnicos y de gestión, tanto para aplicaciones industriales – en la llamada Industria 4.0 o la Industrial IoT (IioT) –, como para soluciones corporativas, así como para aplicaciones de servicios o productos de uso doméstico. Una de las tecnologías que pretende afrontar este reto es la llamada computación en el borde (en inglés: Edge Computing) que aporta un enfoque para el procesamiento local de los datos, de forma más cercana o próxima a los dispositivos que generan y capturan dichos datos. En este trabajo exploramos el uso de la Inteligencia Artificial para optimizar la gestión de los datos, agilizando la toma de decisiones y permitiendo reducir el volumen de datos enviados a la nube. Planteamos una propuesta novedosa para mejorar la localización semántica de un agente robótico móvil.

II. Estado del Arte

A. Retos y oportunidades de la Internet de las Cosas

Desde hace años, tanto la comunidad científica como profesionales de la industria y los servicios mantienen un foco de atención en temas relacionados con el impacto, oportunidades y retos que presenta el auge de los dispositivos conectados a Internet. La experiencia profesional del autor en este ámbito no hace más que confirmar el interés y vigencia de estos temas. Por ejemplo, durante su participación en dos de las ferias/congresos más destacados sobre industria y manufactura (*Advanced Factories*, Barcelona 2018) y robótica (Robot EXPO, Madrid 2018), el autor pudo comprobar en primera persona que varios de los temas tratados con mayor recurrencia y protagonismo fueron precisamente relativos al impacto de la IoT en la robótica, a la gestión de datos (en la nube, en los dispositivos), a las tendencias de cómputo en el borde (Edge Computing), entre otras. La previsión actual de agenda de contenidos de estos eventos para 2019 es similar.

Los retos y oportunidades de la IoT han sido tratados desde distintas perspectivas y disciplinas, incluyendo la gestión de los datos. En el Simposio sobre "*Colossal Data Analysis and Networking*" de 2016 [1] destacaban la importancia de los avances alcanzados y los retos pendientes. Actualmente, varios de estos retos motivan áreas de investigación activa e interés de aplicación en la industria. En este contexto, el uso de técnicas de Inteligencia Artificial para dotar a agentes robóticos de mayor autonomía en la toma de decisiones y generar comportamientos más adaptativos es uno de los aspectos clave del presente trabajo.

B. Recursos de investigación "low-cost" y altas prestaciones

En los últimos años algunos recursos como las Raspberry Pi, han pasado de ser una herramienta de apoyo a la educación en tecnologías de la información, para convertirse en elemento central de una potente plataforma de experimentación [2].

De forma paralela, otros dispositivos y sensores relacionados se han popularizado por el aumento en la demanda de aplicaciones de IoT, un hecho que se ha acelerado con la bajada de costes de muchos de estos recursos. Por ejemplo, el mismo sensor utilizado en el presente trabajo, el Hokuyo URG-04LX, con un coste de mercado en 2018-2019 de alrededor de 1.000 euros [3] ha sido objeto de estudio detallado y propuesto como elemento sensorial central para un sistema low-cost de localización de referencia ("*Ground Truth*") para robots móviles con ruedas [4]. Los autores de [4] observan que "la tecnología de los escáneres basados en laser ha aportado la habilidad de desarrollar aplicaciones para la medición sin contacto, precisa y rápida en un amplio rango de aplicaciones". En dicho trabajo [4], se propone el uso del sensor LRF Hokuyo para detectar formas geométricas circulares en dos situaciones: en primer lugar, como un objeto estático – que pudiera servir como punto de referencia, y en segundo lugar como un objeto móvil – que pudiera ser el propio robot percibido por un LRF desde una posición fija en una estancia o espacio.

C. Sobre SLAM y localización semántica de robots móviles

El problema general de localización y mapeo simultáneo (SLAM, *Simultaneous Localization And Mapping*) es de crucial importancia para numerosas aplicaciones de robótica móvil y especialmente relevante, si se desea dotar de autonomía a un agente robótico móvil para que pueda navegar en el entorno.

En [5] se plantea una propuesta que consiste en realizar una

localización eficiente de un robot usando redes neuronales recurrentes (RNN) de tipo LSTM (*Long Short Term Memory*). En dicho trabajo, los autores utilizan una red neuronal, entrenada con datos de sensores laser de distancia 2D, para aprender – de forma supervisada – un conjunto de etiquetas topológicas definidas como significativas en un entorno doméstico que simula el espacio en el que se encuentra el robot. Cabe destacar que el método propuesto en [5], no requiere la elaboración de un mapa explícito del entorno ni el cálculo de las coordenadas euclidianas.

Cabe destacar que el uso de Redes Neuronales Recurrentes (RNN) de tipo LSTM también ha encontrado gran acogida en el procesamiento de otros tipos de datos de sensores, no necesariamente acoplados a agentes robóticos, sino también para estudiar y reconocer la actividad humana [6] como sistemas de detección de caídas (*Fall Detection Systems*) [7]. Esto se debe a la buena capacidad de las redes RNN/LSTM para aprender y modelar el comportamiento de fenómenos descritos como series temporales [8] y [9].

Las redes RNN también han sido utilizadas para reducir el error sistemático en modelos basados en el EKF (*Extended Kalman Filter*) [10], donde los autores consideran las tareas de SLAM en 2D en las que un robot hace el seguimiento de un punto objetivo (*target*) y reconstruye la trayectoria real usando datos de sensores de distancia. También es interesante notar que los autores usaron datos de simulación con un robot móvil e-puck, concebido originalmente para uso educacional.

D. Localización semántica avanzada con Deep Learning

También existe un amplio número de trabajos en los que la localización incorpora imágenes, videos y/o información de sensores adicionales a los *Laser Range Finders* de 2D, como por ejemplo cámaras [11], sensores 3D, entre otros. Por ejemplo, en [12] se propone el sistema DepthNet con una arquitectura de RNN (*Recurrent Neural Networks*) para plantear un modelo de Deep Learning que combina redes convolucionales y LSTM (ConvLSTM) y técnicas de SFM (*Structure From Motion*) para realizar predicciones sobre un mapa de profundidad de una escena en base a secuencias de video monocular. Como indican los autores de dicho trabajo, "aprender a predecir la profundidad, incluso aunque sólo sea de manera aproximada, proporciona una oportunidad para inyectar información valiosa en los procedimientos de reconstrucción 3D, estimación e inferencia de la posición tridimensional en SLAM" [12].

Por otra parte, el trabajo [13] propone sistema de reconocimiento de lugares – "basado en apariencias" que incorpora BRIEF-Gist, una mejora a los descriptores de tipo BRIEF (*Binary Robust Independent Elementary Features*) para poder aplicar a tareas de reconocimiento y localización el uso de mapas semánticos con imágenes. Como indican los autores de [13], los sistemas modernos de SLAM de estas características suelen estar divididos en dos partes: *front-end* y *back-end*. Donde el front-end construye un conjunto de restricciones del entorno y puntos de interés ("*landmarks*") y el *back-end* contiene un optimizador que construye y mantiene un mapa del entorno.

III. Objetivos y Metodología

A. Objetivos generales

Con este trabajo se pretende aportar un modelo que sirva de ejemplo para el desarrollo ágil de proyectos que incorporan técnicas de inteligencia artificial en un entorno de Edge Computing con agentes de robótica móvil. Por tanto, se desea alcanzar estos objetivos:

1. Desarrollar una guía-Checklist en base a la propuesta de un sistema-integrado para la implementación ágil de técnicas de inteligencia artificial en un entorno de computación en el borde (Edge Computing) aplicado a un agente de robótica móvil.
2. Desarrollar un prototipo como prueba de concepto de un sistema de recogida de datos y localización semántica de un agente robótico móvil usando técnicas de Inteligencia Artificial, basado en una *Single Board Computer* (SBC) Raspberry Pi y un sensor Laser Range-Finder (LRF) Hokuyo URG-04LX.
3. Realizar experimentos que permitan explorar el uso de varias técnicas de Inteligencia Artificial, usando el prototipo-prueba de concepto y la metodología propuesta.
4. Analizar y comparar el desempeño de las técnicas de Inteligencia Artificial, implementadas en el prototipo de sistema de localización del agente robótico móvil.

B. Objetivos específicos y detallados

Para alcanzar los objetivos se han establecido estas metas específicas:

- Guía-resumen ("Checklist") de implementación de técnicas IA:
  - Realizar una descripción de cuestiones clave.
  - Elaborar preguntas de reflexión a plantearse en cada fase.
- Desarrollar el prototipo de un sistema de localización semántica para un agente robótico móvil:
  - Realizar la definición conceptual-esquemática del sistema.
  - Señalar partes con implementación: borde/Edge o nube.
  - Visualizar el despliegue y explotación de los algoritmos IA.
- Realizar experimentos:
  - Configurar y programar la recogida de datos.
  - Realizar la recogida de datos (Edge).
  - Subir los datos a la nube (Cloud)
  - Crear y entrenar modelos usando algoritmos de Inteligencia Artificial (SVM, KNN, ANN, etc.) y los datos recogidos.
  - Explorar alternativas de uso de los datos y de los algoritmos en la toma de decisiones del agente robótico móvil.
  - Desplegar los modelos entrenados en el agente robótico móvil y comprobar su funcionamiento en "tiempo real".
  - Validar el prototipo de sistema de localización.
- Analizar y comparar el desempeño de los modelos:
  - Obtener y comparar métricas de desempeño de los modelos.
  - Identificar ventajas/desventajas y otras consideraciones en la implementación de tareas de aprendizaje automático.
  - Explorar formas de mejorar el desempeño de los modelos.
  - Elaborar estimaciones para comparar el almacenaje en la nube y en borde (ej. distancias VS. datos calculados).
  - Obtener conclusiones específicas en base a los resultados.

C. Metodología

La metodología utilizada se ha basado en un enfoque de trabajo ágil e iterativo. De esta forma se ha buscado poder desarrollar el proyecto de forma incremental y completarlo de manera satisfactoria, considerando las características singulares de un alcance híbrido (incluyendo métodos, hardware, software, experimentos, gestión de datos, etc.) y los plazos de entrega del trabajo. Aunque las fases de la metodología pudieran considerarse de forma secuencial, en la práctica se han desarrollado de forma iterativa y solapada. Estos solapes han permitido la retroalimentación entre varias actividades que a su vez han facilitado la adaptación continua del trabajo a los retos encontrados y a las soluciones, observaciones y conclusiones preliminares que se iban obteniendo. A continuación, se describen las fases del trabajo.

Fase 1. Desarrollo metodológico

En esta fase se han abordado las consideraciones metodológicas para poder abordar un proyecto de estas características. En una primera parte, se realizó un planteamiento inicial consistente en grandes líneas en las mismas fases aquí descritas. Si bien en una primera instancia no estaba claro o decidido cuántos experimentos se iban a realizar, durante el desarrollo del proyecto se consideró necesario extender-

ampliar los experimentos y mediciones realizadas para incluir otras viviendas adicionales a las inicialmente previstas y poder obtener más datos con los que trabajar.

Fase 2. Revisión bibliográfica y documental

La fase de investigación bibliográfica y revisión documental ha tenido lugar durante prácticamente toda la duración del trabajo. Dada la relevancia y multiplicidad de perspectivas del problema tratado (robótica móvil, Edge Computing, localización semántica, aprendizaje automático supervisado, etc.), se ha estudiado una amplia diversidad de trabajos, incluyendo publicaciones científicas, libros, información divulgativa sobre tendencias, blogs y foros afines, así como webs y material publicitario de algunos productos innovadores relacionados con los objetivos de investigación del proyecto.

Fase 3. Diseño de experimentos

En la fase del diseño experimental, se han tenido en cuenta las características propias y las restricciones más relevantes del dominio del problema para plantear las preguntas, hipótesis y asunciones relevantes que se deben explorar, validar o rechazar. Tratándose de la localización semántica de un agente robótico móvil (Edge Computing)

Training & Cross-testing de los modelos		Probado con datos de las VIVIENDAS				
		1SY	2CA	3ME	4TR	TODAS
Entrenado con datos de las VIVIENDAS	1SY	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	
	2CA	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	
	3ME	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	
	4TR	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	1, 2, 3, 5	
	TODAS					1, 2, 3, 4

MODELOS INCLUIDOS:
1) SVM: Support Vector Machines (LinearSVC)
2) SVM-RBF: Radial Basis Function
3) KNN: K-Nearest Neighbors
4) ANN-MLP: Artificial Neural Networks - Multilayer Perceptron
5) *Yielding Associations Network* (Propuesta de este TFM): usando SVM (kernel=linear) con el "Filtro de Contexto"

Fig. 1. Enfoque de entrenamiento y pruebas cruzadas de los modelos.

mediante el uso exclusivo de datos de sensores, se han planteado cuestiones e hipótesis de trabajo relativas a la selección, recogida y etiquetado de los datos, se han considerado y gestionado situaciones que podían haber generado sesgos, incorporando en el diseño y ejecución de los experimentos variables y factores para garantizar una diversidad adecuada de los datos.

Fase 4. Preparación y configuración del sistema

Para realizar el trabajo se ha preparado y configurado una prueba de concepto ("*proof-of-concept*") de un sistema hardware que ha sido utilizado como prototipo funcional del subsistema de recogida de las mediciones de distancia, emulando lo que pudiera ser un dispositivo para tal fin, usando una Raspberry Pi 3B+ y un sensor laser HOKUYO como elementos centrales de este componente del agente robótico.

Fase 5. Programación de algoritmos y modelos

En esta fase se han desarrollado varios programas, funciones, scripts y algoritmos usando técnicas de Inteligencia Artificial y de la ciencia del dato ("*Data Science*") para:

o Capturar datos del sensor laser, proceder a su etiquetado y registrarlos de manera utilizable para los modelos que se desean desarrollar. Estos desarrollos han sido realizados en Python, desplegando una interfaz efectiva a través de componentes ROS *Packages* para el sensor Laser. Estos algoritmos y funciones se programaron directamente en la Raspberry Pi 3B+ y fueron ejecutados sobre el sistema operativo Linux Ubuntu (16.04) instalado en dicha Raspberry Pi.

o Crear, entrenar y validar varios modelos basados en técnicas de Inteligencia Artificial, incluyendo Máquinas de Vector de Soporte (Support Vector Machines, SVM), *K-Nearest Neighbors* y Redes Neuronales Artificiales (Perceptrón Multicapas, MLP). Estos modelos han sido programados y entrenados en Python, usando la plataforma de Google Colab. Su ejecución y validación ha tenido lugar en la nube (Google Colab) y también en el borde (Edge Computing). Ver punto siguiente.

o Desplegar los modelos de Inteligencia Artificial en el borde (Edge Computing). Una vez entrenados en Google Colab, se han desplegado y ejecutado en tiempo real en la Raspberry Pi, realizando la captura de datos de los sensores "en vivo" y clasificando, "prediciendo" en qué estancia, espacio o ubicación se encontraba el agente robótico.

Fase 6. Recogida de datos con el agente robótico móvil

En esta fase se han usado los algoritmos y funciones antes mencionadas, y se ha seguido el Plan de recogida de datos del diseño experimental para realizar varias sesiones de recogida de datos. Estas sesiones de recogida de datos con el agente robótico móvil han tenido lugar en varias viviendas ubicadas en la comunidad de Madrid, por lo que el autor ha desplazado hasta éstas el prototipo "prueba de concepto" y lo ha configurado para que la recogida de datos ocurra con etiquetado simultáneo a la captura de las mediciones. Los datos recogidos de esta forma han sido almacenados localmente en la tarjeta de memoria de la Raspberry Pi. Posteriormente, los datos han sido "subidos" (enviados) a la nube, a un servidor de Google Drive para ser almacenados de forma segura y procesados para limpiarlos y prepararlos para el modelado como parte de la siguiente fase.

Fase 7. Análisis y comparativa de resultados

La fase de análisis y comparativa de los modelos ha sido muy enriquecedora y productiva. En esta fase se han obtenido diversas métricas del desempeño de los modelos implementados, tanto a través de conjuntos de datos de validación y pruebas, como con datos capturados en directo, obtenidos tras el despliegue de los modelos en el prototipo.

Usando los resultados de estas métricas de desempeño de los distintos modelos se han realizado comparaciones diversas. Para maximizar la riqueza de esta parte de los experimentos se ha definido y utilizado un enfoque de entrenamiento y *cross-testing*, según el cual varios de los modelos desarrollados se han entrenado con datos de una vivienda y probado con datos de todas las viviendas. Así mismo, también se han entrenado y probado varios modelos usando los datos de todas las viviendas. Estos modelos "integrados" también han sido probados con datos de todas las viviendas. La Figura 1 resume el enfoque de pruebas cruzadas que ha permitido realizar más de 68 pruebas-experimentos con los modelos entrenados.

Fase 8. Obtención de conclusiones

En base a los resultados de los experimentos, las mediciones y los análisis realizados, se han extraído varias conclusiones destacadas del trabajo, resaltando también varios aprendizajes obtenidos con el proyecto. Además, se han explorado posibles líneas futuras de investigación y/o de aplicación a oportunidades de negocio que pueden dar continuidad al trabajo realizado.

IV. Contribución

A. Sobre el diseño de la solución y de los experimentos

Los experimentos se han diseñado para realizar la localización semántica de un robot móvil usando sólo datos de distancia obtenidos con un sensor laser, considerando los siguientes elementos:

- El movimiento del robot ha sido simulado con una maleta con ruedas, desde donde se han recogido las mediciones del sensor láser, mientras la maleta era desplazada por dentro de cada vivienda
- Las mediciones se realizaron en seis tipos de habitaciones diferentes (R01S-Salón, R02C-Cocina, R03H-habitación/dormitorio, R04P-pasillo, RP5D-despacho y R06B-baño).

- Se han recogido datos de mediciones de los seis tipos de estancias anteriores en cuatro viviendas diferentes (“1SY”, “2RJ”, “3ME” y “4TR”) de Madrid, todas de uso familiar.

Para poder dar respuesta al reto de localización del robot móvil usando los datos de un sensor láser, hay que destacar el importante papel que juegan los datos en la implementación de algoritmos y técnicas de Inteligencia Artificial. Por tanto, desde el principio se ha trabajado en entender, conocer y decidir las siguientes cuestiones:

- ¿Cuáles, cómo y cuántos datos se pueden capturar con el sensor que vamos a utilizar? ¿En qué formato se obtienen estos datos? ¿Con qué precisión y frecuencia se deben recoger, guardar, etc.?
- ¿Cómo realizar el etiquetado de los datos manera automática?

Para satisfacer la necesidad de etiquetar los datos de forma simultánea a su recogida, se ha programado una aplicación en Python que permite la elección de la estancia en la que se realiza la recogida de datos y automatizar el etiquetado.

B. Desarrollo de modelos y técnicas de Inteligencia Artificial

El desarrollo de los modelos se ha iniciado con la recogida de los datos: la limpieza y preparación de los datasets. Se han considerado varios posibles atributos o *features* como el uso de las mediciones de distancia del sensor láser o características estadísticas (Figura 2).

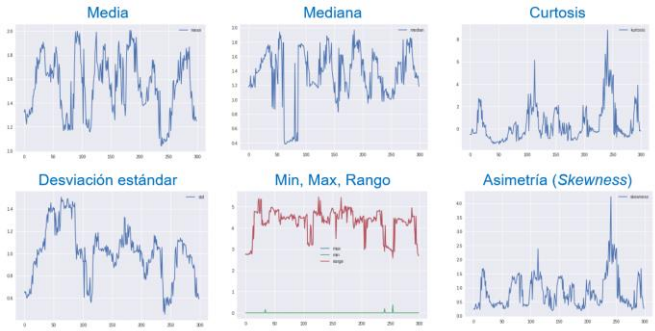


Fig. 2. Evolución de las “features” para la habitación R03H (vivienda 2CA).

Se han entrenado y probado diversos modelos usando el enfoque descrito en la Figura 1. A continuación, se presentan algunos resultados destacados incluyendo la Matriz de Confusión de una Support Vector Machine (SVM) de núcleo radial (Figura 3).

		CLASE PREDICCIÓN						
		R01S	R02C	R03H	R04P	R05D	R06B	TOTAL
CLASE REAL	R01S	284	20	34	37	5	2	382
	R02C	10	237	4	38	30	21	340
	R03H	18	14	301	17	18	8	376
	R04P	21	33	5	267	27	11	364
	R05D	1	52	13	19	247	19	351
	R06B	2	19	7	14	15	290	347
	TOTAL	336	375	364	392	342	351	2160

Fig. 3. Matriz de confusión del modelo “integrado” (SVM-RBF). Modelo Máquina de Vector de Soporte, con kernel Función de Base Radial (RBF), entrenado con datos de todas las viviendas/habitaciones. La precisión (“*accuracy*”) del modelo en datos de prueba: 0,7528 (75,28%)

Se han realizado numerosos experimentos, entrenando varios modelos y arquitecturas de redes neuronales con distintos parámetros e hiperparámetros aplicando técnicas de regularización (ej. *Batch Normalization* y *Dropout*) para minimizar el *overfitting*. El mejor resultado obtenido fue con redes Perceptrón Multicapa alcanzando una precisión (“*accuracy*”) en datos de prueba de 0,590 (59%), usando como datos de entrada las “features” estadísticas normalizadas. Otra red similar, pero usando las mediciones de distancias del láser como entradas sólo alcanzó una precisión del 0,375. (37,5%).

C. Desempeño cuantitativo y comparativa de los modelos

En la Figura 4 se muestra una comparativa de los mejores resultados de más de 50 pruebas cruzadas, realizadas con modelos entrenados con los datos de una vivienda y probados con los datos del resto de viviendas. En estos experimentos se usaron todos los modelos estudiados y sus variantes (SVM: SVC y RBF, KNN, ANN, etc.).

Modelo LinearSVC		Probado con datos de las VIVIENDAS				
		1SY	2CA	3ME	4TR	TODAS
Entrenado con datos de las VIVIENDAS	1SY	0,652	0,547	0,439	0,285	
	2CA	0,532	0,712	0,463	0,308	
	3ME	0,530	0,432	0,586	0,395	
	4TR	0,315	0,292	0,371	0,544	
	TODAS					0,518

Modelo SVM-RBF		Probado con datos de las VIVIENDAS				
		1SY	2CA	3ME	4TR	TODAS
Entrenado con datos de las VIVIENDAS	1SY	0.867	0,502	0,377	0,292	
	2CA	0,438	0,846	0,389	0,262	
	3ME	0,359	0,378	0,848	0,433	
	4TR	0,302	0,322	0,480	0,767	
	TODAS					0,752

Modelo KNN, K=4		Probado con datos de las VIVIENDAS				
		1SY	2CA	3ME	4TR	TODAS
Entrenado con datos de las VIVIENDAS	1SY	0,855	0,492	0,385	0,290	
	2CA	0,466	0,809	0,376	0,252	
	3ME	0,363	0,369	0,867	0,407	
	4TR	0,347	0,305	0,460	0,785	
	TODAS					0,773

Fig. 4. Comparativa del desempeño de los modelos (+50 pruebas cruzadas). Métrica usada: la precisión (“*accuracy*”), calculada como la ratio entre el número de aciertos y el total de instancias.

Los resultados obtenidos permiten destacar varios hechos:

- Los mejores resultados son con: SVM-RBF y KNN, K=4.
- En general, todos los modelos entrenados con datos de una sola vivienda, “sufren” una pérdida considerable de precisión al ser probados “puestos en producción” en los datos de las otras viviendas.
- Se aprecia que los modelos “integrados” tienen un desempeño similar al “peor” de los modelos de cada tipo.
- También son destacables las diferencias entre los resultados de las viviendas 3ME y 4TR, a pesar de ser prácticamente iguales en términos de tamaños y distribución física de sus habitaciones.

Aunque estos resultados pueden considerarse bastante buenos, en el siguiente apartado se presenta una propuesta para mejorar el desempeño de la clasificación usada en la localización semántica.

D. Mejora del desempeño con un grafo topológico semántico

La propuesta novedosa que se plantea se basa en utilizar el conocimiento del entorno del robot extraído de la topología semántica para ajustar y mejorar las predicciones de su ubicación. Teniendo en consideración las restricciones que impone la arquitectura del entorno (en estos experimentos: las viviendas), en la cual se encuentra y por donde realiza su recorrido el robot, podemos corregir algunas de las predicciones erróneas del clasificador utilizado.

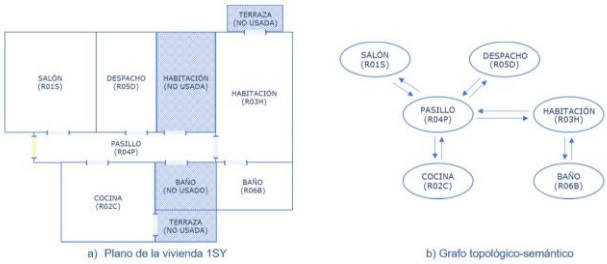


Fig. 5. Representaciones del conocimiento del entorno.

Para realizar esta corrección consideramos que, si el robot está en una estancia determinada, desde dicha ubicación sólo serían posible el movimiento a algunas, pero no a todas las ubicaciones de la vivienda, o sería posible permanecer en la misma estancia. Estas opciones las visualizamos usando un grafo topológico-semántico (Figura 5).

E. Yielding Associations Network: sistema mejorado de localización semántica

El sistema propuesto lo denominamos "*Yielding Association Network*" (Red Generadora de Asociaciones). Como se ilustra en la Figura 6, esta red incluye un primer componente que tiene la función de hacer una "propuesta" de clasificación-asociación, y un segundo componente que funciona como un operador de "Filtro de Contexto".

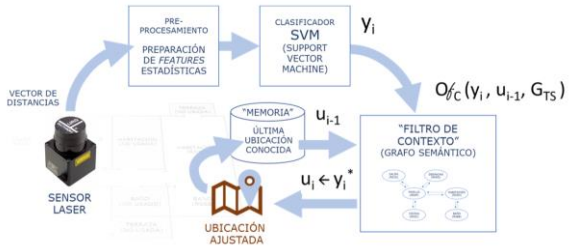


Fig. 6. Sistema de localización semántica: *Yielding Associations Network*. Usando un clasificador SVM y Operador de "Filtro de Contexto".

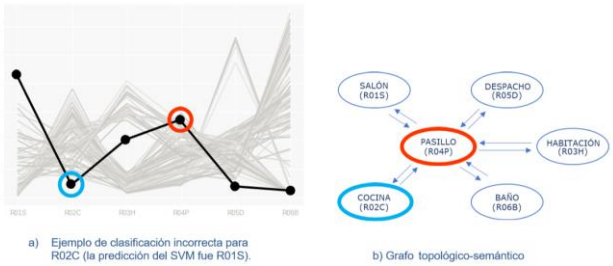
Proponemos dos variantes del Filtro de Contexto. Una variante que denominamos "rezagada" ("*lazy*"), corrige parte de los errores usando la última posición conocida, se define en la Ecuación 1.

$$\text{a) } O_{f_C}(y_i, u_{i-1}, G_{TS}) \rightarrow y_i^* \qquad \text{b) } u_i \leftarrow y_i^*$$

$$y^*_i = y_i \quad si\ y_i = u_{i-1}$$
$$y^*_i = y_i \quad si\ y_i \in G_{TS}(u_{i-1})$$
$$y^*_i = u_{i-1} \quad si\ y_i \notin G_{TS}(u_{i-1})$$

Ecuación 1. Definición formal del Filtro de Contexto Of_C (variante: "*lazy*"). Donde yi es la predicción del clasificador, ui-1 es la ubicación anterior, G_{TS} es el grafo topológico-semántico y $G_{TS}(ui)$ representa el conjunto de nodos conectados directamente con u_i y y_i* es la ubicación ajustada.

La segunda variante, que definimos como "osada" o "aventurera" ("*gambler*"), de modo que asume riesgos dándole una "segunda oportunidad" al clasificador en el caso de una predicción contradictoria. En este caso, se podría seleccionar la siguiente opción de nodo/ubicación "viable" que el clasificador considere más probable.



a) Ejemplo de clasificación incorrecta para R02C (la predicción del SVM fue R01S).

b) Grafo topológico-semántico

Fig. 7. Diferencias en la clasificación dependiendo del filtro (*"Lazy"* VS. *"Gambler"*). Clasificaciones incorrectas, pero mejorables (vivienda 4TR).

En la Figura 7.a se muestra un ejemplo de instancia en la que el sistema propuesto tendría un desempeño diferente dependiendo de la variante usada ("*lazy*" o "*gambler*"). Si el agente robótico móvil estaba (venía) de una ubicación en la cocina (R02C), serían válidas las predicciones de continuar en la cocida o moverse al pasillo R04P.

V. Evaluación de los Resultados

Se han realizado numerosas pruebas para estudiar y evaluar los resultados del uso de las dos variantes del Filtro de Contexto.

A. Evaluación del Filtro de Contexto: variante "Lazy"

Todas las pruebas realizadas confirman que el uso del Filtro de Contexto (variante "*Lazy*") mejora considerablemente los resultados del clasificador y, por tanto, la localización semántica del robot. En la Figura 8 se muestran los resultados en las 4 viviendas y habitaciones.

VIVIENDA 1SY							
SVM (kernel = linear)	**R01S**	**R02C**	**R03H**	**R04P**	**R05D**	**R06B**	**TODAS**
Predicción correcta del SVM	235	103	269	208	232	268	1315
Pred. incorrecta, pero factible (ej. R04P)	9	76	6	68	17	0	176
Pred. incorrecta, pero imposible	56	121	25	24	51	32	309
Número TOTAL de muestras	300	300	300	300	300	300	1800
% de error descartable	18,67%	40,33%	8,33%	8,00%	17,00%	10,67%	17,17%
% de precisión (sólo SVM)	78,33%	34,33%	89,67%	69,33%	77,33%	89,33%	73,06%
% precisión (SVM+Grafo semántico)	97,00%	74,67%	98,00%	77,33%	94,33%	100,00%	90,22%

VIVIENDA 2CA							
SVM(kernel=linear)	**R01S**	**R02C**	**R03H**	**R04P**	**R05D**	**R06B**	**TODAS**
Predicción correcta del SVM	240	171	177	238	265	270	1361
Pred. incorrecta, pero factible (ej. R04P)	7	34	20	62	9	1	133
Pred. incorrecta, pero imposible	53	95	103	0	26	29	306
Número TOTAL de muestras	300	300	300	300	300	300	1800
% de error descartable	17,67%	31,67%	34,33%	0,00%	8,67%	9,67%	17,00%
% de precisión (sólo SVM)	80,00%	57,00%	59,00%	79,33%	88,33%	90,00%	75,61%
% precisión (SVM+Grafo semántico)	97,67%	88,67%	93,33%	79,33%	97,00%	99,67%	92,61%

VIVIENDA 3ME							
SVM(kernel=linear)	**R01S**	**R02C**	**R03H**	**R04P**	**R05D**	**R06B**	**TODAS**
Predicción correcta del SVM	210	131	237	183	221	248	1230
Pred. incorrecta, pero factible (ej. R04P)	33	97	32	117	37	20	336
Pred. incorrecta, pero imposible	57	72	31	0	42	32	234
Número TOTAL de muestras	300	300	300	300	300	300	1800
% de error descartable	19,00%	24,00%	10,33%	0,00%	14,00%	10,67%	13,00%
% de precisión (sólo SVM)	70,00%	43,67%	79,00%	61,00%	73,67%	82,67%	68,33%
% precisión (SVM+Grafo semántico)	89,00%	67,67%	89,33%	61,00%	87,67%	93,33%	81,33%

VIVIENDA 4TR							
SVM(kernel=linear)	**R01S**	**R02C**	**R03H**	**R04P**	**R05D**	**R06B**	**TODAS**
Predicción correcta del SVM	221	158	164	102	102	209	956
Pred. incorrecta, pero factible (ej. R04P)	24	43	31	198	48	17	361
Pred. incorrecta, pero imposible	55	99	105	0	150	74	483
Número TOTAL de muestras	300	300	300	300	300	300	1800
% de error descartable	18,33%	33,00%	35,00%	0,00%	50,00%	24,67%	26,83%
% de precisión (sólo SVM)	73,67%	52,67%	54,67%	34,00%	34,00%	69,67%	53,11%
% precisión (SVM+Grafo semántico)	92,00%	85,67%	89,67%	34,00%	84,00%	94,33%	79,94%

Fig. 8. Mejora del clasificador SVM con el grafo topológico-semántico. Resultados de aplicación en las viviendas 1SY, 2CA, 3ME y 4TR.

B. Evaluación del Filtro de Contexto: variante "Gambler"

La Figura 9 muestra una comparación del desempeño de ambas variantes ("rezagado" VS. "aventurero") del Filtro de Contexto en la habitación R03H de la vivienda 4TR.

Cinco modelos SVM entrenados en VIVIENDA 4TR y probados con datos de R03H						
Resultados usando Método LAZY	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	**% medio**
Predicción correcta del SVM	164	154	139	153	161	51,40%
Pred. incorrecta, pero factible (ej. R04P)	31	24	22	25	29	8,73%
Pred. incorrecta, corregidas como R03H	105	122	139	122	110	39,87%
Número TOTAL de muestras	300	300	300	300	300	**Promedio**
% de error descartable	35,00%	40,67%	46,33%	40,67%	36,67%	39,87%
% de precisión (sólo SVM)	54,67%	51,33%	46,33%	51,00%	53,67%	51,40%
% precisión (Método LAZY)	89,67%	92,00%	92,67%	91,67%	90,33%	91,27%
MEJORA (Método LAZY)	35,00%	40,67%	46,33%	40,67%	36,67%	39,87%

Cinco modelos SVM entrenados en VIVIENDA 4TR y probados con datos de R03H						
Resultados usando Método GAMBLER	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	**% medio**
Predicción correcta del SVM	164	154	139	153	161	51,40%
Pred. incorrecta, pero factible (ej. R04P)	31	24	22	25	29	8,73%
Pred. Incorrecta, corregidas como R03H	80	101	109	92	84	31,07%
Pred. Incorrecta, corregidas como R04P	25	21	30	30	26	8,80%
Número TOTAL de muestras	300	300	300	300	300	**Promedio**
% de error descartable	35,00%	40,67%	46,33%	40,67%	36,67%	39,87%
% de precisión (sólo SVM)	54,67%	51,33%	46,33%	51,00%	53,67%	51,40%
% precisión (Método GAMBLER)	81,33%	85,00%	82,67%	81,67%	81,67%	82,47%
MEJORA (Método GAMBLER)	26,67%	33,67%	36,33%	30,67%	28,00%	31,07%

Fig. 9. Comparando el desempeño del Filtro de Contexto (variantes: *Lazy* VS. *Gambler*). Evaluados en la habitación R03H de la vivienda 4TR.

VI. Discusión y Análisis de Resultados

En los datos en la Figura 9 se puede observar que la variante "aventurera" tiene "peor" desempeño en términos porcentuales de la mejora que aporta su uso en los datos de la habitación R03H. No obstante, esta variante permite apreciar en promedio un 8% más de ejemplos en los que el clasificador "sugiere" que ya está en el pasillo. En este caso, se puede interpretar este hecho como que nos acercamos desde la habitación R03H al pasillo R04P. Algo similar ocurre, aunque a la inversa, con los datos de las mediciones del pasillo cuando nos acercamos a la habitación. Se generan "falsas" clasificaciones de

localización en la habitación, cuando "oficialmente" estamos en el pasillo. Estos cambios en la predicción tienen mayor frecuencia usando la variante "aventurera" ("*gambler*") del Filtro de Contexto.

Los resultados obtenidos también permiten reflexionar sobre el almacenamiento en la nube, en el dispositivo y el Edge Computing. La Figura 10 muestra la gran diferencia de espacio necesario para almacenar datasets de mediciones y de "features" estadísticas.

Nº	Archivo	Tamaño en disco	%
1	Registro de 681 mediciones (Sin incluir TimeStamp en el archivo)	12.943 bytes	0,03%
2	Dataset: 4 viviendas = 7.200 registros x 681 mediciones (con Timestamp)	41.877.504 bytes	100,00%
3	Dataset: 4 viviendas = 7.200 registros x 9 *"features"* (con TimeStamp)	757.760 bytes	1,81%
4	Dataset: 4 viviendas = 7.200 registros x 1 Predicción (con TimeStamp)	110.592 bytes	0,26%
5	Modelo de clasificador entrenado (SVM-RBF.joblib)	462.848 bytes	1,11%
6	Código/algoritmo que carga y ejecuta el modelo (en Python)	4.096 bytes	0,01%
7	Dataset: 4 viviendas = 7.200 registros x 6 probabilidades (incl. TimeStamp)	565.248 bytes	1,35%
8	Grafo topológico-semántico (matriz de NxN), N=Nº de habitaciones (N=6)	78 bytes	0,00%

Fig. 10. Diferencias en la clasificación dependiendo del filtro (*"Lazy"* VS. *"Gambler"*). Clasificaciones incorrectas, pero mejorables (vivienda 4TR).

A modo ilustrativo, consideremos que necesitamos guardar en la nube un histórico ("log") de las ubicaciones en las que un robot ha estado durante una misión de vigilancia o seguridad para poder ser auditado posteriormente. Si se guarda su posición cada 12 segundos se obtendrían 7.200 registros al día. Extrapolando los valores de la Figura 10, podemos calcular que el espacio requerido para registrar 1 año de actividad sería de 0,02Gb guardando el *dataset* de 9 features y el clasificador entrenado (filas 3 y 5 de la Figura 10, respectivamente). En cambio, si se guarda el *dataset* de todas las mediciones (fila 2 de la Figura 10) se necesitan 1,17Gb por año. En el hipotético caso de tratarse de una fábrica o centro logístico con una flota de 100 robots y un requisito legal de almacenar durante tres años los datos históricos, la diferencia sería de 76,22Gb versus 4.212,16Gb. Lo que constituye una significativa reducción del 98%, atribuible al uso de las técnicas y modelos de Inteligencia Artificial que hemos desarrollado.

VII. Conclusiones

A continuación, se presentan las conclusiones y un resumen de los resultados destacados.

En primer lugar, el sistema propuesto "*Yielding Associations Network*" aporta una mejora efectiva de la localización semántica de un agente robótico móvil, usando "features" estadísticas de las mediciones de un sensor láser y un grafo topológico-semántico. El Filtro de Contexto propuesto permite aplicar el conocimiento del entorno en la tarea de localización.

Con el presente trabajo se ha demostrado que es posible realizar prototipos funcionales de sistemas de localización semántica de bajo coste y alto rendimiento. Permitiendo una comparativa exhaustiva y detallada de varios modelos, variantes y configuraciones (parámetros, hiperparámetros, condiciones de contexto y training, etc.).

Se ha evaluado experimentalmente el impacto de la elección y uso de distintos tipos de variables de entrada, incluyendo atributos calculados ("*engineered features*") y atributos físicos (distancias), demostrando el mejor desempeño de las "features" estadísticas en los modelos estudiados para tareas de aprendizaje automático con robótica móvil. El uso de entornos reales ha permitido demostrar el valor de aplicar íntegramente el ciclo de vida de la gestión de datos en un proyecto basado en Inteligencia Artificial.

El trabajo ha permitido apreciar el impacto de las decisiones, diseños y alternativas planteadas en el almacenamiento de los datos, así como en la efectividad de los modelos estudiados, demostrando el valor de diversas técnicas de Inteligencia Artificial aplicadas para afrontar los retos actuales de Edge Computing y robótica móvil.

Además, este trabajo sugiere varias líneas de investigación futura:

- Oportunidades de implementar o integrar el sistema propuesto con técnicas de aprendizaje por refuerzo. Por ejemplo, usando la detección de novedad o *outliers*, así como el posible uso de métricas de curiosidad/novedad, etc. Ver referencias [14], [15] y [16].
- Ampliación del modelo para el uso de un sensor 360º y volumetría 3D. Por ejemplo, enriqueciendo los datasets con datos del ángulo de inclinación, de forma conjunta con las otras mediciones.
- Extender al trabajo a viviendas con topologías más complejas. Por ejemplo, incluyendo todas las estancias de la vivienda, habitaciones muy similares, estancias con una única dirección, etc.
- Uso del reconocimiento de objetos u otros puntos de interés ("*landmarks*"). Por ejemplo, el volumen de un objeto habitual como puede ser una TV en un salón puede ayudar a mejorar las predicciones.
- Uso combinado del modelo propuesto con otros sensores. Por ejemplo, un sensor de luz, temperatura, giróscopo, etc.
- Uso de los Filtros de Contexto propuestos en modelos *ensembles*, para integrar los resultados de modelos individuales.
- Uso de los Filtros de Contexto para trabajar con secuencias temporales. Por ejemplo, variando la frecuencia de muestreo, etc.

Referencias

[1] R. Gupta, "ABC of Internet of Things: Advancements Benefits Challenges Enablers and Facilities of IoT", Symposium on Colossal Data Analysis and Networking (CDAN), 2016, March 2016.

[2] S. J. Johnston, S. J. Cox, "The raspberry Pi: A technology disrupter and the enabler of dreams", Electronics, vol. 6, no. 3, pp. 51, 2017.

[3] L. Wang, L. Zhao, G. Huo, R. Li, Z. Hou, P. Luo, Z. Sun, K. Wang, C. Yang, "Visual semantic navigation based on deep learning for indoor mobile robots", Complexity, vol. 2018, 2018.

[4] Y. Bello. "BIOBOT: A Fall Detection System (FDS) using Artificial Intelligence". Internet: http://www.spaceminds.com/wp/biobot-a-fall-detection-system-fds-using-artificial-intelligence/, [Jan. 7, 2019].

[5] RobotShop. "Hokuyo URG-04LX-UG01 Scanning Laser Rangefinder". Internet: https://www.robotshop.com/eu/en/hokuyo-urg-04lx-ug01-scanning-laser-rangefinder-eu.html, [Jan. 13, 2019].

[6] A. Förster, A. Graves, J. Schmidhuber, "RNN-based learning of compact maps for efficient robot localization", Proceedings 2007, pp. 537-542, April 25-27, 2007.

[7] F. J. Ordóñez, D. Roggen. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. Sensors, 2016.

[8] T. J. O'shea, T. C. Clancy, R. McGwier, "Recurrent neural radio anomaly detection", 2016.

[9] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, "Long Short Term Memory Networks for Anomaly Detection in Time Series", European Symposium on Artificial Neural Networks, vol. 23, 2015.

[10] K. Y. Kotov, A. S. Maltsev, M. A. Sobolev. "Recurrent neural network and extended Kalman filter in SLAM problem", IFAC Proceedings Volumes, vol 46, 2013.

[11] L. Piardi, J. Lima, P. Costa. "Development of a Ground Truth Localization System for Wheeled Mobile Robots in Indoor Environments based on Laser Range-finder for Low-cost Systems", 2018.

[12] A. CS Kumar, S. M. Bhandarkar, P. Mukta, "Depthnet: A recurrent neural network architecture for monocular depth prediction", 1st International Workshop on Deep Learning for Visual SLAM (CVPR), 2018.

[13] N. Sünderhauf, P. Protzel, "BRIEF-Gist—Closing the loop by simple means", Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., pp. 1234-1241, Sep 2011.

[14] J. Schmidhuber, "Developmental robotics optimal artificial curiosity creativity music and the fine arts", Connect. Sci., vol. 18, no. 2, pp. 173-187, 2006.

[15] D. Pathak, P. Agrawal, A.A. Efros, T. Darrell, "Curiosity-driven exploration by self-supervised prediction", Proc. Int. Conf. Machine Learning, pp. 2778-2787, 2017.

[16] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, A. A. Efros. "Large-scale study of curiosity-driven learning", 2018.

Anexo 2: Repositorio de código fuente

El código fuente desarrollado como parte de este trabajo ha sido publicado en un repositorio GitHub con licencia abierta tipo MIT. Los códigos se han desarrollado por el autor en Python usando el entorno de Google Colab y están organizados en los siguientes apartados.

- 01. General data preparations
- 02. Define and train SVM (Linear & RBF) and KNN models
- 03. Define and train Neural Network models
- 04. Cross testing and analysis of the models
- 05. Yielding Associations Network & Context Filters
- 06. Support functions: Laser_print.py & Room_teller.py

Enlace/dirección para acceder al repositorio: http://www.github.com/spaceminds/

El código fuente incluye una mención reconociendo que ha sido desarrollado como parte del presente TFM: *"Los modelos y el código Python han sido desarrollados por Yan Bello, como parte del Trabajo Final de Máster (TFM) en el Máster en Inteligencia Artificial (UNIR)".*

Anexo 3. Resumen de datos de los experimentos

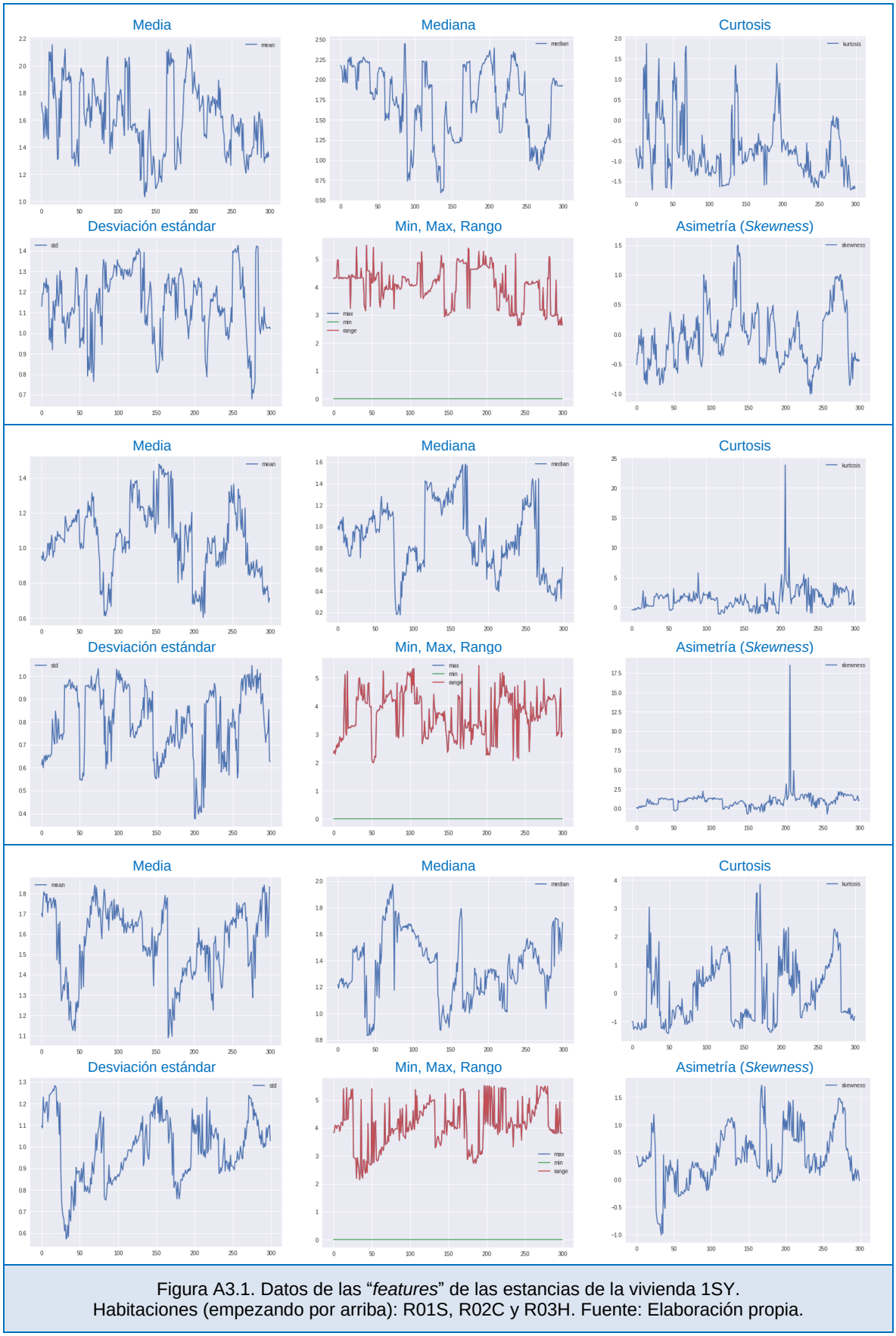


Figura A3.1. Datos de las "*features*" de las estancias de la vivienda 1SY.
Habitaciones (empezando por arriba): R01S, R02C y R03H. Fuente: Elaboración propia.

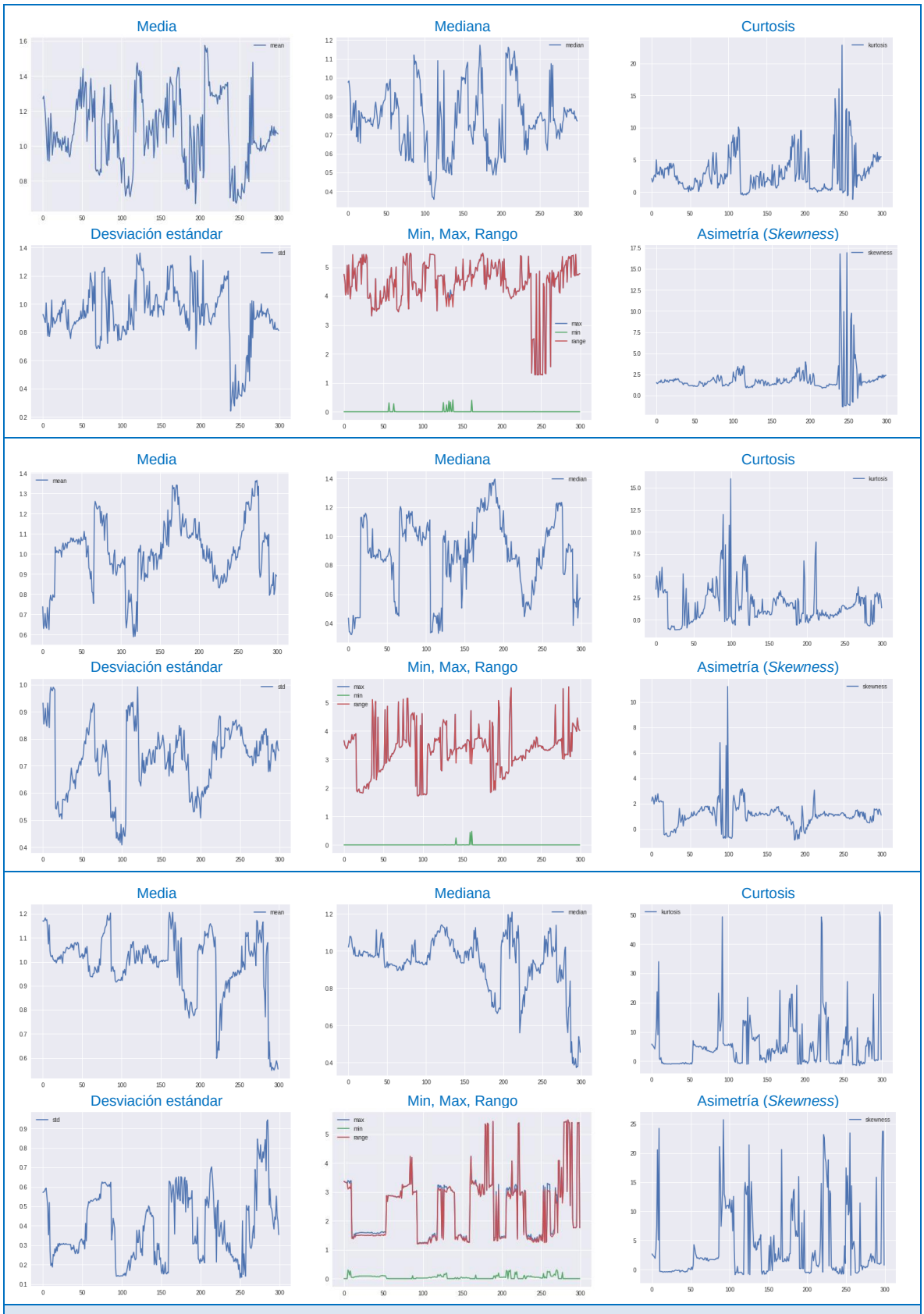


Figura A3.2. Datos de las "*features*" de las estancias de la vivienda 1SY.
Habitaciones (empezando por arriba): R04P, R05D y R06B. Fuente: Elaboración propia.

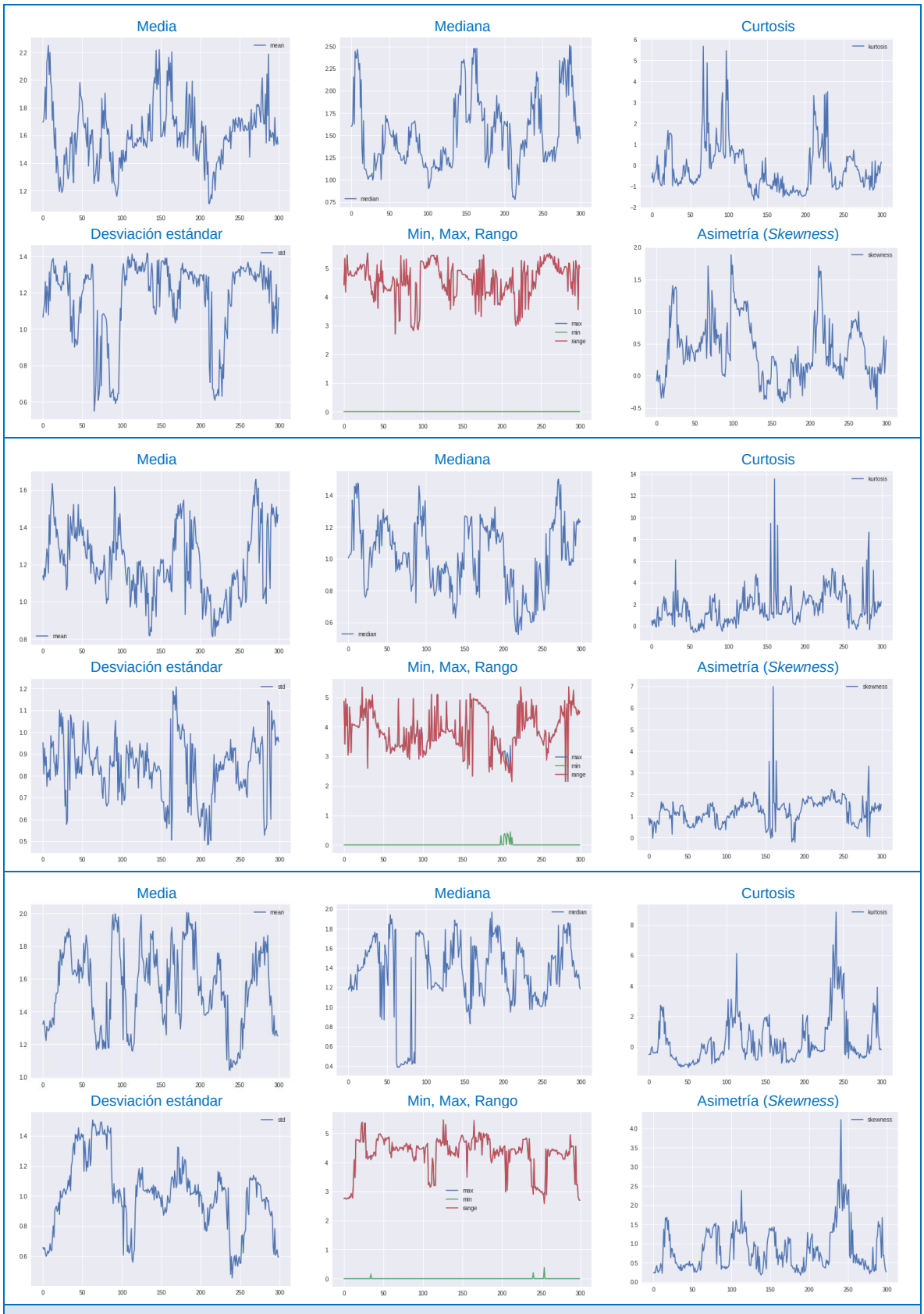


Figura A3.3. Datos de las "*features*" de las estancias de la vivienda 2CA.
Habitaciones (empezando por arriba): R01S, R02C y R03H. Fuente: Elaboración propia.

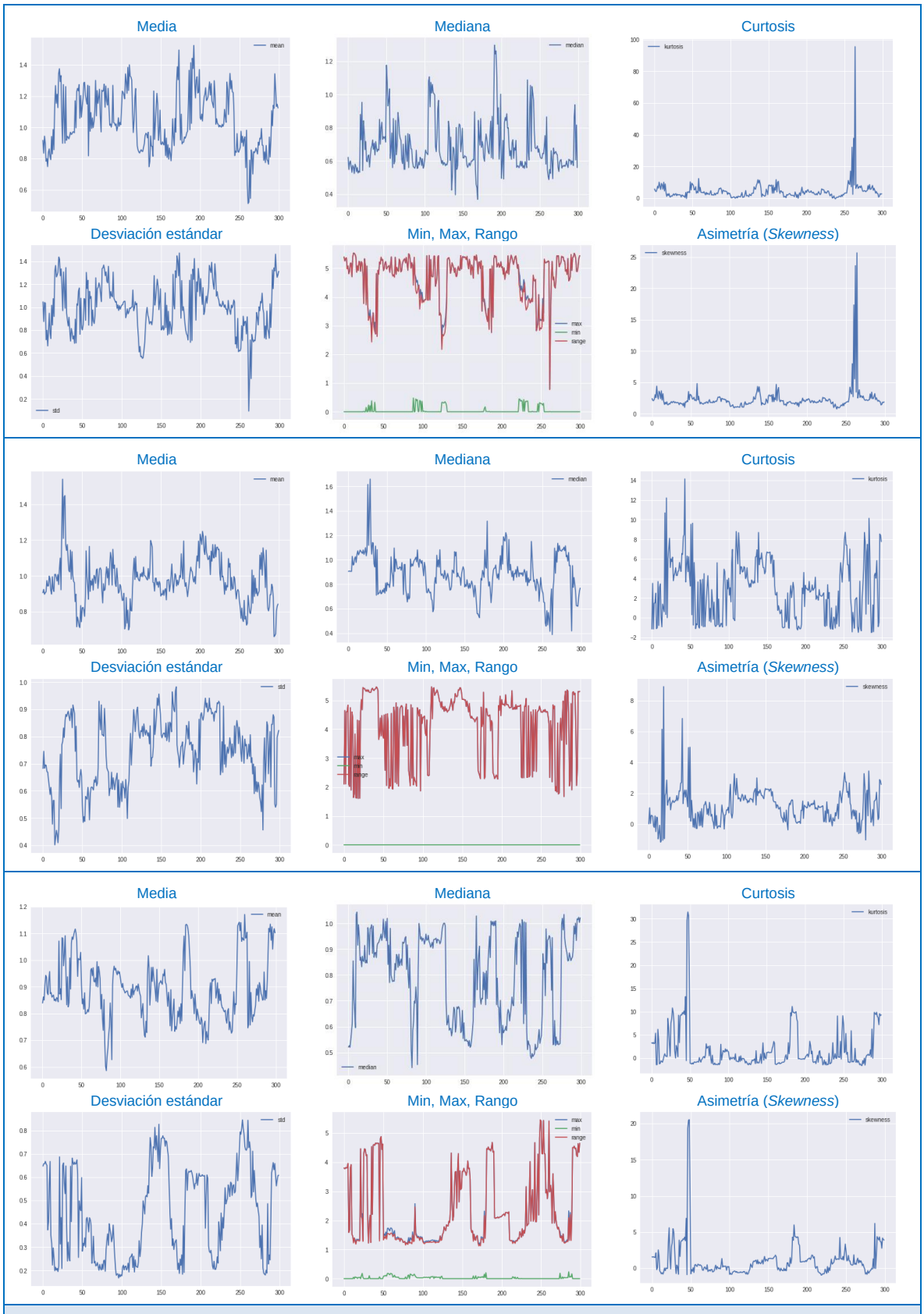


Figura A3.4. Datos de las "*features*" de las estancias de la vivienda 2CA.
Habitaciones (empezando por arriba): R04P, R05D y R06B. Fuente: Elaboración propia.

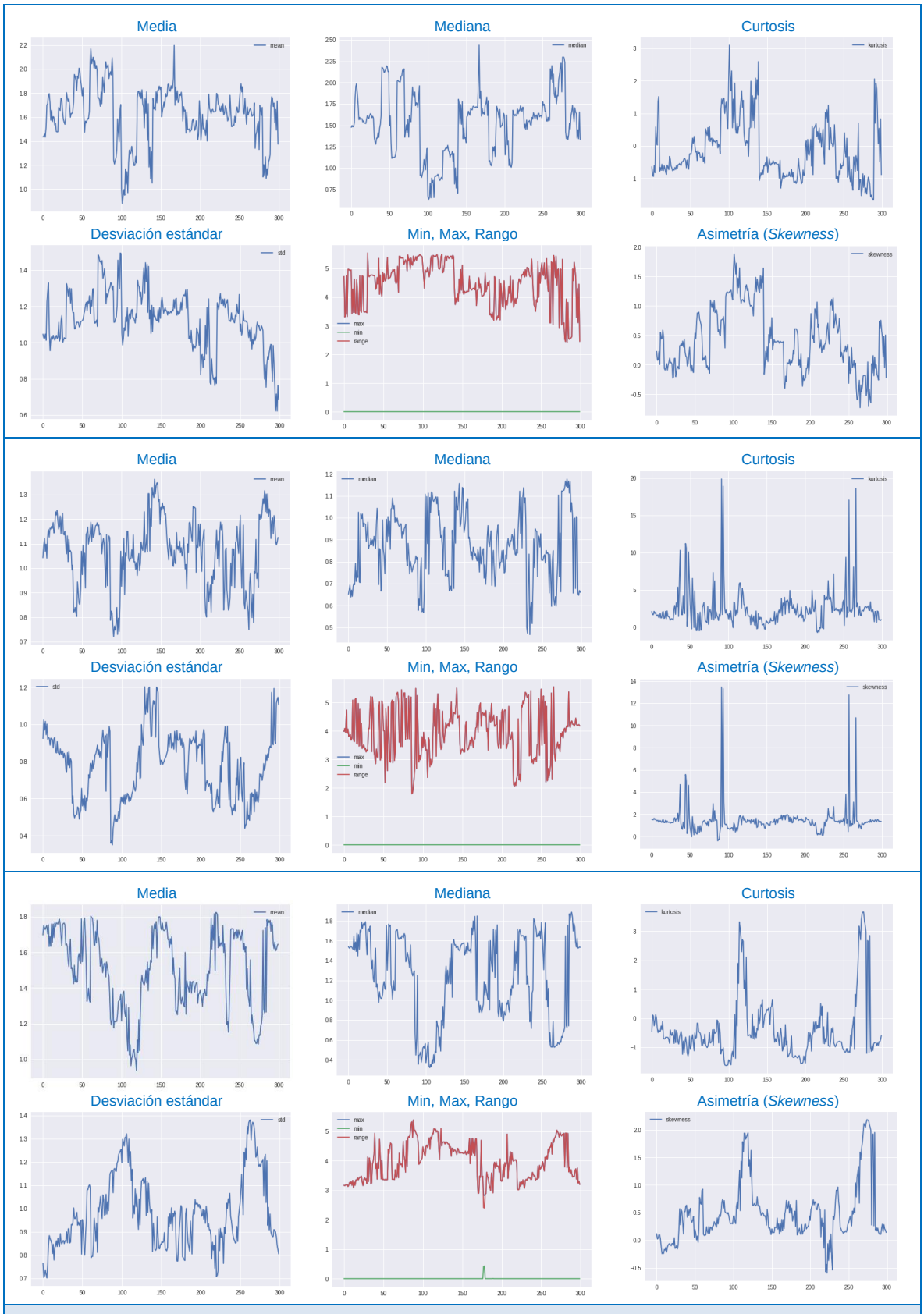


Figura A3.5. Datos de las "*features*" de las estancias de la vivienda 3ME.
Habitaciones (empezando por arriba): R01S, R02C y R03H. Fuente: Elaboración propia.

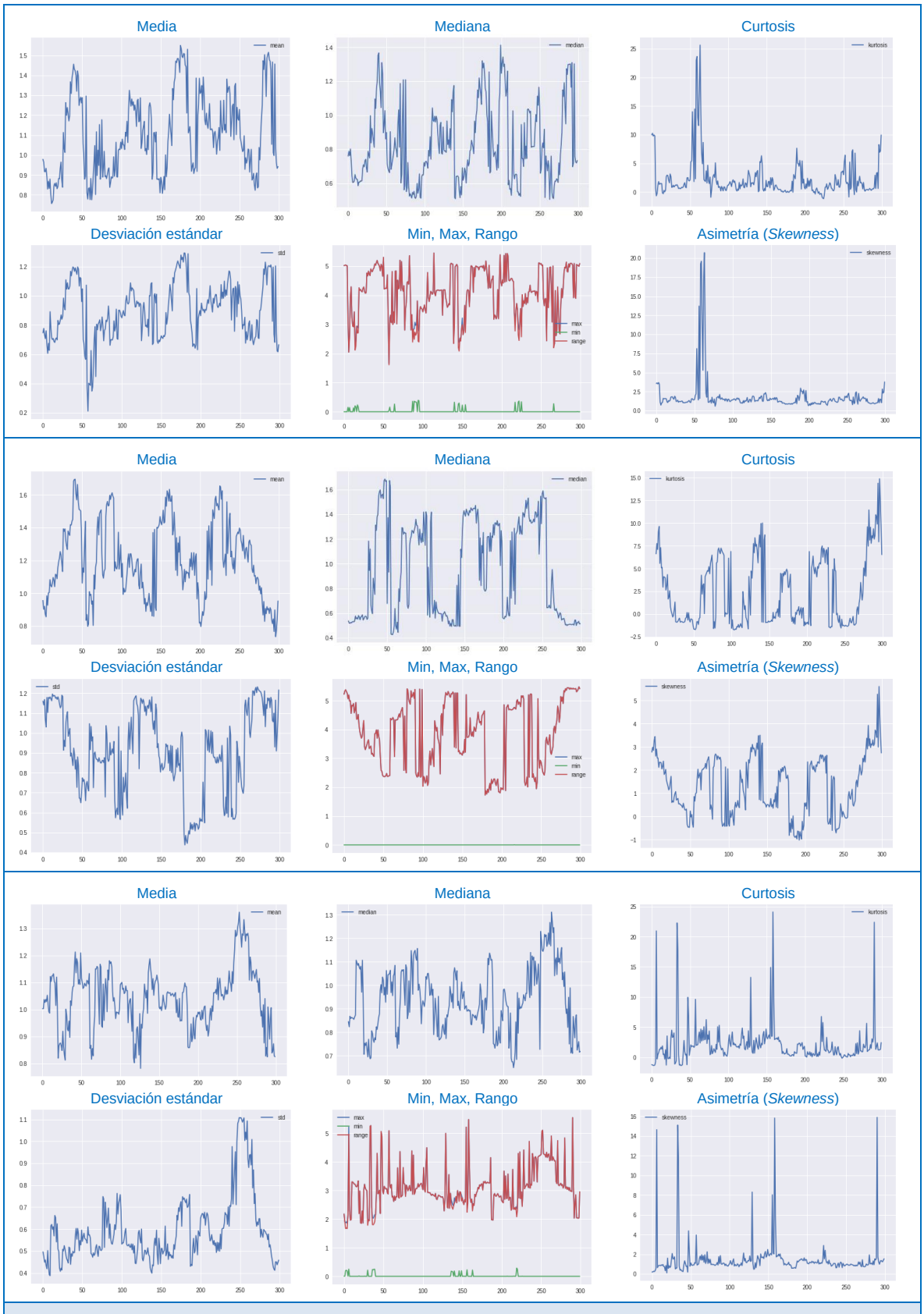


Figura A3.6. Datos de las "*features*" de las estancias de la vivienda 3ME.
Habitaciones (empezando por arriba): R04P, R05D y R06B. Fuente: Elaboración propia.

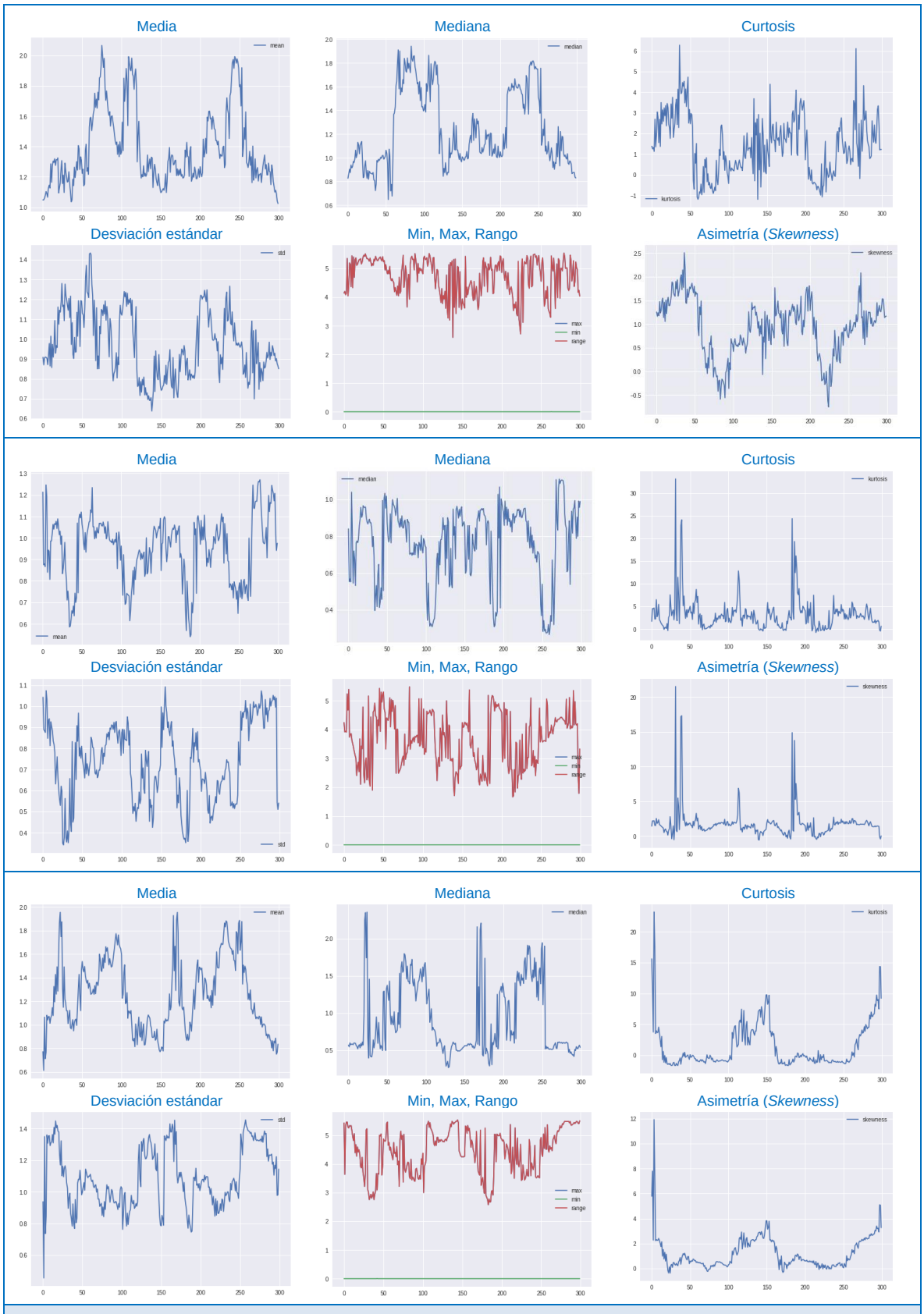


Figura A3.7. Datos de las "*features*" de las estancias de la vivienda 4TR.
Habitaciones (empezando por arriba): R01S, R02C y R03H. Fuente: Elaboración propia.

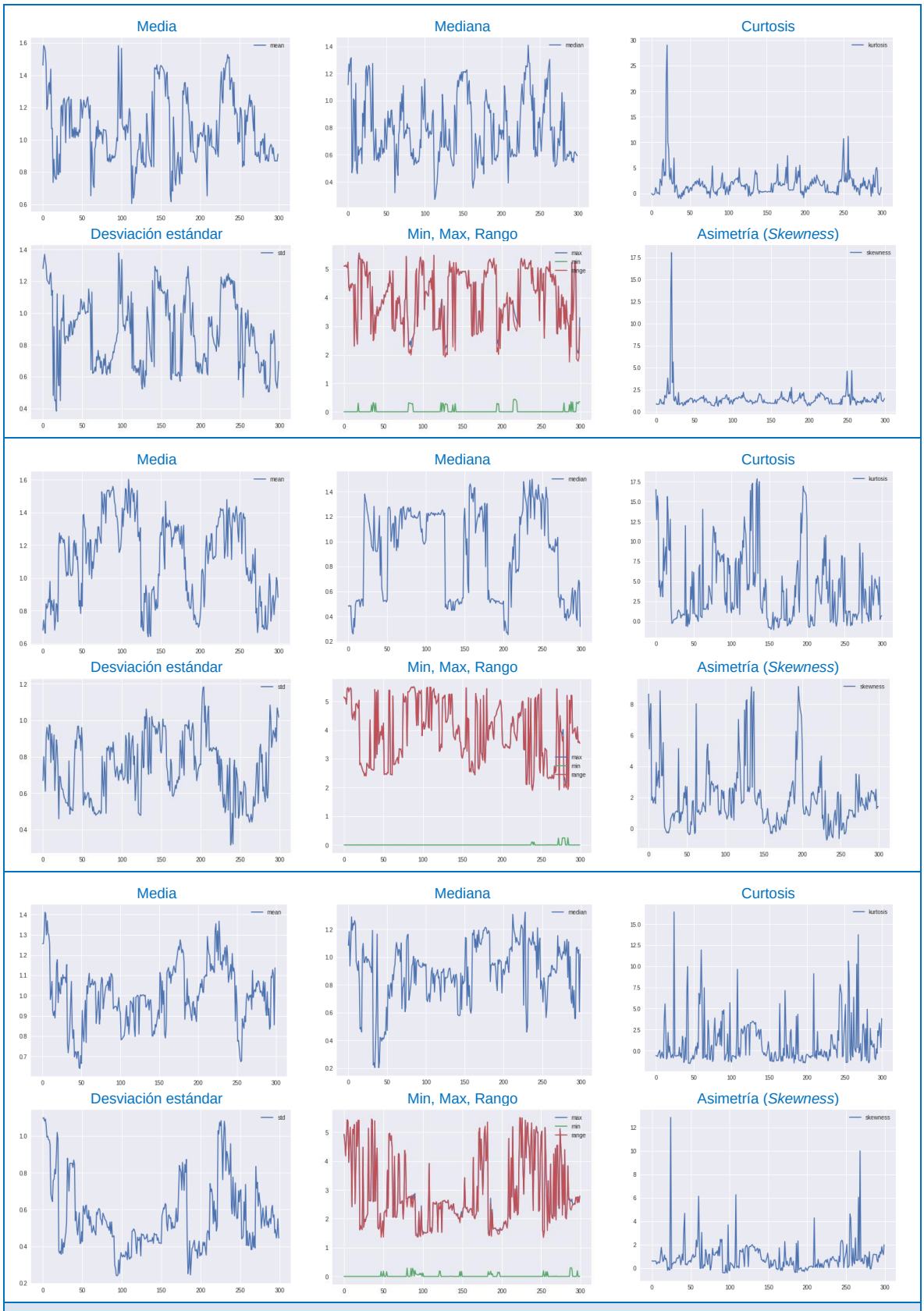


Figura A3.8. Datos de las "*features*" de las estancias de la vivienda 4TR.
Habitaciones (empezando por arriba): R04P, R05D y R06B. Fuente: Elaboración propia.

Anexo 4: Guía-Checklist metodológica

En la Tabla A4.1 se describen varias consideraciones derivadas de la experiencia y los aprendizajes del trabajo realizado. Para darle un carácter práctico, estas ideas se han formulado como preguntas para invitar a la reflexión en cada fase del proyecto.

Tabla A4.1. Preguntas de reflexión por fase del proyecto

F0. GESTIÓN DEL PROYECTO	• ¿Qué riesgos comporta el proyecto? ¿Cómo gestionarlos? • ¿Qué oportunidades de valor (negocio, social, etc.) existen? • ¿Qué sinergias se pueden desarrollar entre el trabajo propuesto y otras iniciativas, investigaciones o productos?
F1. DESARROLLO METODOLÓGICO	• ¿Cómo organizar el trabajo para facilitar su ejecución? • ¿Cómo agilizar el aprendizaje durante el proyecto? • ¿Qué entregables e hitos aceleran la creación de valor?
F2. REVISIÓN BIBLIOGRÁFICA Y DOCUMENTAL	• ¿Qué áreas de investigación y aplicaciones son relevantes? • ¿Qué experiencias previas y resultados anteriores existen? • ¿Qué fuentes de información usar para garantizar una diversidad de perspectivas acorde a los retos del proyecto?
F3. DISEÑO DE EXPERIMENTOS	• ¿Qué hipótesis y asunciones se quieren validar o rechazar? • ¿Cómo plantear los experimentos para potenciar las oportunidades de descubrir información de valor? • ¿Qué métricas usar para evaluar el desempeño y resultados?
F4. PREPARACIÓN Y CONFIGURACIÓN DEL SISTEMA	• ¿Cómo mantener una visión sistémica del proyecto? • ¿Qué partes y parámetros del sistema son clave para pivotar y/o extrapolar los resultados de los experimentos? • ¿Cuáles son las interrelaciones críticas entre componentes?
F5. PROGRAMACIÓN DE ALGORITMOS Y MODELOS	• ¿Qué algoritmos y modelos pueden generar resultados más interesantes (ej. que aún no se hayan usado en estos retos) • ¿Cómo integrar varios algoritmos y modelos (ej. *ensembles*) • ¿Qué datos, parámetros e hiperparámetros son relevantes?
F6. RECOGIDA DE DATOS CON EL AGENTE ROBÓTICO	• ¿Cómo realizar el etiquetado de los datos (ej. anonimizar)? • ¿Qué conocimiento extraer (ej. grafo topológico-semántico)? • ¿Cómo trabajar incremental e iterativamente con los datos?
F7. ANÁLISIS Y COMPARATIVA DE RESULTADOS	• ¿Qué modelos y algoritmos tienen mejor o peor desempeño? • ¿Qué factores causan – y/o explican – estas diferencias? • ¿Qué alternativas explorar para mejorar los resultados?
F8. OBTENCIÓN DE CONCLUSIONES	• ¿Qué conclusiones se derivan de los resultados del trabajo? • ¿Cuáles han sido y cómo aplicar las lecciones aprendidas? • ¿Qué líneas de trabajo ofrecen oportunidades de futuro?