Navindoor: Plataforma de simulación para el diseño, prueba y evaluación de sistemas de localización

***Resumen*—En este trabajo se presenta la plataforma de simulación, Navindoor. Esta es una plataforma desarrollada en MATLAB para el diseño, prueba y evaluación de sistemas de localización, que provee de herramientas para la definición del escenario, la generación de trayectorias, la simulación de señales sintéticas, el procesamiento de estas señales y la comparación de algoritmos. Navindoor se ha diseñado de forma modular, de manera que los modelos de simulación y algoritmos de procesamiento sean independientes de la plataforma. De esta forma nuevos modelos y algoritmos pueden ser fácilmente implementados.**

***Index Terms*—localización, procesamiento de señales, simulador**

I. Introdución

Las metodologías utilizadas para el diseño, pruebas y validación de sistemas de localización son tan variadas como la diversidad de tecnologías y técnicas usadas para su implementación. Sin embargo, existen una serie de pasos comunes en el diseño de sistemas de localización. Estos son: la obtención de la planimetría, la generación de trayectorias, la toma de medidas de señales en dicha trayectoria, el procesamiento de las señales y por último la evaluación de su rendimiento por comparación. Las medidas tomadas deberán ser variadas, ya que una colección en pocas casuísticas puede agregar sesgos no deseados a nuestros sistemas. En consecuencia es necesario realizar pruebas en varios entornos, tomando medidas en distintas trayectorias. Sin embargo, la obtención de medidas experimentales de este tipo son costosas en tiempo y recursos.

En consecuencia las herramientas de simulación son una alternativa muy interesante. Buenos modelos de simulación, tanto en la trayectoria, como en la generación de señales pueden generar un amplio banco de pruebas. Sin embargo, por ahora no existe un marco estándar donde desarrollar algoritmos de diversa índole. Los pocos simuladores que existen suelen estar enfocados en alguna tecnología en concreto, lo que no permite la comparación con otros tipos de sistemas de localización.

En este artículo se presenta la plataforma de simulación, Navindoor [1]. Esta es una librería para MATLAB de código abierto, en donde el usuario puede construir la planimetría, generar trayectorias, simular señales, desarrollar algoritmos y compararlos con otros. Navindoor contiene una interfaz gráfica (GUI), que nos facilita la interacción con la plataforma. Además, contiene una interfaz de línea de comando (CLI), que facilita automatizar simulaciones voluminosas y la integración con otras herramientas. A diferencia de otras plataformas de simulación de sistemas de localización, en donde los modelos de simulación están anclados al diseño, este se ha desarrollado de forma modular. Debido a esto, la implementación de nuevos modelos y algoritmos es más versátil. Dentro de Navindoor podemos utilizar funciones de MATLAB como parámetros de entrada. De esta manera desarrollar un modelo de simulación o algoritmo de localización se simplifica a crear una función de MATLAB, con una interfaz de entrada/salida específica.

El resto del documento se organiza de la siguiente manera: En la sección II, se hará un revisión sobre simuladores en el ámbito de la localización. En la sección III, se describirá el diseño de Navindoor. Por último en la sección IV se resumirá lo expuesto y se presentará futuros desarrollos.

II. Estado del arte

Los simuladores son una gran herramienta para mejorar el diseño, prueba y validación de sistemas de localización. Es por ello que podemos encontrar distintos intentos en la comunidad científica. A continuación se resumirá algunos trabajos existentes en el ámbito, además de describir sus principales características.

SMILe [2] es una herramienta de simulación de código abierto, para desarrollar y evaluar métodos de localización en interiores basados en el tiempo de propagación como métricas de localización, tales como el Tiempo de Vuelo (ToF) o la Diferencia de Tiempo de Llegada (TDoA). Esta herramienta está basada en el simulador OMNET+++ y un paquete de funcionalidades escritas en Python para el análisis y el procesamiento de datos. SMILe permite al usuario configurar un espacio, donde se pueden cambiar varios factores que afectan significativamente el rendimiento de la localización. Estos factores incluyen el despliegue de diferentes nodos, capacidades de radio y la inexactitud de relojes.

Tanto en Navindoor como en SMILe existen herramientas necesarias para el diseño, prueba y evaluación de algoritmos de localización a partir de ToF. Sin embargo, mientras que en SMILe señales de otra índole todavía no están contempladas, en Navindoor se abarcar más diversidad de tecnologías. Aun así, el modelo de simulación de señales de ToF de SMILe es más complejo que el modelo de simulación de Navindoor.

PyLayers [3] es un simulador de radio frecuencia de código abierto. Se ha diseñado para evaluar el rendimiento de algoritmos de localización. El canal de radio se sintetiza mediante el uso de un método de trazado de rayos basado en gráficos. El movimiento de personas se modela con un enfoque de fuerzas virtuales. Los datos simulados se pueden procesar directamente con uno de los algoritmos de localización incorporados o

se pueden exportar a varias extensiones para el procesamiento externo.

Al igual que en el caso anterior, PyLayers contiene un simulador complejo para una tecnología concreta. Aunque ofrece la posibilidad de procesar la señales dentro del propio simulador, hace énfasis en la exportación de sus resultados para el procesamiento externo. En Navindoor, se opta por ofrecer un interfaz para la incorporación de nuevos modelos de simulación, manteniendo todo el proceso en un mismo entorno de desarrollo.

Sensor Fusion and Tracking Toolbox [4] es una librería desarrollada por Mathworks que incluye algoritmos y herramientas para diseñar, simular y analizar sistemas que fusionan datos de varios sensores con el objetivo de hacer seguimiento de la posición y orientación de objetos. Esta librería incluye funcionalidades para la generación de trayectorias y escenarios, simular mediciones de sensores inerciales (acelerómetro, giroscopio, magnetómetro), receptores de GPS, radar, sonar e infrarrojo, diferentes algoritmos de fusión y estimación (filtros de Kalman y filtros de partículas), y herramientas para visualizar, analizar y comparar el rendimiento de los diferentes algoritmos.

En está librería, al igual que en Navindoor, se provee de herramientas en todo el proceso de diseño de sistemas de localización. Sin embargo, en *Sensor Fusion and Tracking Toolbox* por ahora solo se ha desarrollado una CLI, careciendo de GUI. Por otra parte, los modelos de simulación en la generación de trayectorias son generales, no simulan un activo en concreto. En Navindoor los modelos de simulación están enfocados al movimiento de personas.

III. Descripción de la plataforma Navindoor

Navindoor ha sido desarrollado bajo el paradigma de la programación orientada a objetos. Se han definido clases para representar elementos en el diseño de sistemas de localización. La planimetría, las trayectorias o las señales son clases que contiene métodos asociados para su visualización, la extracción de datos e interacción entre ellos. Por otro lado, tenemos los algoritmos de localización y las métricas para comparar algoritmos. Estas están representadas por funciones que actúan sobre las clases antes mencionadas.

Estas clases y funciones se han desarrollado en cinco módulos que siguen el proceso de experimentación en el diseño de sistemas de localización. Para tener una visión global de ello, se describirá brevemente este proceso y el módulo propuesto en cada punto:

1. Obtención de la planimetría. El primer paso en la experimentación es la elección del escenario y la obtención de la planimetría de este. En Navindoor, se ha creado el módulo de planimetría con la clases necesarias para construir una planimetría de varios niveles.
2. Generación de trayectorias. En la experimentación, se toman puntos de referencia para poder construir trayectorias a través de ellas. Esto limita la variedad de las trayectorias, sin embargo es necesario para obtener medidas precisas de la posición en cada instante. En Navindoor se ha creado el módulo de generación de trayectorias, dedicado a la simulación de trayectorias. Debido a que estamos en un entorno de simulación tenemos información exacta sobre la trayectoria real, generada a partir de una sucesión de puntos proporcionada por el usuario.
3. Toma de medidas de señales. En la experimentación, una vez generada la trayectoria, se debe enriquecer esta con señales. El tercer módulo, llamado modulo de generación de señales, ésta dedicado a la simulación de señales sintéticas. Gracias a la simulación, podemos crear las señales después de haber sido generada la trayectoria.
4. Procesamiento de señales. Tras la toma de medidas, los algoritmos de localización se encargan de procesar las señales. El cuarto módulo de Navindoor, llamado módulo de procesamiento de medidas, provee algoritmos capaces de crear estimaciones de la trayectoria real a partir de las señales simuladas. Además de ofrecer un esquema donde nuevos algoritmos puedan ser desarrollados.
5. Comparación de Algoritmos. Por último, para poder validar los algoritmos desarrollados es necesario la comparación con otros algoritmos ya consolidados. El quinto módulo de Navindoor es el módulo de validación de algoritmos, que nos provee de métodos de comparación de algoritmos proporcionados por el usuario o entre algoritmos por defecto.

Dentro de estos módulos se encuentran repartidas las clases y funciones definidas en Navindoor. Cabe mencionar que tanto el CLI, como el GUI son compatibles en versiones de MATLAB superiores a MATLAB R2017b y se encuentran disponibles en [1].

A continuación se expondrán los detalles de cada uno de los módulos.

III-A. Módulo de planimetría

El objetivo de este módulo es caracterizar el escenario donde transcurre el experimento. Actualmente las trayectorias en Navindoor se centran en el desplazamiento en interiores, por lo que se ofrece las herramientas básicas para caracterizar un edificio de varias plantas. Esta caracterización nos proveerá información sobre las restricciones de movimiento, posición de puntos de acceso, altura entre plantas, etc. Esta información puede ser usada por los modelos generación de trayectorias y señales, además de los algoritmos de localización.

Se ha diseñado una jerarquía de clases presentado en la figura 1, que representarán los objetos necesarios para la caracterización de un edificio. Se ha creado la clase *node*, para representar un punto en el espacio tridimensional. A partir de este objeto se construyen las demás clases dentro del módulo de planimetría. Los objetos *walls*, que representa las paredes del edificio, son construidos a partir de dos objetos *node*. Por otra parte, también existen elementos puntuales, por lo que se ha creado clases que heredan las propiedades de *node*. Estos son los objetos *stairs*, *elevators*, *doors* y *beacons*. Los

objetos *stairs* y *elevators* nos indican los puntos de salida de una planta. Los objetos *doors*, se definen dentro de los objetos *walls*, y permiten el paso en su entorno. Por último, los objetos *beacons* representan puntos de acceso generadores de señales de radio frecuencia. Los objetos antes mencionados se almacenan en una clase llamada *level* en forma de atributos. Por último se ha creado la clase *building* que en sus atributos contiene un lista de objetos *level*.

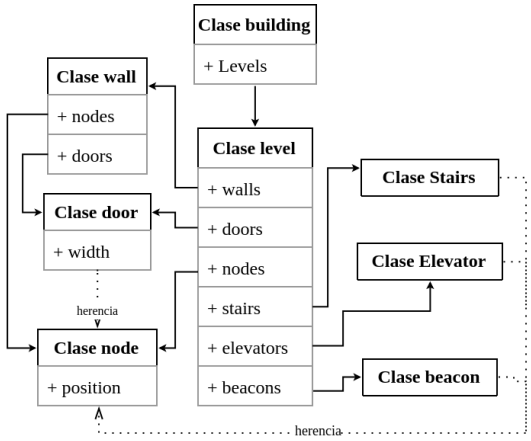

Figura 1. Esquema de la clases.
Las flechas salientes indican de qué clase es el objeto. Mientras que el listado dentro de cada caja representa las propiedades que contiene una clase.

Para la creación de cada uno de estos objetos existe un constructor siguiendo las bases de la programación orientada a objetos, sin embargo la caracterización de la planimetría de esta forma puede ser muy tediosa. Es por ello que se ha optado en el desarrollo de una GUI (figura 2), que nos ayude en este trabajo. De esta forma se puede definir objetos *nodes* con simples *clicks*, y paredes uniendo objetos *nodes*. Además la GUI permite cargar imágenes, en formato PNG, de la planimetría de interés dentro del espacio de trabajo. Esto se puede realizar para cada una de las plantas de manera que podamos crear una fiel reproducción de la planimetría utilizando los planos reales como plantillas. Un vez construido la planimetría podemos ver el objeto *building* en 3D, gracias a las opciones de la GUI.

III-B. Módulo de generación de trayectorias

El objetivo de este módulo es generar trayectorias dentro de la planimetría creada. La modelización de las trayectorias nos permite conseguir una vista preliminar del comportamiento de los algoritmos. Los modelos de movimiento toman importancia en este módulo, ya que de ello dependerá la similitud de los resultados obtenidos con la realidad.

En Navindoor, las trayectorias están modeladas como una sucesión de puntos dentro de la planimetría. Estos puntos definen los tramos por donde se moverá el activo. Estos se pueden definir con la ayuda de GUI, simplemente haciendo *click* dentro de la planimetría, incluyendo los cambios de planta.

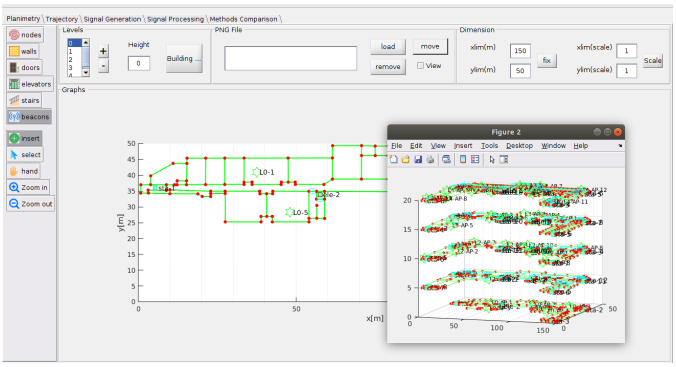

Figura 2. Interfaz gráfica para el diseño de la planimetría *building*.
En la imagen se muestra la interfaz con un edificio de cinco plantas. La figura externa es una representación tridimensional del edificio.

Un vez definida los puntos por donde pasará el activo, con ayuda de modelos de simulación se genera una sucesión de puntos más fina. Esta contiene coordenadas de los puntos, además del instante de tiempo correspondiente. De esta forma, las velocidades y aceleraciones pueden ser obtenidas mediante diferenciación. Los modelos de generación de la trayectorias en nuestro caso están centrados en la simulación del pie de una persona, por lo que se simula medidas de los sistemas inerciales *foot-mounted*. En concreto se utiliza el modelo propuesto en [5]. Además en los casos de cambios de planta se agrega un modelo de velocidad constante.

A partir de la trayectoria del pie de la persona, se genera la trayectoria del centro de masas. Esta se usará en los casos en el que el sensor esté sobre el tronco de la persona. En la figura 3 se muestra el proceso para la generación de un objeto trayectoria.

Es importante notar que tanto los modelos de simulación de la trayectoria del pie, como el modelo de generación de la trayectoria del centro de masas, son parámetros opcionales. Es decir, los modelos de simulación son independientes de Navindoor, por lo que nuevos modelos pueden ser implementados creando funciones con la misma interfaz de entrada/salida que las funciones ofrecidas por defecto.

Por último, en la GUI tenemos herramientas para la generación de la trayectoria, desde la opción para crear una sucesión de puntos con *clicks*, pasando por la creación de nuevos modelos, hasta la visualización de la trayectoria generada (figura 4).

III-C. Módulo de generación de señales

En este módulo se podrán simular distintos tipos de señales a partir de la trayectoria generada. El objetivo es generar datos que los algoritmos de localización procesarán. Al igual que en el módulo anterior, los modelos de simulación son cruciales para que el comportamiento del simulador sea fiel a la realidad.

En Navindoor se ha creado la clase abstracta *Signal* con propiedades que contiene la información de las medidas a lo largo de un periodo de tiempo. Además, se ha creado dos clases que heredan los métodos y propiedades de la clase *Signal*. Estas son:

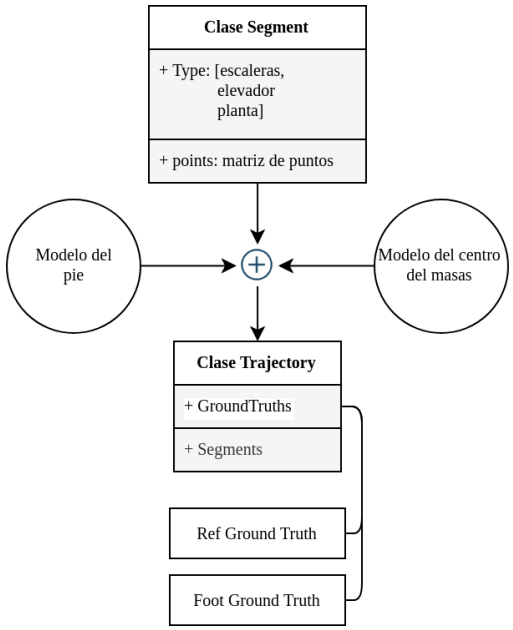

Figura 3. Proceso de construcción de una trayectoria.
En este esquema se puede ver cómo el constructor de la trayectoria, genera dos objetos *GroundTruths* a partir de segmentos que contienen información extraída de la planimetría. Además, es necesario un modelo que construya la trayectoria del pie de una persona a través de los segmentos y un modelo que sea capaz de crear la trayectoria del centro de masas a partir de la trayectoria del pie. El objeto *Ref GroundTruth* representa la trayectoria del centro de masas, mientras que *Foot GroundTruth* representa la trayectoria del pie de la persona.

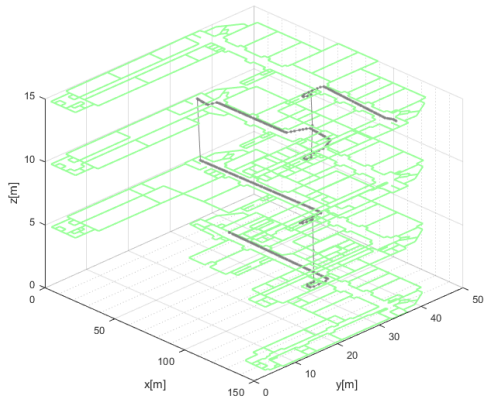

Figura 4. Captura de una animación de una trayectoria
Esta tiene su inicio en la planta cinco y baja usando distintas escaleras y ascensores hasta la primera planta.

- La clase *BeaconBased*: Representando señales que necesitan puntos de acceso para poder ser simuladas.
- La clase *BeaconFree*: Representando señales que pueden ser simuladas sin la necesidad de los puntos de acceso.

Dentro del entorno de simulación, las diferencias entre estas clases se ven en la forma de construirlas. Mientras que la clase *BeaconFree*, solo necesita la información de la trayectoria para ser definida, la clase *BeaconBased* necesita los objetos *beacons* definido en el módulo de la planimetría (figura 5).

La clase *BeaconBased* puede representar señales de atenuación de nivel de potencia (RSS), ángulo de llegada (AoA) y tiempo de vuelo (ToF), mientras que la clase *BeaconFree* puede representar señales inerciales (aceleraciones lineales y velocidades angulares), intensidades de campo magnético y presión atmosférica. Los modelos de simulación de señales utilizados por defecto son básicos con una componente de ruido gaussiano.

En Navindoor se trata a los modelos como parámetros de entrada, por lo que es posible cambiarlos por modelos más complejos simplemente creando funciones con una interfaz de entrada/salida específica. En el caso de las clases *BeaconBased*, los parámetros de entrada deberán ser un punto de la trayectoria y un objeto *beacon*; y dar como parámetro de salida el valor de la señal. Mientras que en el caso de la clase *BeaconFree*, solo se usa un punto de la trayectoria como parámetro de entrada y el valor de la señal como parámetro de salida.

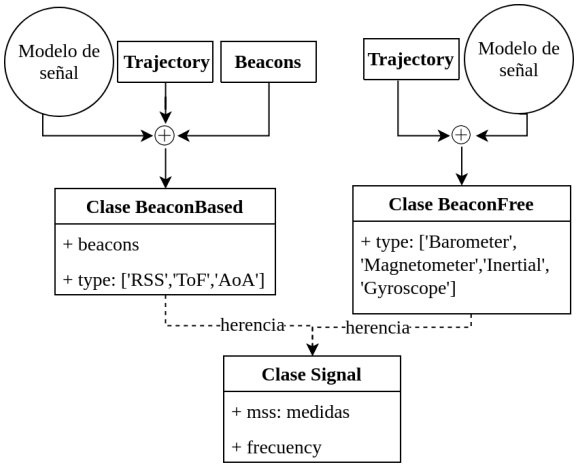

Figura 5. Representación de la generación de las señales.
Se muestra un esquema de la generación de las señales *BeaconFree* y *BeaconBased*. Además, también se muestra la clase *Signal* de la cual heredan las propiedades donde estarán las medidas.

Por la parte de la GUI, en Navindoor se ha implementado un apartado para la generación de señales. En esta se muestra un listado de trayectorias previamente generadas. También se muestra un selector donde poder elegir el tipo de señal que queramos generar. Para cada uno de estos tipos nos enseña un listado de los modelos existentes. Además nos permite crear nuevos modelos a partir de una plantilla para cada tipo de señal. De esta manera dentro de la GUI podemos crear varias señales rápidamente, además de visualizar el resultado obtenido.

III-D. Módulo de procesamiento de medidas

El objetivo de este módulo es proveer al usuario algoritmos básicos para la localización a partir de las señales generadas. Estos algoritmos se utilizarán como punto de partida para el diseño de nuevos.

En Navindoor se encuentra implementados el filtro de Kalman extendido (EKF) [6] y el Filtro de Kalman *Unscented*

(UKF) [7]. Para implementar nuevos algoritmos es necesario que estos tengan una interfaz de entrada/salida específica. Los algoritmos deberán tener tres parámetros de entrada: El objeto *building*, una lista de las señales generadas en todos los instantes de tiempo y una estructura con toda la información a *priori* de la trayectoria. Además deberán tener como parámetro de salida una matriz de cuatro columnas. Las tres primeras columnas indicando el espacio y la última columna el tiempo (figura 6). Navindoor se encarga de procesar esta información para crear un objeto trayectoria. De esta manera la estimación contará con las propiedades y métodos de la trayectoria real.

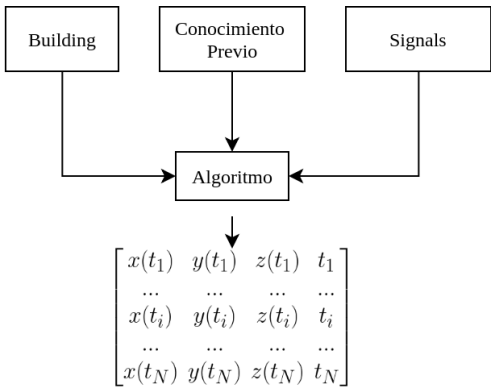

Figura 6. Interfaz de entrada/salida para los algoritmos de localización Como entrada tiene tres variables, la primera representando la planimetría, la segunda representado todo el conocimiento previo sobre la trayectoria y el tercero un listado de las señales. El algoritmo devuelve una matriz $4 \times N$, siendo N el número de medidas

Con respecto a la GUI, Navindoor muestra un listado de las trayectorias que han sido generadas. Para cada una de estas trayectorias podemos ver las señales disponibles. De esta manera podemos seleccionar la trayectoria y las señales que queramos procesar. Dado que los algoritmos tienen el interfaz entrada/salida antes presentado, es posible ejecutarlos con un solo *click*. Navindoor, luego de ejecutar el algoritmo muestra la trayectoria real y la trayectoria estimada dentro de la planimetría cargada. Además gracias a que la estimación es un objeto trayectoria, podemos utilizar la animación utilizada en el módulo de generación de trayectorias, dando una visión a tiempo real del comportamiento de la trayectoria real frente a la trayectoria estimada.

III-E. Módulo de evaluación de algoritmos

El objetivo de este último módulo es comparar distintas estimaciones de la trayectoria. La diferencia de las ejecuciones puede estar en el algoritmo utilizado para estimar, sin embargo también puede estar en los modelos de simulación de la trayectoria, en los modelos simulación de señales, o simplemente la frecuencia de muestreo de alguna señal. Deberemos notar que las estimaciones en Navindoor son objetos trayectorias por lo que desde el punto de vista del código, no existe diferencia estructural entre la trayectoria real y la estimación. Esto hace que la comparación entre ellas sea más fácil.

En la GUI, Navindoor nos muestra un lista de las trayectorias generadas, pudiendo seleccionar una de ellas. Para cada una de estas trayectorias se muestran las estimaciones disponibles. Podemos seleccionar las estimaciones que queramos para compararlas en un mismo gráfico representando la función de distribución acumulada empírica (eCDF), tomando como referencia la trayectoria real. De esta forma tenemos una visión del comportamiento de las dos estimaciones en un mismo gráfico.

IV. Conclusiones

Se han mostrado las principales características de Navindoor. Esta plataforma ha sido desarrollada para ser escalable, donde los propios usuarios puedan contribuir con modelos de simulación de trayectorias, modelos de señales y con algoritmos de localización. Aunque en este momento los modelos implementados son básicos, su estructura modular es capaz de integrar modelos complejos.

A continuación mencionaremos los futuros desarrollos en la plataforma Navindoor.

Con respecto al módulo de la planimetría, es importante notar que la clase *building* es una abstracción de un edificio, por lo que la traslación a otros formatos como *XML* o *JSON* es inmediata. Este desarrollo permitirá comunicaciones con plataformas como *Open Street Maps*, permitiendo el uso de señales *GNSS*, además de la fusión de tecnologías de localización en interiores con tecnologías de localización globales.

En cuanto a la simulación de trayectorias, se han implementado modelos de simulación de personas. Sin embargo esto puede generalizarse a otros tipos de activos como pueden ser los drones, robots, etc. Gracias a la fácil implementación de modelos, Navindoor es una plataforma muy versátil.

Por parte de la simulación de señales, gracias a las APIs de MATLAB para otros lenguajes de programación, es posible la integración de otros simuladores en distintos lenguajes. Simuladores como los presentados en la sección II, pueden ser integrados.

Por último, se agregarán nuevas métricas para la comparación de trayectorias, siguiendo la misma filosofía modular tomada en los modelos de simulación y los algoritmos. De esta forma, podremos crear métricas de localización personalizadas e independientes a la plataforma.

Referencias

[1] DeustoTech. navindoor-code. GitHub; 2018. https://github.com/DeustoTech/navindoor-code.

[2] Jankowski T, Nikodem M. SMILe – Simulator for Methods of Indoor Localization. 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN). 2018;(September):24–27.

[3] Amiot N, Laaraiedh M, Uguen B. PyLayers: An open source dynamic simulator for indoor propagation and localization. 2013 IEEE International Conference on Communications Workshops, ICC 2013. 2013;(July 2014):84–88.

[4] Mathworks. Sensor Fusion and Tracking Toolbox; 2018. Available from: https://es.mathworks.com/help/fusion/.

[5] Zampella FJ, Jiménez AR, Seco F, Prieto JC, Guevara JI. Simulation of foot-mounted IMU signals for the evaluation of PDR algorithms. 2011 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2011. 2011;.

[6] Kay SM. Fundamentals of Statistical Signal Processing: Estimation Theory. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.; 1993.

[7] J Julier S, K Uhlmann J. A New Extension of the Kalman Filter to Nonlinear Systems. Proc SPIE. 1999 02;3068.