Navindoor: Software de simulación para algoritmos localización

***Resumen*—En este artículo se presentará la herramienta de simulación, Navindoor. Esta es una librería para MATLAB con las herramientas necesarias en el diseño y comprobación de sistemas de localización. Ésta nos ofrece la posibilidad de desarrollar modelos de simulación de trayectorias, modelos de simulación de señales asociadas a ésta, además de desarrollar y probar algoritmos de localización. Navindoor es un espacio de trabajo donde desarrollar modelos de simulación y algoritmos relacionados con la localización.**

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

I. Introducción

Las metodologías utilizadas para los sistemas de localización son tan variadas como la diversidad de sensores existentes. Sin embargo, aunque algoritmos y tecnologías sean diversas, existe una serie de pasos comunes en el desarrollo de un algoritmo de localización. Estas son, la toma de medidas experimentales de señales, así como medidas de la propia trayectoria. Esta medidas deben ser variadas, ya que una recolección en pocas casuísticas puede agregar sesgos no deseados a nuestros algoritmos. Es por ello que es necesario realizar pruebas en varios entornos con distintas trayectorias. Sin embargo, la obtención de medidas experimentales de este tipo son costosas en dinero y tiempo.

Es por ello que la simulación de todo el desarrollo es una propuesta 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 estandar donde desarrollar algoritmos de diversa índole. Los pocos simuladores que existen, suelen desarrollar en detalle alguna tecnología en concreto, obteniendo un gran simulador pero rígido ante la comparación entre algoritmos o modelos de simulación externos al simulador.

En este artículo se propone el *software* de simulación, Navindoor[1]. Este es un *software* para MATLAB, en donde el usuario puede construir escenarios, simular trayectorias, simular señales, desarrollar algoritmos y compararlos con distintas métricas. Navindoor contiene una interfaz gráfica (GUI), que nos facilita en el proceso de aprendizaje del *software*. Sin embargo, Navindoor no solo contiene una GUI, si no también una interfaz de lineas de comando (CLI). Esto facilita automatizar algunas tareas, de esta manera usuarios con experiencia, no se ven limitados por las restricciones propias de toda GUI. Pudiera parecer que Navindoor es una estructura rígida, en donde los modelos de simulación estan anclados al diseño del *software*, sin embargo, éste se ha desarrollado de forma que la implementación de nuevos modelos y algoritmos sea inmediata. 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 funcion de MATLAB, con los parámetros de entrada/salida correctos.

El resto del documento seguiremos de la siguiente manera: En la sección II, se hará un pequeña revisión sobre los algunos simuladores, con el mismo propósito. Luego en la sección III, discutiremos sobre las principales características y el diseño de Navindoor. En la sección IV mostraremos un pequeño ejemplo de uso, y por último en la sección V mostraremos un pequeño resumen de lo expuesto y de las siguientes desarrollos.

II. Estado del arte

Los simuladores son una gran solución para mejorar el desarrollo de la sistemas de localización, es por ello que dentro la comunidad científica existen distintos intentos. A continuación mencionaremos algunos trabajos relacionados con el tema, ademas de sus principales características.

II-A. SMILe

SMILe [1] es un software que propone una solución de simulación completa y unificada que ayude al desarrollo y evaluación de los métodos de localización basados en ToF. El objetivo es proporcionar una herramienta de simulación bien definida y altamente configurable, donde se puedan evaluar varios métodos de localización. SMILe permite al usuario configurar un espacio, donde se pueden configurar varios factores que afectan significativamente el rendimiento general de la localización. Estos factores incluyen: despliegue de diferentes nodos, capacidades de radio, inexactitud de relojes de hardware. Aunque SMILe, nos provee herramientas necesarias para la comparación de algoritmos de estimación de la posición a partir de señales ToF, señales de otra índole todavía no esta implementadas. Por lo que lo hace muy potente en su especialidad y débil en otras tecnologías.

II-B. PyLayers

PyLayers [2] es un simulador de radio frecuencia. El canal de radio se sintetiza mediante el uso de un método de trazado de rayos basado en gráficos, de esta forma es capaz de simular el efecto de la reflexión de las ondas. PyLayers, esta desarrollado en python, y esta pensado para ser independientes de los algoritmos de procesamiento. Al igual que el simlulador

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

anterior, aunque existe un esfuerzo por crear herramientas para todo el proceso de la localización, PyLayers se centra en las señales de radio frecuencia dejando de lado tecnologías inerciales.

II-C. Sensor Fusion and Tracking Toolbox

Sensor Fusion and Tracking Toolbox [3] es un toolbox desarrollado por Mathworks, que incluye algoritmos y herramientas para el diseño, simulación y análisis de sistemas que fusionan datos de múltiples sensores para sistemas de localización. Con este toolbox se puede importar y definir escenarios y trayectorias, transmitir señales y generar datos sintéticos para sensores, incluidos sensores de RF, acústicos, EO / IR y GPS / IMU. También puede evaluar la precisión y el rendimiento del sistema con puntos de referencia estándar, métricas y gráficos animados. Esta toolbox es bastante reciente, por lo que por ahora contiene pocos ejemplos, sin embargo contiene un gran potencial.

III. Diseño de Software

III-A. Creación de Planimetría

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

Con este fin se ha definido clases de MATLAB que representarán los distintos 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. Los objetos que representa las paredes del edifício, son construidos a partir de dos objetos *node*. Existen tambien elementos que tiene una posición puntual, por lo que se ha creado clases que heredan las propiedades de *node*. Estos son objetos *stairs*, *elevators*, *doors*, *beacons*. Los dos primeros nos idican los puntos de escape de una planta. Los objetos *doors*, se definen dentro de los objetos *wall*s, y permiten el paso en su entorno. Por último los objetos *beacons*, representan puntos de acceso generadores de señales de radio frecuencia. Estos elementos son encapsulados en la clase *level*, de modo que los objetos antes mencionados son atributos de los objetos *level*. Por último se ha creado un clase, llamado *building*, que en sus atributos contiene un lista de objetos *levels*. Este esquema de clases se muestra en la figura 1.

Para la creación de cada uno de estos objetos existe un constructor siguiendo las bases de la programación orientada a objetos, sin embargo la carazterización del escenario de esta forma puede ser muy tediosa. Es por ello que se ha optado en el desarrollo de una interfaz gráfica (figura 2), que nos ayude en este trabajo. De esta forma se puede definir objetos *nodes* con simples *clicks*, y paredes uniendo objectos *nodes*. Debido a que el proceso de creación de la planimetría

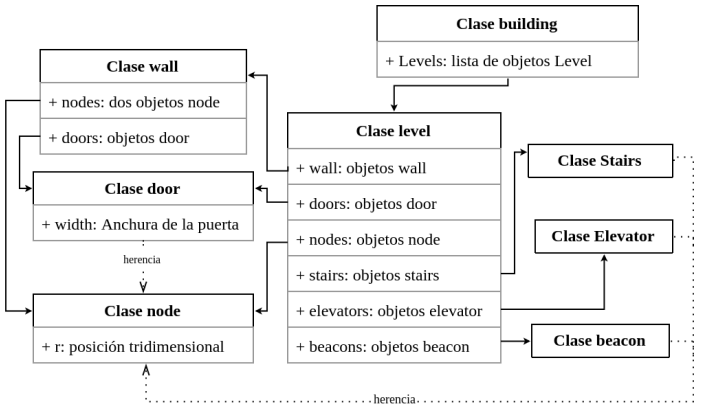

Figura 1. Esquema de la clases
En la imagen se intenta mostra

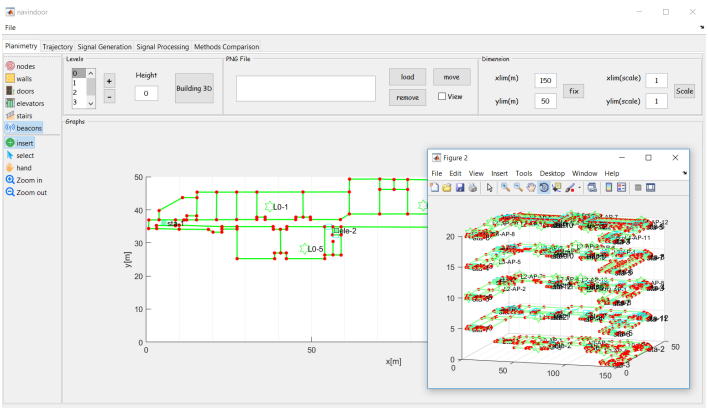

Figura 2. Interfaz gráfica para el diseño de la planimetría *building*. En la imagen se muestra la interfaz con un edificio de cuatro plantas. La figura externa es una representación tridimensional del edificio.

puede ser tedioso, navindoor contiene una GUI capaz de generar un objeto *building* (figura 2). La interfaz nos permite crear los objetos antes mencionados con unos simples *clicks*, además de permitir cargar una imagen de los planos reales con la que ayudarnos a construir la planimetría. Ademaś, cabe mencionar que la GUI contiene herramientas necesarias para la visualización en 3D, la modificación y eliminación de los elementos, haciendo la construcción de la planimetría intuitiva.

III-B. Simulación de las trayectorias

La modelización de las trayectorias nos permite conseguir una primera visión del comportamiento de nuestro algoritmo antes de la pruebas experimentales. Los modelos de movimiento del objetivo toman importancia, ya que de ello dependerá la similitud con la realidad.

En navindoor, las trayectorias son creadas a partir de una sucesión de puntos dentro de una planimetría previamente definida. Estos puntos definen los tramos por donde se transladará el objetivo. Gracias a la información obtenida de la planimetría, se puede generar distintos modelos para cada tramo, tiendo en cuenta sus características.

Un vez definida los puntos por donde pasará el objetivo,

a partir de modelos de simulación, se genera una sucesión de puntos más fina que además de contener las cordenadas del punto, contiene el instante en el tiempo cuando esta en éste. De esta forma, las velocidades y aceleraciones pueden ser obtenidas mediante diferenciación.

Los modelos de generación de la trayectorias estan centrados en la simlulación del pie de un peatón. A partir de ésta se crear otra trayectoria asociada que representa el movimiento del centro de masas, esta es llamada *GroundTruth* de referencia. Estas dos *GroundTruth*s, son indispensable para la definición de la trayectoria. La trayectoria del centro de masas es utilizada par la generación de señales que son medidas en el centro del peatón, mientras que la trayectoria del pie, es utilizada para la generación de señales inerciales. La generación del *GroundTruth* del pie por defecto sigue el modelo de simulación propuesto en el artículo [4], mientras la simulación de cambio de plantas se realiza con un modelo simplificado de velocidad constante. En la figura 3, se muestra los elementos necesarios para la construcción de una clase trayectoria. Es importante notar, que los modelos de simulación del pie y el modelo de generación de *GroundTruth* de referencia son parámtero opcionales dentro del *framework*. Por lo que nuevos modelos de simulación pueden ser fácilmente implementados.

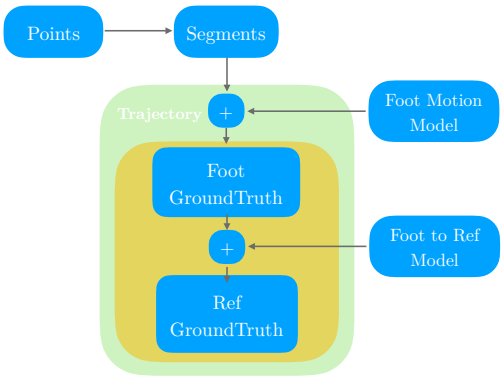

Figura 3. Proceso de construcción de una trayectoria
En este esquema se puede ver como el constructor de la trayectoria, genera los dos *GroundTruths*, a partir de los modelos de simulación seleccionados.

El proceso de generación de trayectoria, con ayuda de la GUI, es la siguiente:

1. Definimos una secuencia de puntos compatibles con la planimetría definida. Con ayuda de la GUI, podemos crear trayectorias compatibles con las restricciones la planimetría.
2. Seleccionamos los modelos de simulación correspondientes. Existen tres tipo de modelos: modelos en un misma planta, modelos a lo largo de una escalera, y modelos a lo largo de los elevadores. Además deberemos seleccionar la función que construirá el *GroundTruth* de referencia. Esta eleecciones se puede realizar con la GUI.
3. Por último, podemos hacer *click* en el botón *Generate!*

Al igual que el módulo de planimetría, tenemos herramientas que nos permite visualizar el resultado obtenido. El botón *animate*, es capaz de mostrarnos una animación de la trayectoria dentro de planimetría (figura 4).

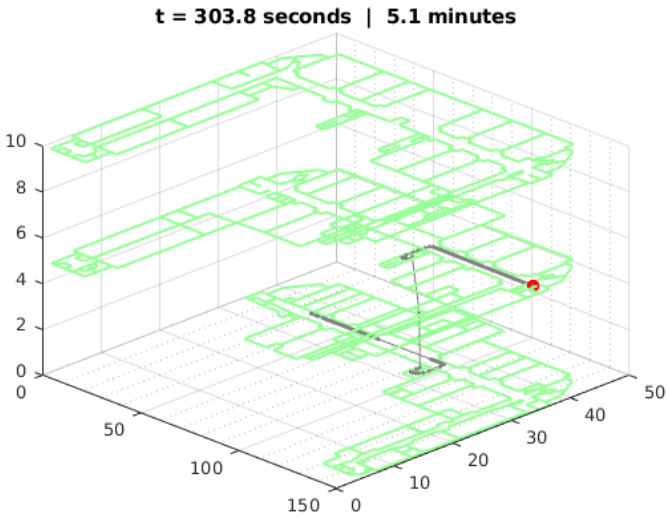

Figura 4. Captura de una animación de una trayectoria
La animación muestra como es la trayectoria dentro de la planimetría. El punto rojo muestra la possición de objetivo en cada instante.

III-C. Simulación de señales

En este módulo se podrán simular distintos tipos de señales a partir de la trayectoria generada por el módulo anterior. El objetivo es generar los datos que los algoritmo de posicionamiento 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 han definido dos tipo de señales, unas que dependen de la posición de balizas y otras que solo dependen de la trayectoria; eston son objetos *BeaconBased* y *BeaconFree* respectivamente (figura 6). Es clasificación es debido a que los parámetros de entrada para la generación de estas señales son distintos.

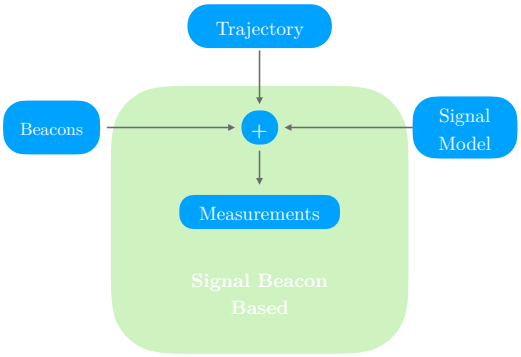

Figura 5. Representación de la construcción de las señales.
Se muestra un esquema de la construcción de las señales. Estas necesitan la trayectoria y un modelo de simulación. En el caso de la señales *BeaconBased*, además necestian los objetos *beacons* de definidos en el modulo de planimetría.

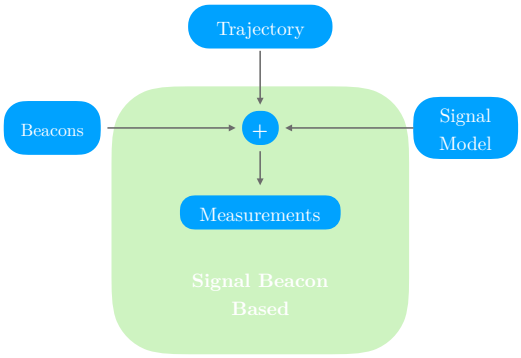

Figura 6. Representación de la construcción de las señales.
Se muestra un esquema de la construcción de las señales. Estas necesitan la trayectoria y un modelo de simulación. En el caso de la señales *BeaconBased*, además necestian los objetos *beacons* de definidos en el modulo de planimetría.

La simulación de señales se realiza a partir del objeto de la trayectoria. La trayectoria más un modelo de generación de señales son lo mínimo necesario para crear los objetos señales. En navindoor se han definido dos tipo de señales, unas que dependen de la posición de balizas y otras que solo dependen de la trayectoria; eston son objetos *BeaconBased* y *BeaconFree* respectivamente (figura 6).

BeaconBased	**BeaconFree**
Received Signal Strength (RSS)	Barómetro
Time of flight (ToF)	Acelerómetro
Angle of Arrival (AoA)	Magnetómetro

Cuadro I
TIPO DE MODELOS DE SIMULACIÓN DE SEÑALES

Al igual que en los modelos de simulación de trayectorias. Los modelos de generación de señales no son más que parámetros de entrada del constructor de las señales. Dentro de los tipo de modelos que pueden implementarse en navindoor se pueden ver en la tabla I. Se ha implementado un modelo para cada tipo de señales.

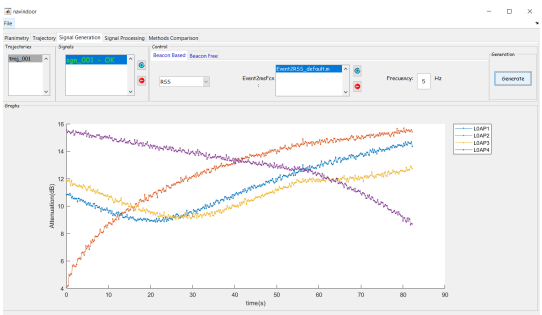

Figura 7. Interfaz gráfica para el módulo de las señales
En esta captura se puede ver la representación de una señales RSS, en una planimetría con cuarto *beacons*. La gráfica la evolución en el tiempo de la atenuación de cada uno de los *beacons*

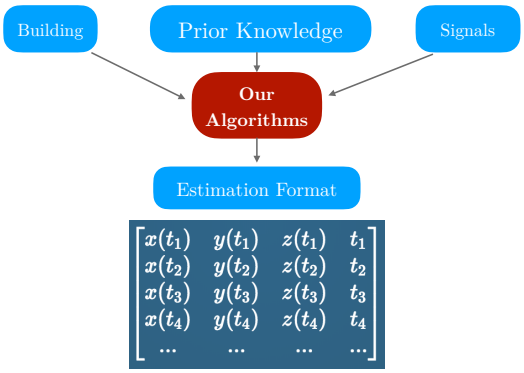

Figura 8. Esquema del formato de entrada y de salida
Los algoritmos en navindoor son funciones de MATLAB que deben tener las variables de entrada mostradas en el esquema. Los elementos como building o Signals pueden ser procesados con herramientas proporcionadas por el framework. Además tambien se puede explorar las propiedades de cada elemento para procesarlo.

III-D. Procesamiento

En esta sección se encuentran los algoritmos capaces de generar estimaciones de la trayectoria a partir de las señales generadas. Por defecto, se encuentra implementado el filtro de Kalman extendido (EKF) [5]. El cual da buenos resultados para simulaciones relativamente simples. De la misma forma que en las anteriores secciones los algoritmos sean parámetros de entrada del *framework*, por lo que se ha diseñado para que sea fácilmente ampliable. Para ello, es necesario que los algoritmos desarrollados contengan una estructura predefinida. Este formato se muestra en la figura 8.

Los algoritmos deberán proporcionar una matriz de estimación que luego será transformada a un objeto *traj*. De esta forma podemos reutilizar todos los metodos creados para las trayectorias, también para la estimaciones.

III-E. Comparación

Por ultimo, podemos comparar los resultados de distintas ejecuciones. La diferencia de las ejecuciones puede estar el algoritmo utilizado para estimar, sin embargo también puede estar en los modelos de trayectoria, en la simulación de señales, o simplemente la frecuencia de muestreo de alguna señal. Debido a que al final del ciclo, navindoor crea trayectorias a partir de las estimaciones, las comparaciones se puede realizar entre dos estimaciones o entre la estimación y la trayectoria real. Un ejemplo de uso pordemos ve en la figura (figura 9).

IV. CASO DE USO

V. CONCLUSIÓN

Se ha mostrado como los aspectos básicos de navindoor. Este software ha sido escrito para ser escalable, donde los propios usuarios puedan contribuir con nuevos modelos de simulación y de procesamiento. Aunque en este momento los modelos implentados son simples, su estructura modular es capaz de integrar modelos complejos. Además gracias a

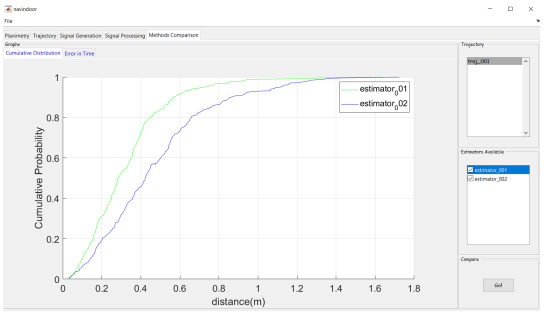

Figura 9. Interfaz gráfica para la comparación de estimaciones. En este caso podemos ver el efecto de la frecuencia de muestreo en nuestras estimaciones. La estimación azul se ha realizado con una señal RSS de frecuencia 1Hz, mientras que la ver se ha realizado con una frecuencia de 5Hz.

las APis de MATLAB para otros lenguajes de programación, es posible la integración de otros simuladores en distintos lenguajes, con una simple adaptación. Simluadores como los presentados en la sección II.

Es importante notar que la clase *building* es una abstracción de la planimetría de un edificio, por lo que la traslación a otros formatos como *XML* o *JSON* es posible, y se están explorando por parte del equipo de desarrollo. Este desarrollo permitirá comunicaciones con plataformas como *Open Street Maps*.

Navindoor es un software desarrollado en MATLAB, que mejora el desarrollo de algoritmos de posicionamiento. Esta constantemente en desarrollo y continuará agregando funcionalidades.

REFERENCIAS

[1] 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.

[2] 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.

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

[4] 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;.

[5] Mycielski J. Quantifier-free versions of first order logic and their psychological significance. Journal of Philosophical Logic. 1992;21(2):125–147.