Navindoor: Software de simulación para algoritmos localización

***Resumen*—En este artículo se presentará la herramienta de simulación, Navindoor. Esta es una librería para MATLAB que nos provee de las herramientas necesarias para todo el proceso de sistemas de localización.**

***Index Terms*—localización en interiores, procesamiento de señales**

I. Introdución

Las metodologías utilizadas para los sistemas de localización son tan variadas como la diversidad de sensores existentes. Sin embargo, aunque los algoritmos sean diversos, existe una serie de pasos comunes que se deben realizar antes que desarrollar un algoritmo de posicionamiento. Estas son, la toma de medidas experimentales de señales, así como medidas de la propia trayectoria. Esta recolección de datos debe ser variada, ya que una colección en pocas casuísticas puede agregar sesgos no deseados a nuestros algoritmos. Es por ello que lo ideal es realizar pruebas en varios entornos con distintas trayectorias. Sin embargo, esto hace que el tiempo y el coste del desarrollo aumente.

Es por ello que la simulación de señales es una tarea muy interesante. Buenos modelos de simulación pueden evitarnos pruebas experimentales antes de tener confianza completa en nuestro algortimo. Sin embargo, por ahora no existe un estandar de software donde poder desarrollar algortimos de distintas características. Los pocos simuladores que existen con este fin, suelen desarrollar en detalle alguna tecnología en concreto. Esto hace que la fusión de algoritmos o de modelos de simulación, requiera un prepoceso y el conocimiento de varios frameworks.

En este artículo se propone el software de simulación, Navindoor[1]. Este es un software para MATLAB, en dodne el usuario puede crear diseñar el escenario del movimiento, simular una trayectoria, geenerar señales sintéticas, además de procesar las señales generadas para crear una estimación, con ayuda de algoritmos implementados en el framework. Navindoor contiene una interfaz gráfica (GUI), que nos ayuda a generar lo antes mencionado. Sin embargo, Navindoor no solo es GUI, si no también una interfaz de lineas de comando (CLI). Esto le da la facilidad de automatizar algunas tareas. Pudiera parecer que Navindoor tambien es una estructura rígida, en donde los modelos de simulación estan anclados al framework, sin embargo, este se ha desarrollado de forma que la implementación de nuevos modelos es inmediata. Utilizando los objetos *function_handle* de MATLAB, podemos dar funciones como parámetros de entrada a los objetos de Navindoor. De esta manera desarrollar un modelo de simulación se simplifica a crear una funcion de MATLAB, con los parámetros de entrada/salida correctos.

El resto del documento se estrutura 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 los mostrado y de las siguientes desarrollos.

II. Estado del arte

En la comunidad del posicionamiento se esta buscando una solución para estandarizar los desarrollos. Es por ello que se han realizado trabajos relacionados. En este apartado mencionaremos algunos trabajos y 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 ambiente interior artificial, 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. SMILe, nos provee herramientas necesarias para la comparación de algoritmos de estimación de la posición a partir de señales ToF, sin embargo por ahora otras tecnologías no están implementadas.

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.

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 mantener la

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

posición, la orientación y el conocimiento de la situación. Los ejemplos de referencia proporcionan un punto de partida para implementar componentes de sistemas navegación.

Con este toolbox se puede importar y definir escenarios y trayectorias, transmitir señales y generar datos sintéticos para sensores activos y pasivos, 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. Arquitectura de Software

Navindoor contiene una CLI y una GUI, estas son dos versioones del mismo framework. Esto se ha realizado ya que la

III-A. Creación de Planimetría

Dentro del módulo de planimetría tenemos las herramientas necesarias para definir el escenario donde transcurrirán nuestras trayectorias. En primera instancia las trayectorias se definen a lo largo de un edificio, es por ello que las siguientes herramientas tiene el objetivo de caracterizarlo.

Para la definición del escenario se ha definido clases en MATLAB que representarán los distintos objetos. La clase más grande que contiene a los demás elementos, es la clase *building* (figura **??**). Estos objetos pueden contener objetos *level* que representan las distintas plantas. A su vez, los objetos *level*, contienen objetos type *walls*, *doors*, *beacons*, *stairs* y *elevators*, de esta forma cada planta queda completamente definida. Estos objetos nos ayuda en la construcción de la trayectoria, ya que nos da las restricciones que debe seguir, además de dar información al modelo de simulación de trayectoria. También estan presentes en la generación de las señales, así como en el propio procesamiento de las señales.

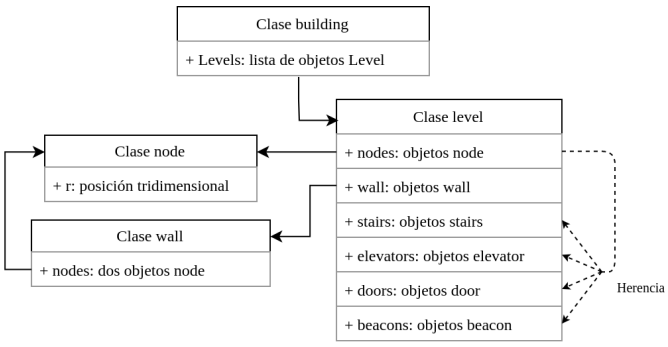

Figura 1. Esquema de la clase *building*

Debido a que el proceso de creación de la planimetría puede ser tedioso, navindoor contiene una GUI capaz de generar un objeto *building* (figura 2). La interfaz nos permite crear los objectos antes mencionados con unos simples *clicks*, además de permitir cargar una imagen de los planos reales con la que ayudarnos a construir la planimetría.

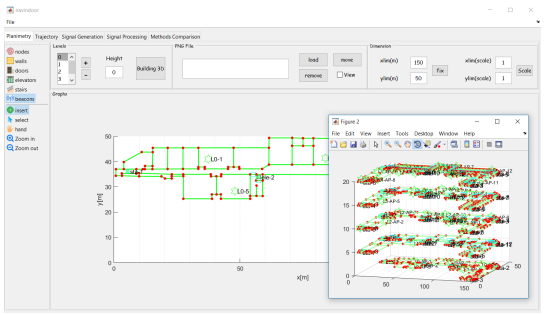

Figura 2. Interfaz gráfica para el diseño de la planimetría *building*. En la imagen se muestra la interfaz con un edificio de cuatro plantas. La figura externa es una representación tridimensional del edificio.

Ademaś, cabe mencionar que la GUI contiene herramientas necesarias para la visualización en 3D, la modificación y eliminación de los elementos. De esta forma, la construcción de la planimetría es 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.

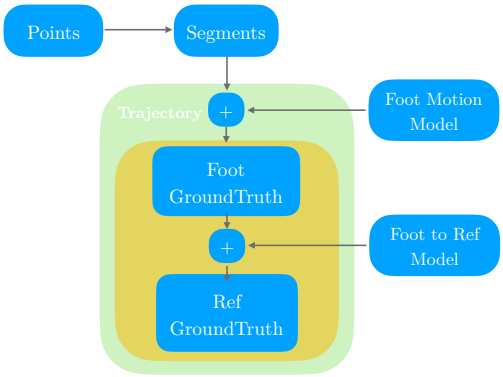

Figura 3. Proceso de construcción de una trayectoria
En este esquema se puede ver como el constructor de la trayectoria, genera los dos *GroundTruths*, a partir de los modelos de simulación seleccionados.

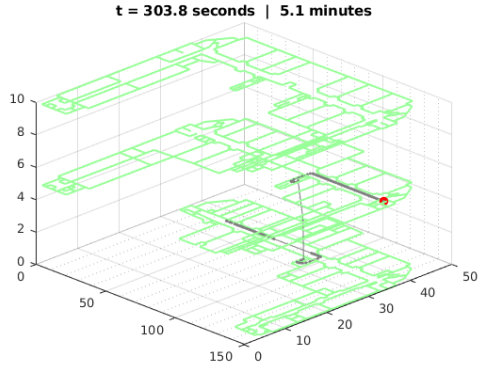

Figura 4. Captura de una animación de una trayectoria
La animación muestra como es la trayectoria dentro de la planimetría. El punto rojo muestra la possición de objetivo en cada instante.

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).

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

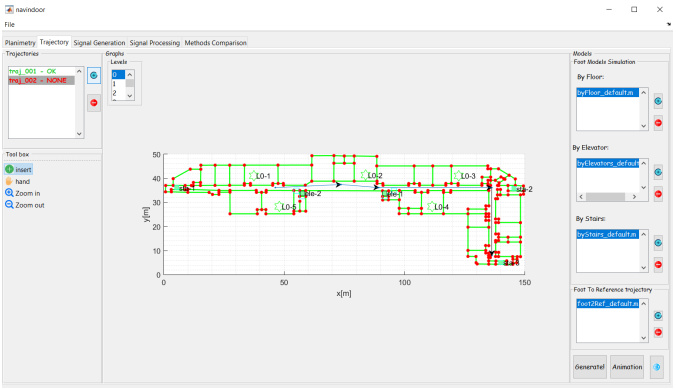

Figura 5. Interfaz gráfica para el diseño de las trayectorias
La derecha de la imagen se encuentra los modelos de simulación. Estos son funciones de matlab, por lo que el usuario puede agregar sus propias funciones con el formato adecuado y continuar en navindoor.

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 7). Es clasificación es debido a que los parámetros de entrada para la generación de estas señales son distintos.

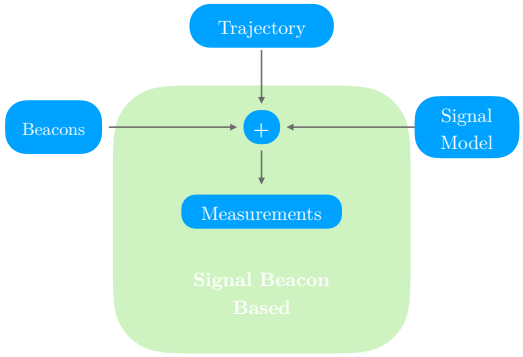

Figura 6. Representación de la construcción de las señales.
Se muestra un esquema de la construcción de las señales. Estas necesitan la trayectoria y un modelo de simulación. En el caso de la señales *BeaconBased*, además necestian los objetos *beacons* de definidos en el modulo de planimetría.

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 7).

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

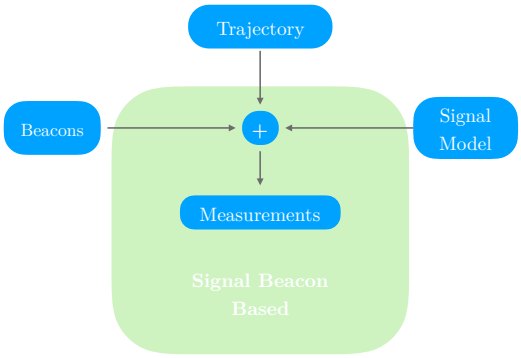

Figura 7. Representación de la construcción de las señales.
Se muestra un esquema de la construcción de las señales. Estas necesitan la trayectoria y un modelo de simulación. En el caso de la señales *BeaconBased*, además necestian los objetos *beacons* de definidos en el modulo de planimetría.

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

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.

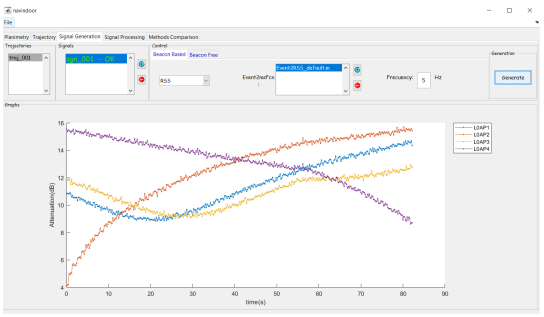

Figura 8. Interfaz gráfica para el módulo de las señales
En esta captura se puede ver la representación de una señales RSS, en una planimetría con cuarto *beacons*. La gráfica la evolución en el tiempo de la atenuación de cada uno de los *beacons*

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 9.

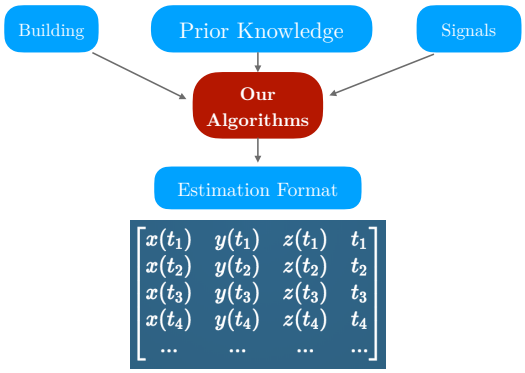

Figura 9. Esquema del formato de entrada y de salida
Los algoritmos en navindoor son funciones de MATLAB que deben tener las variables de entrada mostradas en el esquema. Los elementos como building o Signals pueden ser procesados con herramientas proporcionadas por el framework. Además tambien se puede explorar las propiedades de cada elemento para procesarlo.

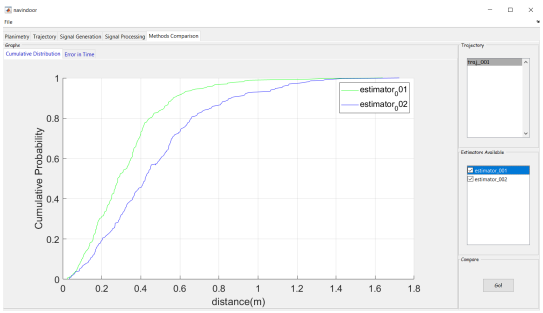

Figura 10. Interfaz gráfica para la comparación de estimaciones.
En este caso podemos ver el efecto de la frecuencia de muestreo en nuestras estimaciones. La estimación azul se ha realizado con una señal RSS de frecuencia 1Hz, mientras que la ver se ha realizado con una frecuencia de 5Hz.

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 10).

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 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.