Contribución al desarrollo de una aplicación móvil de posicionamiento en interiores basado en luz (VLP) usando los sensores inerciales del dispositivo

Aitor Alcázar Fernández

Smart Cities
Máster Universitario en Ingeniería de Telecomunicación

Antoni Pérez Navarro (UOC)
Álvaro De La Llana Calvo (UAH)
(Carlos Monzo Sánchez)

27 de diciembre de 2021

FICHA DEL TRABAJO FINAL

Título:	Contribución al desarrollo de una aplicación móvil de posicionamiento en interiores basado en luz (VLP) usando los sensores inerciales del dispositivo
Nombre autor/a:	Aitor Alcázar Fernández
Nombre PDC:	Antoni Pérez Navarro (UOC) Álvaro De La Llana Calvo (UAH)
Nombre PRA:	Carlos Monzo Sánchez
Fecha de entrega:	27 de diciembre de 2021
Titulación:	Máster Universitario en Ingeniería de Telecomunicación
Área:	Smart Cities
Idioma:	Castellano
Núm. de créditos:	12
Palabras clave:	VLP, IMU, AoA, PDR, dispositivo móvil, Android

Resumen

Los sistemas de posicionamiento resultan una herramienta fundamental en el ámbito industrial y social. Desde actividades de monitorización en tiempo real hasta la navegación de los automóviles, es imprescindible contar con sistemas que garanticen una alta precisión en el posicionamiento.

El posicionamiento basado en luz visible (VLP) ofrece una solución *indoor* de este ámbito tecnológico. Se utiliza el espectro de la luz visible para posicionar objetos en el espacio, de modo que se utilizan los focos de luz para iluminar el entorno y para integrar sistemas inteligentes de posicionamiento.

El objeto de este trabajo es contribuir al desarrollo de una aplicación móvil de guiado y posicionamiento en un museo. Esta aplicación utiliza los sensores de la cámara del dispositivo móvil para detectar focos de luz pertenecientes a una red de focos inteligentes.

Ante la existencia de diversas fuentes de error, este proyecto se centra en la resolución de dos de ellas mediante sendos algoritmos que aprovechan la unidad de medida inercial de los dispositivos.

El primer algoritmo permite corregir la posición captada de un foco, en función de los ángulos de posición del dispositivo móvil. El segundo, utiliza un sistema de navegación inercial para predecir el rumbo del usuario en aquellos momentos que no se capta un foco.

Los resultados obtenidos con dichos algoritmos muestran que se incrementa la precisión en el posicionamiento del usuario, así como en la naturalidad de su movimiento. Además, estos algoritmos pueden ser implementados en otras soluciones existentes de VLP.

Abstract

Positioning systems are an essential tool in the industrial and social spheres. From real-time monitoring activities to automotive navigation, systems that ensure high positioning accuracy are essential.

Visible Light Positioning (VLP) offers an indoor solution in this technological field. The visible light spectrum is used to position objects in space, so that light sources are used to illuminate the environment and to integrate intelligent positioning systems.

The purpose of this work is to contribute to the development of a mobile application for guidance and positioning in a museum. This application uses the camera sensors of the mobile device to detect light sources belonging to a network of intelligent light sources.

Given the existence of several sources of error, this project focuses on the resolution of two of them by means of algorithms that take advantage of the inertial measurement unit of the devices.

The first algorithm allows correcting the captured position of a spotlight, depending on the position angles of the mobile device. The second uses an inertial navigation system to predict the user's course when a spotlight is not captured.

The results obtained with these algorithms show that the accuracy of the user's positioning is increased, as well as the naturalness of the user's movement. Furthermore, these algorithms can be implemented in other existing VLP solutions.

Índice general

Índice de figuras

Índice de tablas

Índice de códigos

Capítulo 1

Introducción

1.1. Contexto y justificación del trabajo

Los sistemas de posicionamiento en interiores o IPS (del inglés *Indoor Positioning Systems*) cuentan con diferentes desarrollos e implementaciones con el fin de localizar a los usuarios en grandes entornos interiores, como pueden ser: actividades industriales, centros de ocio o aplicaciones logísticas para fábricas inteligentes, entre otros.

Los sistemas IPS ya están implementados en las actividades cotidianas, pero mejorar su funcionamiento puede ampliar el espectro de posibles aplicaciones para actividades actuales y futuras. Este aspecto puede contribuir al área tecnológica de las ciudades inteligentes o *Smart Cities*.

Estos sistemas no cuentan con ninguna tecnología estándar o más extendida que otras, como sucede con los sistemas globales de posicionamiento o GNSS (del inglés *Global Navigation Satellite System*). De hecho, las diferentes técnicas utilizadas son complementarias y cooperan entre sí, con limitaciones intrínsecas en el propio entorno. En ese sentido, la coexistencia de sistemas que trabajan en bandas similares, la infraestructura del interior o incluso los materiales, son ejemplos de limitaciones.

Dadas las aplicaciones de estos sistemas, es clave consumir poca energía, disponer de buena cobertura, tener una buena precisión, un coste asumible y disponer de privacidad. Este conjunto de puntos clave se cubrirán e implementarán a lo largo de este Trabajo Final de Máster (a partir de ahora TFM).

El trabajo en cuestión contribuye a una línea de investigación del Grupo de Ingeniería Electrónica Aplicada a Espacios Inteligentes y Transporte o GEINTRA oficialmente reconocido por la Universidad de Alcalá[1]. En concreto, se aportarán mejoras en el proyecto "Sistema de Guiado y Localización en Interiores usando la Iluminación de los Edificios" o GUIA, que cuenta con la financiación de la Junta de Castilla La-Mancha. El propósito de este proyecto es

[1] GEINTRA (referencia CCTT2006/R37): http://www.geintra-uah.org/

implementar un IPS basado en luz visible o VLP (del inglés *Visible Light Positioning*) que sea capaz de localizar a usuarios en el Museo de Guadalajara, situado en el Palacio del Infantado de la ciudad de Guadalajara, haciendo uso de un terminal móvil como unidad de adquisición y procesado y una red de luminarias que hagan las veces de balizas de posicionamiento inteligentes y de focos de luz para el edificio.

En esta fase, la aplicación móvil debe posicionar al usuario mediante la detección de solo una luminaria a través del sensor de la cámara. Para ello hay que poder detectar e identificar cada una de las distintas luminarias inteligentes. Conocida *a priori* la posición de las distintas luminarias, se ubica al usuario justo debajo de la luminaria detectada. Con esta propuesta de posicionamiento se comete un error que dependerá del campo de cobertura de la cámara. Además, solo se puede posicionar al usuario si se detecta alguna luminaria.

En este trabajo se proponen mejoras en la precisión de localización del usuario ante distintas situaciones adversas:

- Mejorar la precisión de la localización con la información del ángulo de llegada o AoA (del inglés *Angle of Arrival*). A partir de los datos obtenidos del sistema de lentes y posición en píxeles donde se ha detectado la luminaria en el sensor, se tienen en cuenta los problemas intrínsecos en la posición de la mano y del usuario con respecto de las luminarias.
- Solucionar las pérdidas en la localización por la ausencia de luminarias captadas cuando el usuario se desplaza. Aprovechar el hardware del dispositivo móvil para este propósito.

1.2. Objetivos del trabajo

En el desarrollo de este TFM se persiguen dos objetivos principales:

1. **Utilizar los sensores inerciales del dispositivo para implementar un algoritmo de corrección de la posición de objetos detectados en tiempo real.**

   En este apartado se desarrolla un algoritmo que aproveche los sensores inerciales de la unidad de medida inercial o IMU (del inglés *Inertial Measurement Unit*) de los dispositivos móviles para corregir la posición de objetos detectados en función de los ángulos de posición del dispositivo. Dado que al detectar una única luminaria no se pueden aplicar directamente técnicas de triangulación para posicionar el sensor. Por tanto, es necesario conocer la información de la posición de la luminaria en el sensor (píxeles donde se detecta la luminaria), el sistema de lentes del dispositivo móvil (distancia focal, centro óptico) y tamaño del sensor. Adicionalmente, es necesario conocer otros datos como la estimación de la distancia entre el sensor y la luminaria o la orientación del propio sensor. Esta última información es la que se puede obtener a partir de los datos de la IMU del dispositivo móvil.

2. **Utilizar los sensores inerciales del dispositivo para mejorar las estimaciones de la posición y la trayectoria de un usuario en tiempo real.**

   En este caso, se utiliza la IMU para estimar constantemente la posición del usuario y poder así contribuir a la mejora de los errores en el posicionamiento determinado por las luminarias. Partiendo de una posición conocida, cuando el usuario se desplaza cabe esperar dos estados: cuando se capta un foco y cuando no se capta un foco. En el caso de captar un foco, se utilizan las estimaciones habituales proporcionadas por la captación de luz (las establecidas en la aplicación del proyecto de GUIA) y, adicionalmente, se utilizan las estimaciones calculadas con la IMU.

   Por otro lado, cuando no se capta un foco, se utiliza únicamente la IMU de manera que se contribuye al estado general del posicionamiento con un error mayor que en el caso anterior. Sirve para dar continuidad al movimiento, tal y como sucede en la realidad. Esto otorga naturalidad a la trayectoria a representar y evita pérdidas en el posicionamiento.

   Este fenómeno de naturalidad se estudia mediante un método matemático de análisis conocido como navegación por estima cuyo fin es ubicar y posicionar a un objetivo, en función de su situación inicial, orientación y velocidad. La navegación por estima peatonal se refiere, por tanto, al análisis del caminar de los individuos. En este segundo objetivo, se implementará un algoritmo basado en PDR haciendo uso de los parámetros obtenidos de la IMU.

1.3. Enfoque y método

En la línea de lo expuesto en los apartados anteriores, el desarrollo de este TFM supone una contribución al diseño de una aplicación para mejorar las características del producto final. Por ello, la implementación de los algoritmos que permiten alcanzar los objetivos establecidos en el alcance del trabajo supone la implementación de un nuevo producto o, en otras palabras, la contribución a las mejoras necesarias para obtener el producto final.

El método de trabajo seguido se compone de tres fases: estudio, implementación y conclusión.

1. Estudio: se refiere al estado del arte y a la contextualización del problema.
2. Implementación: se refiere al desarrollo, programación, montaje y pruebas con respecto del problema a solventar.
3. Conclusión: se refiere al análisis de los resultados obtenidos y del impacto en el problema.

Este método de trabajo se corresponde con un modelo teórico-experimental, utilizado habitualmente en los proyectos de investigación, como es el caso, o en los proyectos industriales.

Como planteamiento técnico del proyecto en términos experimentales, se debe tener en cuenta que la aplicación está basada en la plataforma móvil Android, de manera que todas las implementaciones deben ser compatibles con dicha plataforma o fácilmente adaptables.

En este caso, se opta por trabajar con el entorno de desarrollo integrado o IDE (del inglés *Integrated Development Environment*) Android Studio. Este entorno ofrece la posibilidad de codificar aplicaciones para el sistema operativo Android de forma nativa, mediante el uso de su propio kit de desarrollo nativo o NDK (del inglés *Native Development Kit*). Se puede implementar la lógica, la interfaz gráfica de usuario o GUI (del inglés *Graphical User Interface*) y las directrices para interactuar con el hardware del sistema. Los lenguajes de programación que se utilizan para el desarrollo pueden ser Java o Kotlin indistintamente, mientras que la interfaz se diseña con el lenguaje de marcado XML (*Extensible Markup Language*).

Otro aspecto a considerar a la hora de elegir Android Studio como herramienta de trabajo es que su comunidad es muy activa, dado que las aplicaciones móviles son una de las principales mercados de desarrollo en el ámbito software. Además, la posibilidad de desarrollar en Kotlin o en Java simultáneamente en el proyecto permite utilizar bibliotecas de terceros que incrementan la versatilidad, sencillez y elegancia en la programación de la aplicación.

Finalmente, se destaca la herramienta de diseño que integra el IDE de Android: el editor de diseño. En esta herramienta se pueden compilar los diseños rápidamente arrastrando elementos de la GUI al editor de diseño visual, en lugar de escribir el fichero XML de diseño de forma manual. De esta manera, se agiliza el proceso de diseño, el trabajo es eficiente y resulta puramente creativo.

1.4. Planificación del trabajo

Este apartado establece una organización y estructura del TFM, con el fin de planificar las tareas que se desarrollan durante el cuatrimestre, analizar los posibles riesgos de su desarrollo y diseñar un plan de contingencia ante estos.

Planificación de actividades

Se definen tres tipos de actividad: PEC, tarea e hito. Las PEC son las Pruebas de Evaluación Continua del Máster, cuya finalidad es establecer los tiempos de entrega a lo largo de la asignatura, de modo que se tiene una duración de las actividades determinada. Las tareas son, por tanto, el conjunto de actividades que se realizan dentro de una PEC. Los hitos son sucesos excepcionales de gran importancia para la materia.

La tabla 1.1 recoge la organización de las actividades principales del Trabajo. Se presenta un identificador de actividades, la fecha de inicio, la fecha de fin y la duración en días.

La codificación del identificador y su correspondiente color en el Diagrama de Gantt es:

- HT**X**: Actividad **X** del tipo Hito codificados en verde.
- PEC**X**: Prueba de Evaluación Continua **X** codificados en azul oscuro.
- PEC**X.Y**: Tarea número **Y** de la PEC**X** codificados en azul claro.

Actividad	ID	Fecha de inicio	Fecha de fin	Duración
Comienzo de la asignatura	HT1	15-sep.-21	16-sep.-21	1
Objetivos del TFM y organización de tareas	PEC1	15-sep.-21	26-sep.-21	11
Resumen y propuesta del título	PEC1.1	15-sep.-21	17-sep.-21	2
Motivación, objetivos y enfoque	PEC1.2	17-sep.-21	22-sep.-21	5
Planificación del proyecto	PEC1.3	17-sep.-21	25-sep.-21	8
Índice preliminar de la memoria	PEC1.4	22-sep.-21	26-sep.-21	4
Estado del arte	PEC2	27-sep.-21	10-oct.-21	13
Contexto y búsqueda de otros trabajos	PEC2.1	27-sep.-21	30-sep.-21	3
Estudio del marco teórico	PEC2.2	29-sep.-21	5-oct.-21	6
Enfoque y descripción teórica	PEC2.3	5-oct.-21	10-oct.-21	5
Desarrollo y resultados obtenidos	PEC3	11-oct.-21	3-dic.-21	53
AoA: Obtención de los parámetros del dispositivo	PEC3.1	11-oct.-21	16-oct.-21	5
AoA: Modelado matemático basado en pinhole	PEC3.2	17-oct.-21	24-oct.-21	7
AoA: Implementación del algoritmo en Kotlin	PEC3.3	25-oct.-21	4-nov.-21	10
AoA: Análisis de comportamiento con y sin detección	PEC3.4	5-nov.-21	10-nov.-21	5
PDR: Detección de pasos, longitud de pasos y rumbo	PEC3.5	11-nov.-21	21-nov.-21	10
PDR: Análisis del comportamiento entre focos	PEC3.6	22-nov.-21	3-dic.-21	11
Estudio de resultados obtenidos con ambos algoritmos	PEC3.7	25-nov.-21	3-dic.-21	8
Elaboración final de la memoria	PEC4	4-dic.-21	27-dic.-21	23
Conclusiones	PEC4.1	4-dic.-21	7-dic.-21	3
Líneas futuras del proyecto y repercusión	PEC4.2	4-dic.-21	6-dic.-21	2
Revisión bibliografía	PEC4.3	7-dic.-21	12-dic.-21	5
Maquetado final	PEC4.4	13-dic.-21	27-dic.-21	14
Elaboración de la presentación	PEC5	28-dic.-21	5-ene.-22	8
Maquetado diapositivas	PEC5.1	28-dic.-21	30-dic.-21	2
Grabación y montaje de la lectura	PEC5.2	2-ene.-22	5-ene.-22	3
Defensa del TFM	HT2	10-ene.-22	11-ene.-22	1

Tabla 1.1: Organización de actividades del TFM

Una vez identificadas todas las actividades y sus respectivos tiempos de ejecución, se describe cada una de ellas:

- HT1. Comienzo de la asignatura: hito correspondiente al día de inicio de la asignatura de TFM. También supone el kick-off del proyecto.
- PEC1. Objetivos del TFM y organización de las tareas: versión preliminar de la memoria del proyecto donde se definen los objetivos, el plan de trabajo y los resultados previstos del proyecto.

- PEC1.1. Resumen y propuesta del título: elaboración de un breve resumen (sujeto a cambios) y propuesta del título a los tutores.
- PEC1.2. Motivación, objetivos y enfoque: redacción de la razón de ser del TFM, el enfoque y los objetivos que se persiguen durante su desarrollo.
- PEC1.3. Planificación del proyecto: tabla y Diagrama de Gantt con las actividades propuestas, donde se especifiquen los tiempos de ejecución y la descripción de las tareas.
- PEC1.4. Índice preliminar de la memoria: implementación del índice preliminar de la memoria del proyecto, como tabla de contenido con las páginas actualizadas.
- PEC2. Estado del arte: redacción del estado de los proyectos académicos e industriales que tienen relación con el TFM.
- PEC2.1. Contexto y búsqueda de otros trabajos: contextualización del trabajo con respecto del proyecto y exposición de trabajos relacionados con la materia.
- PEC2.2. Estudio del marco teórico: análisis de toda la teoría necesaria para implementar el proyecto, búsqueda y uso de la bibliografía recomendada.
- PEC2.3. Enfoque y descripción teórica: descripción de todos los elementos teóricos necesarios en el contexto del proyecto.
- PEC3. Desarrollo y resultados obtenidos: fase del proyecto en la que se trabaja con el fin de obtener el producto del proyecto.
- PEC3.1. AoA: Obtención de los parámetros del dispositivo: estudio de los parámetros físicos del dispositivo de pruebas.
- PEC3.2. AoA: Modelado matemático basado en pinhole: desarrollo matemático basado en *pinhole* para obtener la posición del usuario.
- PEC3.3. AoA: Implementación del algoritmo en Kotlin: implementación de un algoritmo de corrección de la posición detectada, correspondiente con la resolución del objetivo 1.
- PEC3.4. AoA: Análisis de comportamiento con y sin detección: análisis del algoritmo de corrección para obtener resultados y conclusiones prácticas.
- PEC3.5. PDR: Detección de pasos, longitud de pasos y rumbo: análisis de los parámetros generales de PDR para su aplicación al proyecto, correspondiente a la resolución del 2.
- PEC3.6. PDR: Análisis del comportamiento entre focos: pruebas de los algoritmos de corrección en entornos reales.
- PEC3.7. Estudio de resultados obtenidos con ambos algoritmos: análisis de las pruebas en entornos reales.

- PEC4. Elaboración final de la memoria: elaboración final de la memoria del TFM, con el detalle de las conclusiones, el análisis de las líneas futuras y el maquetado final.
- PEC4.1. Conclusiones: redacción de la conclusión, con un manifiesto de carácter científico basado en los resultados obtenidos.
- PEC4.2. Líneas futuras del proyecto y repercusión: propuesta de líneas futuras de investigación en el ámbito del proyecto o extrapolable. Repercusión: posibles publicaciones y artículos de carácter científico.
- PEC4.3. Revisión bibliografía: revisión de la bibliografía con el fin de referenciar el apoyo teórico-práctico en otros trabajos.
- PEC4.4. Maquetado final: revisión del maquetado del documento, referencias cruzadas y diseño final.
- PEC5. Elaboración de la presentación: entrega a través del campus virtual de la presentación del TFM y posterior defensa.
- PEC5.1. Maquetado diapositivas: diseño de la presentación en diapositivas del TFM.
- PEC5.2. Grabación y montaje de la lectura: grabación y montaje de la defensa asíncrona del TFM.
- HT2. Defensa del TFM: hito correspondiente a la entrega del TFM y a la defensa. Se contempla tanto la defensa asíncrona como la fase de preguntas y respuestas.

Diagrama de Gantt

Se establece también un cronograma de referencia del tipo Diagrama de Gantt como se observa en la Figura 1.1, que presenta de manera visual el flujo de desarrollo de las actividades, así como los instantes de tiempo en los que se realizan tareas en paralelo. Se utiliza el código de colores para conocer qué tipo de actividad se desempeña.

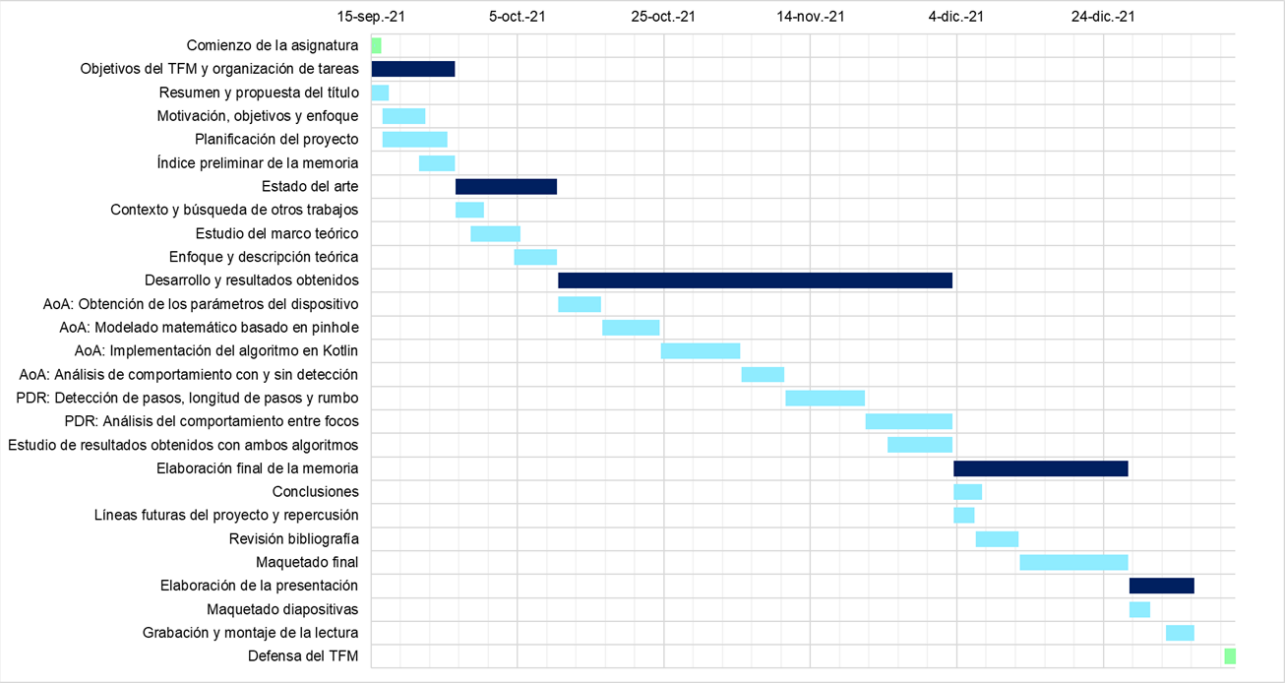

Figura 1.1: Diagrama de Gantt de la planificación del TFM

Análisis de riesgos y plan de contingencia

En el desarrollo de todo proyecto existe un conjunto de posibles imprevistos que pueden perturbar su transcurso y planificación en las fases previstas. Es misión del planificador contemplar las posibilidades y manifestarlas antes de la realización del trabajo, de modo que se pueda clasificar su impacto y priorizar el orden y modo de solucionarlo.

En este caso, los riesgos se presentan en la tabla según su impacto sobre el proyecto, así como posibles soluciones. El impacto se codifica en verde si es bajo, amarillo si es medio y rojo si es alto.

Riesgo	Solución
Incumplimiento de la fecha de entrega	Seguir la planificación del TFM
Solapamiento de tareas con otras asignaturas	Organización de las horas dedicadas y seguimiento del plan
Punto de bloqueo por falta de conocimiento	Realizar consultas a los tutores
Intercambio de correcciones lento	Realizar seguimiento bidireccional
Falta de tiempo por festivos, vacaciones y conferencias	Reorganización de las tareas y hacer un esfuerzo adicional
Acumulación del trabajo por numerosas revisiones	Atender a las indicaciones de los tutores
Nota insuficiente para cada entregable	Leer plan docente y rubrica de corrección

Tabla 1.2: Análisis de riesgos y plan de contingencia

Capítulo 2

Estado del arte

2.1. Trabajos relacionados

Los IPS tienen amplias aplicaciones en los sectores comercial, militar, minorista y de seguimiento de inventarios. Existen varios sistemas comerciales en el mercado, pero no hay estándares para un sistema IPS ya que, a diferencia de los GNSS, no siempre se cumplen los criterios establecidos en el estándar ISO/IEC 24730-1 sobre sistemas de posicionamiento en tiempo real o RTLS (del inglés *Real-Time Locating Systems*) [ISO/IEC, 2014]. Por tanto, cada instalación se adapta a las dimensiones espaciales, los materiales de construcción, las necesidades de precisión y las limitaciones presupuestarias de cada aplicación. Existe una gran variedad de técnicas y dispositivos para proporcionar posicionamiento en interiores, ya sea reutilizando servicios ya desplegados, como dispositivos móviles, redes wifi y Bluetooth; o instalaciones *ad hoc* cuyo funcionamiento depende de balizas o relés [Mautz, 2012]. En función de la aplicación final, se opta por seleccionar alguna de las técnicas con el fin de implementar una solución óptima. En el caso que ocupa el presente TFM, se busca profundizar en diseños que se especializan en el uso de la iluminación para determinar la posición de personas.

Un caso particular de este tipo de diseños es el posicionamiento basado en luz visible o VLP (del inglés *Visible Light Positioning*). Esta técnica consiste en utilizar y detectar fuentes de luz que pertenecen al espectro visible con el fin posicionar a un objetivo. En la literatura, en trabajos como los de [Do and Yoo, 2016] y [Luo et al., 2017] se estudia exhaustivamente el estado del arte de la tecnología VLC, que contempla entre sus aplicaciones el posicionamiento basado en luz visible (VLP).

En general, los casos de uso de esta tecnología se centran en ámbitos de robótica o industriales, de modo que sus resultados son altamente precisos (con errores de escasos milímetros). Trabajos como [De-La-Llana-Calvo et al., 2020] y [Rodríguez-Navarro et al., 2017], que estudian sistemas basados en VLP que utilizan dispositivos sensibles a la posición o PSD (del inglés *Position Sensitive Device*) para determinar la localización de objetos que los integran, son ejemplos de soluciones de alta precisión.

En el ámbito de posicionamiento de personas, los resultados obtenidos pueden ser menos precisos que en las aplicaciones de robótica. En ese sentido, hay que destacar el hecho de que la mayoría de estas soluciones requieren de elementos externos, como es el caso de los basados en PSD. Una alternativa es el uso de los dispositivos móviles, dado que la mayoría de los usuarios dispone de uno. Además, utilizar un *smartphone* permite que el aplicación sea confortable para el usuario final, ya que evita portar otros dispositivos adicionales.

El estudio realizado en el trabajo [Ashraf et al., 2020], se habla sobre la situación actual y los desafíos futuros en el posicionamiento en interiores basado en los sensores de los dispositivos móviles. De todas las fuentes de datos disponibles en estos sistemas (cámara, sensores inerciales, antenas), los trabajos que se relacionan con el proyecto son aquellos que utilizan la luz para posicionar al objetivo.

Bajo las premisas de posicionamiento basado en VLP y con dispositivos móviles, las soluciones existentes aprovechan el hardware del teléfono. Entre estos sensores se encuentran los sensores de luz ambiental y el sensor de la cámara.

- Las soluciones basadas en la captación de la luz ambiental proporcionan una posible implementación para el posicionamiento en interiores. En [Wang et al., 2019] se realiza un estudio de las implementaciones de detectores *indoor* con sensores ambientales. El principal problema de utilizar estos sensores es la tasa de refresco que pueden alcanzar, que habitualmente es menor a 3 kHz. Dado el caso de uso del proyecto, que requiere trabajar con señales de mayor frecuencia, esta solución no es válida.
- Las soluciones basadas en el sensor de la cámara, donde existen trabajos basados en identificación de objetos geométricos mediante inteligencia artificial [Jabborov and Cho, 2020] o basados en identificación de bombillas gracias al fenómeno de *rolling shutter*[1] [Li, 2016], entre otros.

El fenómeno de *rolling shutter* permite trabajar con frecuencias mayores que el sensor ambiental (del orden de 50 kHz), de modo que será una buena opción en el caso de necesitar identificar un mayor número de frecuencias. La limitación de diseño de esta técnica es la distancia a la que se ubica el foco emisor y, consecuentemente, su tamaño: a mayor distancia o menor tamaño del foco, más difícil resulta reconocer la frecuencia.

En el proyecto GUIA, se identifican sendas lámparas mediante la técnica de angulación, que están ubicadas en el techado del Museo de Guadalajara (desarrollo del proyecto independiente de este TFM). Este tipo de medida requiere de cuatro focos emisores, pero la solución de compromiso es desplegar el menor número de luminarias posibles. Para conseguirlo, se debe encontrar un método que permita disminuir los focos aprovechando otras características del dispositivo.

Del resto de hardware disponible en el terminal, resulta interesante el uso de los sensores inerciales que componen la IMU. Existen numerosos trabajos basados en navegación inercial como

[1] El *rolling shutter* es un método de captura de imagen realizado con un sensor CMOS, donde el obturador capta la escena a partir de un barrido

[Jimenez et al., 2009] o [Guo and Uradzinski, 2018], pero en la mayoría de los casos se requiere de una IMU externa (como se ha detallado anteriormente). El caso que ocupa requiere del uso exclusivo del *smartphone* y, para ello, trabajos como [Wang et al., 2018] y [Li and Ning, 2018] son de interés para la aplicación.

El trabajo de [Wang et al., 2018] estudia una ecuación general para implementar un algoritmo de navegación inercial para peatones que se independiente de la forma en la que se sostiene el dispositivo. Por su parte, en [Li and Ning, 2018] se implementa dicho algoritmo dependiente de la posición del usuario pero con la ventaja de calibrar el dispositivo gracias a utilizar mapas georreferenciados. Gracias a estos dos trabajos, se puede desarrollar un algoritmo que permita posicionar al usuario en los momentos en los que no se capta un foco.

En este trabajo final, se propone fusionar los métodos de posicionamiento basados en angulación, que se desarrollan en el proyecto GUIA, con la implementación de un algoritmo basado en PDR. El objetivo de esta fusión es aprovechar el hardware disponible en el dispositivo para disminuir el número de lámparas instaladas en el Museo de Guadalajara y posicionar al usuario en los momentos en los que no se captan focos de luz.

2.2. Marco teórico

Este apartado del estado del arte recoge los aspectos teóricos que se vinculan directamente con el desarrollo del proyecto. Es necesario conocer los conceptos, las definiciones y las matemáticas tras la implementación del sistema. Dado el alcance del TFM, cabe destacar que la profundidad de análisis de este apartado busca enmarcar de manera sucinta y clara la teoría relacionada con el trabajo.

En el apartado 2.2.1 se definen los conceptos de posicionamiento, cómo se mide la posición y las estrategias para aplicar dichas medidas. En el apartado 2.2.2 se describen los distintos tipos de sensores inerciales, así como la clasificación según el tipo de tecnología mediante la que están construidos. Por último, en el apartado 2.2.3 se realiza un análisis en profundidad de las posibles implementaciones del algoritmo de navegación por estima peatonal y los pasos a seguir para implementarlo.

2.2.1. Medidas y estrategias del posicionamiento

La geometría de posicionamiento hace referencia al uso de las matemáticas basadas en la observación de la naturaleza topológica. En general, se utiliza para poder estimar la posición de un punto objetivo dentro de un sistema de referencia determinado, normalmente fijo. La geometría del posicionamiento puede medirse utilizando diversas técnicas, que se aplicarán con una serie de estrategias del posicionamiento. Los siguientes subapartados describen algunas de las medidas y de las estrategias más extendidas en el ámbito de posicionamiento *indoor*.

Medidas de posicionamiento

Las medidas de posicionamiento son aquellos métodos que permiten posicionar a un objetivo en el entorno. Se utilizan métodos matemáticos o empíricos para estimar la posición del objetivo. Existen diversos tipos de medidas como las basadas en angulación, tiempos o potencias, pero en este trabajo únicamente se definen tres de ellas. Se puede ampliar la información de las medidas de posicionamiento en trabajos como [Salido-Monzú, 2015] o [Conesa et al., 2018].

- El ángulo de llegada o AoA (del inglés *Angle of Arrival*) se basa en la medición del ángulo de recepción para estimar la dirección de cierta señal entrante. Este tipo de técnicas pueden proporcionar mediciones muy precisas, aunque su implementación requiere habitualmente de implementaciones complejas. Como se ha detallado en los trabajos relacionados (apartado 2.1), el orden de precisión de estas medidas alcanza escasos milímetros.
- El tiempo de llegada/vuelo o ToA/ToF (del inglés *Time of Arrival/Flight*) y la diferencia de tiempo de llegada o TDoA (del inglés *Time Difference of Arrival*) son dos técnicas de medida que obtienen la información de la distancia o diferencias de distancia, respectivamente, de una señal que se propaga por un canal de espacio libre siguiendo un camino de línea de visión o LOS (del inglés *Line Of Sight*). La medida realizada sobre su tiempo de propagación será directa (ToA/ToF) o indirecta (TDoA).
- La intensidad de la señal recibida o RSS (del inglés *Received Signal Strength*) se basa en la obtención y comparación de la intensidad de sendas señales recibidas. Se estima la distancia en función de los diferentes valores obtenidos basándose en algún modelo de propagación de ondas o en función de un mapa de valores obtenido de forma empírica sobre dicho escenario. Suelen reutilizarse las redes de telecomunicación existentes y, por lo tanto, son sistemas de bajo coste.

Estrategias de posicionamiento

Las estrategias de posicionamiento son métodos que utilizan la información obtenida en las medidas de posicionamiento, de modo que se pueda estimar la posición de un punto objetivo dentro del sistema de referencia en el que se encuentra. En ese sentido se estudian tres tipos de estrategias: lateralización, angulación y navegación por estima.

- La estrategia de lateralización se refiere a la estimación de la posición del punto objetivo mediante medidas de distancia absolutas o diferenciales entre el objetivo y sendos puntos de referencia de posición conocida. Este método se puede aplicar a todo sistema que utilice medidas de posicionamiento de distancia, tal y como sucede en ToA/ToF y TDoA.

  En el caso de hablar de medidas de distancia absolutas, se obtiene la estimación de posición mediante la intersección de esferas. Si el sistema es tridimensional, se requiere un mínimo de tres esferas para estimar la posición. Esta técnica recibe el nombre de trilateración y se relaciona con la medida de ToA/ToF. En la Figura 2.1 (a) se muestra un ejemplo de

trilateración esférica. Se tienen tres puntos de referencia (en verde) desde los que se traza una esfera; por su parte, el objetivo (en rojo), es detectado con la intersección de dichas esferas.

Por otro lado, cuando se habla de medidas de distancia diferenciales, la intersección será de hipérboles, donde se requieren cuatro puntos de referencia para los sistemas tridimensionales. Esta técnica recibe el nombre de trilateración hiperbólica y se relaciona con la medida de TDoA. En la Figura 2.1 (b) se observa un ejemplo de trilateración hiperbólica. Se tienen cuatro puntos de referencia (en verde) desde los que se trazan sendas hipérbolas, con las que el objetivo (en rojo) es detectado.

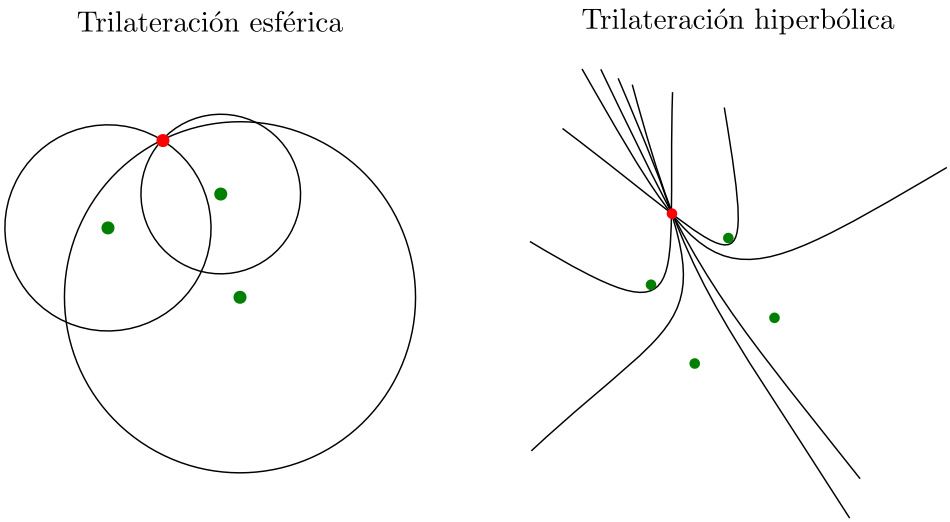

Figura 2.1: Ejemplo de trilateración esférica (a) e hiperbólica (b) aplicado al mismo escenario.

- La estrategia de angulación se refiere al proceso de estimación de la posición de un objetivo a partir de mediciones de ángulos entre este y varios puntos de referencia cuya ubicación es conocida. Esta estrategia se extiende principalmente en mediciones geodésicas y aplicaciones ópticas de alta precisión. Esta técnica se relaciona con las medidas de posicionamiento AoA y RSS. En la Figura 2.2 se observa la angulación de un objetivo mediante la técnica de AoA. En este caso, se tienen dos puntos de referencia (en verde) y un objetivo (en rojo). Se trazan líneas rectas desde la perpendicular del punto de referencia, que da lugar a un ángulo θ conocido.

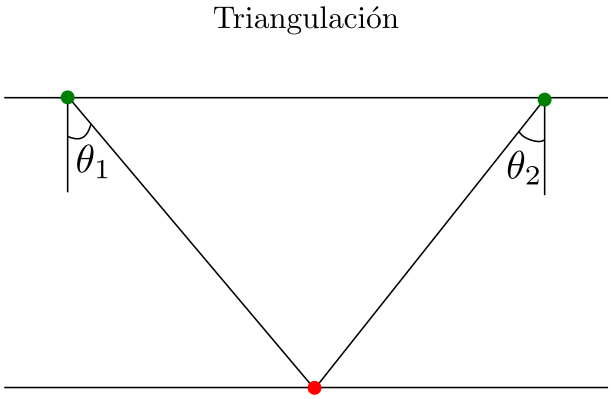

Figura 2.2: Ejemplo de angulación basada en medidas AoA

- La estrategia de navegación por estima o DR (del inglés *Dead Reckoning*) consiste en estimar la posición en función de una o varias posiciones previamente determinadas y en velocidades conocidas o estimadas durante un determinado periodo de tiempo. Esta estrategia utiliza sensores inerciales como fuente de medida (cuya definición se detalla en el apartado 2.2.2). Estos sensores obtienen datos físicos del objetivo para estimar su posición con respecto del entorno, siendo esta su técnica de medida. Una de las aplicaciones habituales de esta estrategia es la navegación marítima, aeronáutica o el análisis de los peatones para determinar su trayectoria. En cuanto a la aplicación marítima, en la Figura 2.3 se ilustra un ejemplo de su funcionamiento: la posición inicial a las 9:00 se representa con un triangulo y, en función de los datos inerciales obtenidos, se estima la posición en a las 9:30 y a las 10:00 (representado con una semi-circunferencia).

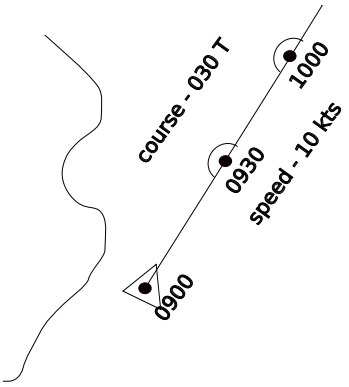

Figura 2.3: Ejemplo de navegación por estima

2.2.2. Sensores inerciales y dispositivos móviles

La navegación inercial es una técnica de navegación autónoma en la que se obtienen medidas con acelerómetros y giroscopios con el fin de seguir la posición y la orientación de un objeto, con respecto a un punto de partida, orientación y velocidad conocidos. En los siguientes subapartados, se definen los tipos de sensores que componen las IMU, la clasificación de estos sistemas [Woodman, 2007] [El-Sheimy and Youssef, 2020] y un análisis de los sensores disponibles en los dispositivos móviles Android [Shala and Rodriguez, 2011] [Android, 2019].

Sensores inerciales básicos: giroscopio y acelerómetro

Las IMU se componen normalmente de tres giroscopios y tres acelerómetros ortogonales, con los que se mide velocidad angular y aceleración lineal respectivamente.

- Un giroscopio es un dispositivo de medida que determina la orientación basándose en los principios de conservación del momento angular.

  La topología habitual de un giroscopio convencional se basa en una rueda giratoria montada sobre dos cardanes que le permiten girar con tres grados de libertad. De esta manera, si el giroscopio se somete a una rotación, la rueda permanece en una orientación global constante, donde se cambian únicamente los ángulos entre los cardanes adyacentes. Por tanto, los giroscopios convencionales miden orientación.

  Actualmente, el funcionamiento de estos dispositivos ha evolucionado hasta los sistemas microelectromecánicos o MEMS (del inglés *Micro Electro-Mechanical Systems*). Esta variante de giroscopios mide velocidad angular en lugar de orientación, de modo que reciben el nombre de giroscopios de velocidad o *rate-gyro*.

- Un acelerómetro es un dispositivo de medidas destinado a determinar aceleraciones. Su funcionamiento no se basa en los cambios de velocidad de un elemento en el espacio, sino que se refiere a la aceleración debida al fenómeno de peso en función de una masa de prueba, que se encuentra en el marco de referencia del elemento. Por tanto, los acelerómetros más simples miden la aceleración calculando cuanta fuerza se ejerce en función de la masa, de acuerdo con la Ley Fundamental de la Dinámica o Segunda Ley de Newton.

  Actualmente la evolución de estos dispositivos converge con los giroscopios, pues se tienen sistemas MEMS integrados en un chip de silicio. En este caso, la tecnología se basa en el traspaso térmico por convención natural, donde se miden cambios internos en la transferencia de calor debida a la aceleración.

Clasificación de las unidades de medida inercial

En general, la mayoría de las IMU se clasifican en sistemas de plataforma estable y sistemas *strapdown*. La diferencia principal entre ambos sistemas es el sistema de referencia que utilizan tanto el acelerómetro como el *rate-gyro*. En la Figura 2.4 se muestra el sistema de referencia utilizado normalmente para referirse a las IMU, donde al sistema de referencia de navegación se le llama *global frame* (indicado con el subíndice g) y al sistema de referencia del sujeto *body frame* (indicado con el subíndice b) .

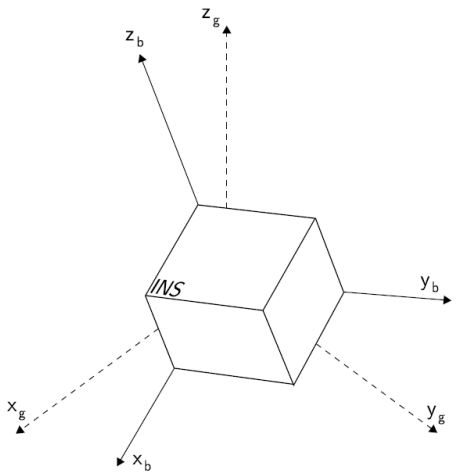

Figura 2.4: Sistema de referencia de la IMU. Los ejes $[x, y, z]_g$ corresponden al *global frame*, mientras que los ejes $[x, y, z]_b$ corresponden al *body frame*. Ilustración extraída de [Woodman, 2007]

- Los sistemas de plataforma estable se montan en una plataforma aislada de cualquier movimiento de rotación externa y, sobre estos, se disponen los sensores inerciales. La plataforma se mantiene alineada con el *global frame* en todo momento.

  Para obtener dicha estabilidad, la plataforma se monta con cardanes o *gimbals* que dan tres grados de libertad, los giroscopios detectan cualquier rotación dada en la plataforma y realimentan a unos motores de torque que giran los cardanes y cancelan dichas rotaciones. En la Figura 2.5, se muestra la topología de los sistemas de plataforma estable, donde se ilustran los diferentes componentes (cardan, torque, acelerómetro y giroscopio).

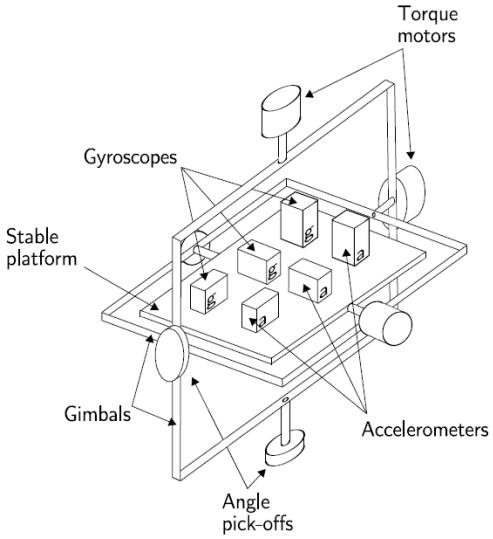

Figura 2.5: IMU del tipo plataforma estable. Ilustración extraída de [Travagnin, 2020]

- Los sistemas *strapdown*, por su parte, ubican los sensores inerciales directamente en el dispositivo, de manera que se obtienen directamente las medidas del *body frame*, en lugar del *global frame* como sucede en los sistemas de plataforma estable. Esta topología se ilustra en la Figura 2.6. Se puede observar tres dispositivos *rate-gyro* y tres acelerómetros dispuestos sobre un vehículo. Además, el sistema cuenta con un conjunto de sensores electrónicos adicionales y un ordenador que sirve para procesar los datos obtenidos por los distintos sensores.

  En estos sistemas, la información se extrae mediante el tratamiento de los datos obtenidos por los sensores, donde la orientación proviene de los datos de los *rate-gyro* y la posición de los datos de los acelerómetros. En la actualidad, estos son los sistemas más utilizados.

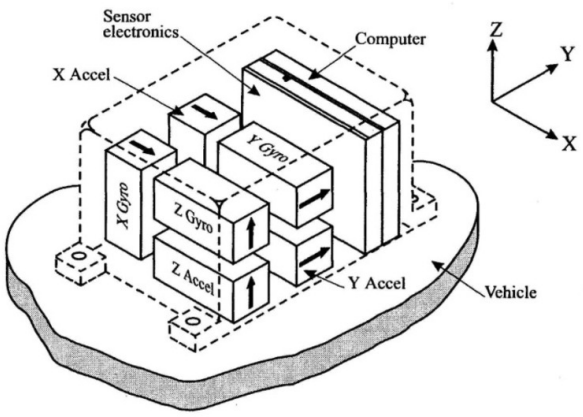

Figura 2.6: IMU del tipo *strapdown*. Ilustración extraída de [Travagnin, 2020]

Sensores inerciales en Android

La mayoría de los dispositivos con Android tienen sensores integrados que miden el movimiento, la orientación y diversas condiciones ambientales. Estos sensores son capaces de proporcionar datos sin procesar con alta precisión y exactitud, y son útiles para supervisar el movimiento o posicionamiento tridimensional del dispositivo [Android, 2019]. La plataforma Android divide los distintos conjuntos de sensores en tres tipos: sensores de movimiento, sensores de posición y sensores de entorno. El tipo de sistema que integra estos dispositivos es *strapdown* con sensores MEMS, detallados en el apartado anterior.

En este trabajo, resulta de interés analizar únicamente los sensores de movimiento. En la plataforma Android estos sensores se agrupan por topologías: dependientes únicamente de hardware o dependientes de hardware y software.

- Los sensores del acelerómetro y del giroscopio siempre están basados en hardware.
- Los sensores vectoriales de rotación, de gravedad, de aceleración lineal, de movimiento significativo, de contador de pasos y de detector de pasos se basan en hardware y en software.

Los sensores de movimiento resultan útiles para supervisar y analizar los movimientos del dispositivo, tales como la inclinación, la elevación, la vibración, la rotación o el balanceo. Estas variaciones y movimientos se relacionan directamente con como interactúa el usuario con el dispositivo, de modo que se pueden utilizar para obtener parámetros de interés en posicionamiento.

El sistema de coordenadas del sensor, al que a partir de ahora se le hará coincidir con el *body frame* (Figura 2.4), se enmarca en tres ejes coordenados $[x, y, z]$. Todos los sensores se referencian con la pantalla del dispositivo cuando este se dispone en la posición predeterminada. En dicho caso, el sistema de coordenadas se establece como se observa en la Figura 2.7.

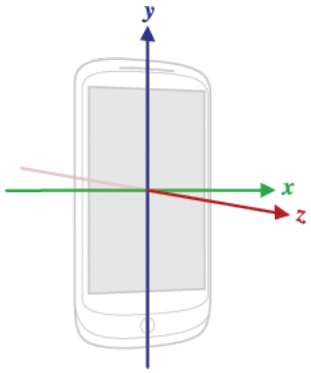

Figura 2.7: Sistema de coordenadas que usa la plataforma Android. Ilustración extraída de [Android, 2019]

2.2.3. Pedestrian Dead Reckoning (PDR)

Debido a su bajo coste, la tecnología MEMS integrada en las IMU se extiende a la gran mayoría de los dispositivos móviles. Las principales ventajas de este tipo de sensores son: su reducido tamaño, su ligereza y su bajo consumo de energía.

La navegación por estima peatonal o PDR (del inglés *Pedestrian Dead Reckoning*) es una particularización de la estrategia DR. Su funcionamiento se basa en el análisis de los datos obtenidos por sensores inerciales (IMU) para determinar la posición de objetivos (personas o peatones) según los movimientos que realizan. En comparación con las técnicas de localización basadas en señales inalámbricas y otros sensores, el PDR puede dar una posición precisa en un período corto de tiempo, lo que permite asumir que la localización es en tiempo real. Resulta interesante destacar que, dado que no se requiere una infraestructura adicional, los sistemas PDR son sencillos y autónomos. Cabe destacar que, debido a los errores acumulativos que se producen, estos sistemas se descartan en ciertas aplicaciones de navegación (sustituidos por GNSS), donde su principal propósito es la navegación inercial.

Actualmente estos sistemas se dividen en dos grandes grupos: PDR montados, basados en un dispositivo especializado y sistemas PDR portátiles, basados en dispositivos, generalmente, sujetos en la mano. En los sistemas PDR montados la precisión del dispositivo es mayor y se monta en una parte determinada del cuerpo, como los pies, las piernas o la cintura. Debido al error acumulativo, la precisión de la localización del PDR montado disminuirá con el tiempo, por lo que se utiliza un algoritmo de actualización de velocidad cero (del inglés *Zero velocity Update*) para controlar el error acumulativo.

Una forma de evitar que el dispositivo se monte en el cuerpo es utilizar los sistemas PDR portátiles. Estos sistemas utilizan los sensores de los dispositivos móviles para obtener la ubicación y la dirección de los usuarios. En los PDR portátiles la ubicación se determina en tres fases.: detección de pasos, estimación de la longitud del paso y determinación del rumbo del usuario [Jimenez et al., 2009].

Habitualmente se asume que el ángulo de dirección o rumbo del usuario (ángulo entre la dirección del smartphone y la dirección del usuario) permanece constante. Esta suposición se satisface cuando los usuarios sostienen los smartphones en la mano, ya sea en la postura habitual de uso (sujeto con la mano y frente al cuerpo) o realizando una llamada (sujeto con la mano y cercano a una oreja). Sin embargo, en la cotidianidad de las personas, la postura del teléfono es arbitraria y no se puede garantizar que dicho ángulo sea constante. Con los trabajos [Wang et al., 2018] y [Li and Ning, 2018] como referencia, se puede obtener una ecuación general de PDR (2.1) que asuma estas perturbaciones en la posición del dispositivo:

$$\begin{cases} X_k = X_{k-1} + L_{k-1,k} \cdot \sin(\psi_{k-1,k}) \\ Y_k = Y_{k-1} + L_{k-1,k} \cdot \cos(\psi_{k-1,k}) \end{cases} \tag{2.1}$$

donde X e Y son las coordenadas en el Este y el Norte (referido a los sistemas georreferenciados[2]), L es la longitud del paso, ψ es el rumbo del usuario durante un paso, y k denota el índice del paso del usuario.

[2]La georreferenciación es el uso de coordenadas de mapa para asignar una ubicación espacial a entidades cartográficas. Norte y Este se refieren a proyecciones de mapa a través de coordenadas cartesianas.

Capítulo 3

Desarrollo

En este Capítulo se proponen e implementan los algoritmos del presente TFM. La finalidad de este apartado es obtener, por medio del desarrollo técnico, dos soluciones que den lugar al cumplimiento de los objetivos expuestos en el apartado 1.2.

En primer lugar (apartado 3.1), se define la topología del proyecto GUIA y el diagrama que gobierna las contribuciones que se van a implementar. A continuación, se detalla el proceso de creación, modelado e implementación de los algoritmos: el apartado 3.2 para las correcciones de la posición mediante el algoritmo basado en AoA y el apartado 3.3 para las estimaciones de la posición mediante el algoritmo basado en PDR.

3.1. Topología de la aplicación

El proyecto GUIA tiene por objetivo desplegar una red de luminarias inteligentes que permiten posicionar a usuarios mediante una aplicación móvil. Para ello, los usuarios deben abrir dicha aplicación móvil, donde aparece un mapa con la posición actual. En la Figura 3.1, se muestra una ilustración de los elementos básicos de la topología foco-usuario: el usuario y el dispositivo móvil (ambos representan al receptor) y el foco inteligente (emisor).

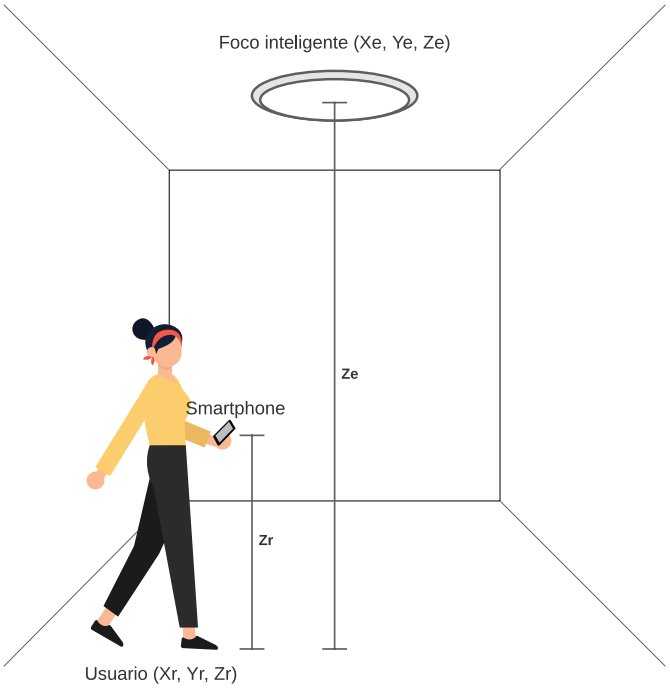

Figura 3.1: Topología foco-usuario del proyecto GUIA

En la Figura 3.1 se muestra a un usuario que sostiene el dispositivo móvil en la mano, hacia arriba y con el ángulo de dirección alineado con respecto del usuario. A esta posición del dispositivo se le denominará **posición natural**. En esta condición, se asume que el dispositivo es coplanario al suelo, es decir, el plano $x - y$ del dispositivo es paralelo al plano $x - y$ del la Tierra (el suelo).

El diagrama de bloques del funcionamiento general de la aplicación se ilustra en la Figura 3.2. En la ilustración, se muestra en **negrita** la contribución de este TFM al proyecto y en *cursiva* aquellos bloques ya implementados y ajenos a este TFM.

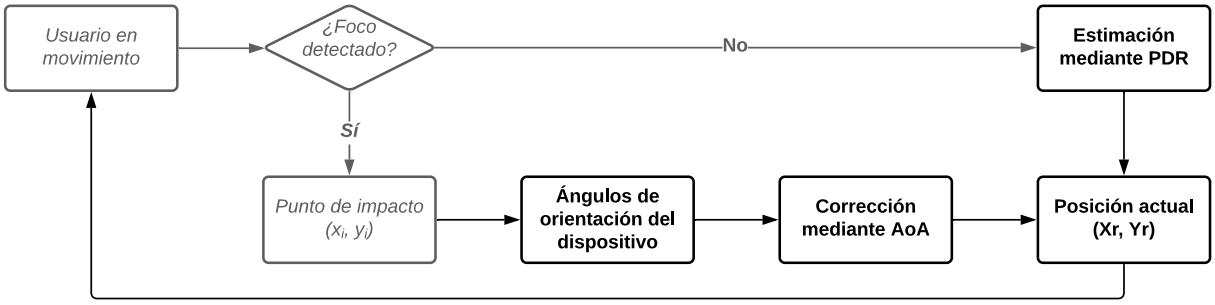

Figura 3.2: Diagrama de bloques de la aplicación del proyecto GUIA

3.2. Correcciones en la posición de objetos detectados para medidas de posicionamiento basadas en AoA

3.2.1. Modelo matemático basado en *pinhole* para determinar la posición de objetos

El modelo formado por la topología de la aplicación (apartado 3.1) está basado en un modelo de *pinhole*. La relación existente entre las coordenadas 3D del emisor (X_e, Y_e, Z_e) y las coordenadas 2D del punto de impacto en el dispositivo móvil (x_i', y_i') se define como:

$$\begin{pmatrix} sx_i' \\ sy_i' \\ s \end{pmatrix} = \begin{pmatrix} f & 0 & C_x \\ 0 & f & C_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X_e - X_r \\ Y_e - Y_r \\ Z_e - Z_r \end{pmatrix} = \mathbf{AR} \begin{pmatrix} X_e - X_r \\ Y_e - Y_r \\ Z_e - Z_r \end{pmatrix} \tag{3.1}$$

donde s es el factor de escala de la proyección, f es la distancia focal del sensor del dispositivo y (C_x, C_y) son las coordenadas de su centro óptico.

La matriz intrínseca $\mathbf{A}$ está definida por los parámetros del objetivo f, C_x y C_x, que pueden obtenerse mediante una calibración geométrica del sistema. El cálculo de las coordenadas 2D del punto de impacto en el dispositivo se obtiene en la aplicación mediante la detección de los focos, que devuelve los valores de impacto medidos (x_i, y_i). La matriz extrínseca $\mathbf{R}$ se calcula a partir de los tres ángulos de Euler que definen la orientación del dispositivo con respecto al sistema de coordenadas del mundo.

En relación con la topología de la aplicación (detallada en el apartado 3.1), el dispositivo móvil recibe una única señal desde el emisor. Con el artículo [De-La-Llana-Calvo et al., 2020] de referencia, se puede relacionar la posición del emisor y la posición del dispositivo móvil mediante una particularización del modelo de *pinhole* definido en la ecuación (3.1) para posicionamiento de un único emisor.

Se suponen conocidos los valores de las matrices $\mathbf{A}$ (parámetros del dispositivo) y $\mathbf{R}$ (orientación del dispositivo), la altura del foco emisor Z_e y del dispositivo Z_r. Por tanto, la posición del receptor (X_r, Y_r, Z_r) se obtiene como:

$$\begin{pmatrix} X_r \\ Y_r \\ Z_r \end{pmatrix} = -\mathbf{R}^{-1}\mathbf{A}^{-1} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} s + \begin{pmatrix} X_e \\ Y_e \\ Z_e \end{pmatrix} \tag{3.2}$$

Si se define el vector $\mathbf{q}$ como:

$$\mathbf{q} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} = -\mathbf{R}^{-1}\mathbf{A}^{-1} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{3.3}$$

y el valor del factor de escalado s:

$$s = \frac{Z_r - Z_e}{q_3} \tag{3.4}$$

se obtiene la posición del receptor como:

$$\begin{pmatrix} X_r \\ Y_r \\ Z_r \end{pmatrix} = \mathbf{q}s + \begin{pmatrix} X_e \\ Y_e \\ Z_e \end{pmatrix} \tag{3.5}$$

En conclusión, conocidos todos los valores de la ecuación (3.5), se puede obtener la posición del usuario a pesar de las perturbaciones en el ángulo de orientación del dispositivo. Es decir, se corrige la posición del dispositivo de forma que no afecte a la posición obtenida del usuario.

3.2.2. Parámetros del modelo

El dispositivo móvil utilizado para las pruebas del proyecto es una tablet Samsung Galaxy Tab S3 (SM-T820). Con los datos proporcionados en los detalles de las imágenes capturadas por la cámara frontal, se pueden determinar los parámetros del dispositivo:

Parámetro	**Valor**
Longitud focal (f)	3,31 mm
Tamaño del píxel	1,20 μm
Nº filas por imagen	2560 px
Nº columnas por imagen	1920 px

Tabla 3.1: Parámetros del sensor frontal de la cámara. Samsung Galaxy Tab S3 (SM-T820)

Matriz intrínseca A

Las coordenadas del centro óptico del dispositivo (que se considera el centro del sensor) se obtienen mediante los parámetros de la tabla 3.1, que relacionan el tamaño del píxel d, con las filas w y columnas h de las imágenes que capta el sensor. Así:

$$C_x = d\frac{w}{2}, \quad C_y = d\frac{h}{2} \tag{3.6}$$

Conocidos los valores de las coordenadas del centro óptico del dispositivo y la longitud focal, se puede escribir la matriz **A** como:

$$\mathbf{A} = \begin{pmatrix} f & 0 & d\frac{w}{2} \\ 0 & f & d\frac{h}{2} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0{,}003310 & 0{,}000000 & 0{,}001152 \\ 0{,}000000 & 0{,}003310 & 0{,}001536 \\ 0{,}000000 & 0{,}000000 & 1{,}000000 \end{pmatrix} \tag{3.7}$$

Matriz extrínseca R

La matriz de rotación o matriz extrínseca $\mathbf{R}$ se obtiene mediante las rotaciones de los ángulos de Euler, que se definen gracias a los sensores del dispositivo (detallado en el apartado 3.2.3).

Se definen las matrices de rotación de cada uno de los ángulos de Euler: $\mathbf{R}_X(\phi)$ para la inclinación, $\mathbf{R}_Y(\theta)$ para la elevación y $\mathbf{R}_Z(\psi)$ para la dirección.

$$\mathbf{R}_X(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \tag{3.8}$$

$$\mathbf{R}_Y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{3.9}$$

$$\mathbf{R}_Z(\psi) = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

Se obtiene la matriz de rotación $\mathbf{R}$ multiplicando las rotaciones de los ángulos de Euler (alrededor de los ejes):

$$\mathbf{R} = \mathbf{R}_Z(\psi)\mathbf{R}_Y(\theta)\mathbf{R}_X(\phi) \tag{3.11}$$

Punto de impacto

El punto de impacto (x_i', y_i') se determina en la aplicación del proyecto gracias a un algoritmo de detección de círculos (focos). Este algoritmo reconoce formas circulares y obtiene la posición en la que se encuentra el centro del foco en la imagen captada y su radio. Estos valores se relacionan con los parámetros del dispositivo de la tabla 3.1 para que el centro óptico coincida con el centro del impacto.

$$\begin{aligned} x_i &= d \cdot (w - x_i') \\ y_i &= d \cdot (h - y_i') \end{aligned} \tag{3.12}$$

En la ecuación (3.12), se obtiene el punto de impacto (x_i, y_i) mediante los parámetros del tamaño del píxel d, las filas w y las columnas h, relacionadas con el punto de impacto medido (x_i', y_i').

Altura del receptor y posición del emisor

La altura del receptor Z_r se corresponde con la distancia entre el centro del dispositivo y el suelo. Con el fin de generalizar este valor, se define como constante para la mayoría de los usuarios a partir de los estudios estadísticos del artículo [Collaboration et al., 2016]. En España, la estatura media se establece en 170 cm para ambos sexos. Por otro lado, el dispositivo se suele sostener a una distancia de 50 cm desde la cabeza, de modo que se establece un valor de $Z_r = 120$ cm.

La posición del emisor (X_e, Y_e, Z_e), se obtiene una vez se ha identificado el foco en cuestión. Como en el despliegue de la red de luminarias inteligentes se disponen los focos en posiciones conocidas, cuando la aplicación determina si se ha detectado un foco u otro se conoce también dicha posición.

3.2.3. Implementación del algoritmo AoA en Kotlin

Una vez definidos los valores de la ecuación (3.5) (apartado 3.2.2), se puede implementar el código en Kotlin que permite realizar las correcciones en las medidas de posicionamiento basadas en AoA.

Se definen los parámetros del dispositivo:

Código 3.1: Declaración de parámetros del dispositivo

```
val numColFoto = 1920        // X
val numFilasFoto = 2560      // Y
val sizePixel = 0.0000012    // metros
val f = 0.00331              // metros
```

Se define la matriz intrínseca **A**:

Código 3.2: Matriz intrínseca `A`

```
val A = Matrix(3,3)
A[0,0] = f                          // a11
A[1,1] = f                          // a22
A[2,2] = 1.0                        // a33
A[0,2] = (numColFoto/2)*sizePixel   // a13
A[1,2] = (numFilasFoto/2)*sizePixel // a23
```

La API de Android destinada a la comunicación entre los sensores del dispositivo y el sistema operativo [Android, 2019] proporciona dos métodos que permiten obtener los ángulos de Euler gracias al acelerómetro y al magnetómetro[1]: `getRotationMatrix` y `getOrientation`.

Código 3.3: Función `updateOrientationAngles`

[1]Cómo computar la orientación del dispositivo: https://developer.android.com/guide/topics/sensors/sensors_position#sensors-pos-orient

```
fun updateOrientationAngles() {
    SensorManager.getRotationMatrix(
        rotationMatrix,
        null,
        accelerometerReading,
        magnetometerReading
    )
    SensorManager.getOrientation(
        rotationMatrix,
        orientationAngles
    )
}
```

La salida de `getOrientation` proporciona un array de tres elementos que contiene los ángulos de Euler:

Código 3.4: Declaración de los ángulos de orientación

```
val azimuth = orientationAngles[0]   // direccion
val pitch = orientationAngles[1]     // elevacion
val roll = orientationAngles[2]      // inclinacion
```

Se obtiene la matriz de rotación **R**:

Código 3.5: Matriz de rotación `R`

```
val rotAzimuth = Matrix(3,3)
rotAzimuth[0,0] = cos(azimuth)
rotAzimuth[0,1] = -sin(azimuth)
rotAzimuth[0,2] = 0.0
rotAzimuth[1,0] = sin(azimuth)
rotAzimuth[1,1] = cos(azimuth)
rotAzimuth[1,2] = 0.0
rotAzimuth[2,0] = 0.0
rotAzimuth[2,1] = 0.0
rotAzimuth[2,2] = 1.0

val rotPitch = Matrix(3,3)
rotPitch[0,0] = cos(pitch)
rotPitch[0,1] = 0.0
rotPitch[0,2] = sin(pitch)
rotPitch[1,0] = 0.0
rotPitch[1,1] = 1.0
rotPitch[1,2] = 0.0
rotPitch[2,0] = -sin(pitch)
rotPitch[2,1] = 0.0
rotPitch[2,2] = cos(pitch)

val rotRoll = Matrix(3,3)
rotRoll[0,0] = 1.0
rotRoll[0,1] = 0.0
rotRoll[0,2] = 0.0
rotRoll[1,0] = 0.0
rotRoll[1,1] = cos(roll)
rotRoll[1,2] = -sin(roll)
rotRoll[2,0] = 0.0
rotRoll[2,1] = sin(roll)
rotRoll[2,2] = cos(roll)

```

```
34 val R = rotAzimuth * rotPitch * rotRoll
```

El punto de impacto se codifica de la forma:

Código 3.6: Vector de impacto en el dispositivo `impact`

```
1 val impact = ColumnVector(3) // [xi; yi; zi]
2 impact[0] = (numColFoto - herramientaFFT.iDEmittersCircleCol[0])*sizePixel      //xi
3 impact[1] = (numFilasFoto - herramientaFFT.iDEmittersCircleRow[0])*sizePixel    //yi
4 impact[2] = 1.0                                                                 //zi
```

Finalmente, se definen $\mathbf{q}$ y s para obtener la posición del receptor deseada:

Código 3.7: Vector de posición del usuario `receiver`

```
1 var receiver = ColumnVector(3) // [Xr; Yr; Zr]
2 receiver[2] = 1.20 // Zr
3
4 val q = - (R.inverse()) * A.inverse() * impact
5 val s = (receiver[2] - emitter[2])/q[2] // (Zr - Ze) / q3
6
7 receiver = q * s + emitter // Modelo de pinhole para un solo emisor
```

3.3. Estimaciones de la posición del usuario mediante PDR

La aplicación del proyecto GUIA desarrollada hasta el momento se centra en aquellos momentos en los que se está captando un foco inteligente mientras el usuario se mueve por el museo (ver Figura 3.2). La contribución del algoritmo de PDR permite estimar cuál es el desplazamiento del usuario en aquellos momentos en los que no se está captando un foco. Esto quiere decir que los parámetros de entrada (posición del estado anterior $\{X_{k-1}, Y_{k-1}\}$) que satisfacen la ecuación general de PDR (2.1) se obtienen de un primer foco captado (estado inicial). Una vez se deje de captar un foco, se ejecuta el algoritmo de estimación basado en PDR que permite obtener la navegación inercial del usuario, hasta que se encuentre con el siguiente foco inteligente.

Como se ha detallado en el apartado 2.2.3, los parámetros necesarios para obtener las coordenadas inerciales del usuario $\{X_k, Y_k\}$ se dividen en tres fases: detección de pasos (apartado 3.3.1), longitud del paso (apartado 3.3.2) y rumbo del usuario (apartado 3.3.3).

3.3.1. Detección de pasos

El parámetro k de la ecuación general (2.1) se refiere al evento producido en cada paso del usuario. Es decir, cada vez que el usuario da un paso se tiene una muestra nueva en la ecuación.

Como se estudia en el artículo de referencia [Wang et al., 2018], se utilizan los sensores inerciales del dispositivo para capturar los movimientos del usuario. En concreto, se aprovecha el acelerómetro para analizar el movimiento del dispositivo cuando el usuario está caminando y mirando el dispositivo a la vez (situación deseada en la aplicación del proyecto). En esta condición, el *smartphone* se sostiene en la posición natural (ver apartado 3.1), por lo tanto el dispositivo irá de arriba a abajo (movimiento oscilatorio) según se avance. Esto produce una señal sinusoidal en el eje z y los máximos representan cada paso k.

En la Figura 3.3, se muestran las señales obtenidas del acelerómetro respecto de los tres ejes coordenados cuando un usuario camina en las condiciones descritas. El valor de la señal en el eje z tiene su valor medio en torno al valor de la aceleración de la gravedad en la superficie de la Tierra $g = 9{,}8$ m/s^2. Este valor se acerca más a g cuando se cumple la condición de mantener el dispositivo coplanario al suelo.

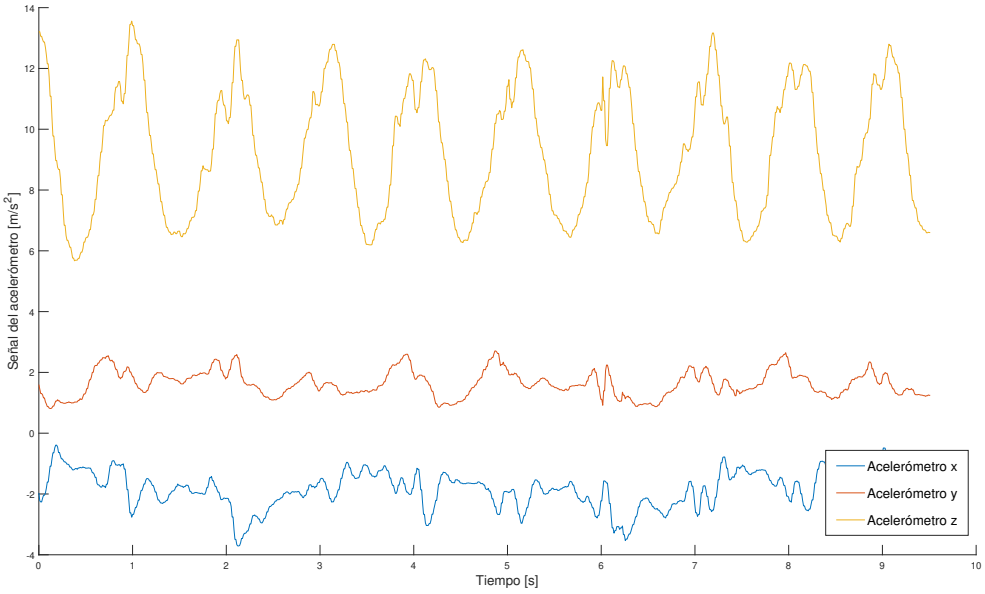

Figura 3.3: Señal obtenida del acelerómetro en los ejes coordenados

Si se obtiene el valor absoluto de las tres señales, se pueden representar las variaciones totales producidas por el dispositivo en una misma señal. Para extraer el valor máximo de la señal, se fija un valor de umbral desde el que se decide si se ha producido un paso o no. En la Figura 3.4, se muestra el valor absoluto de la señal del acelerómetro, el umbral definido y los pasos detectados.

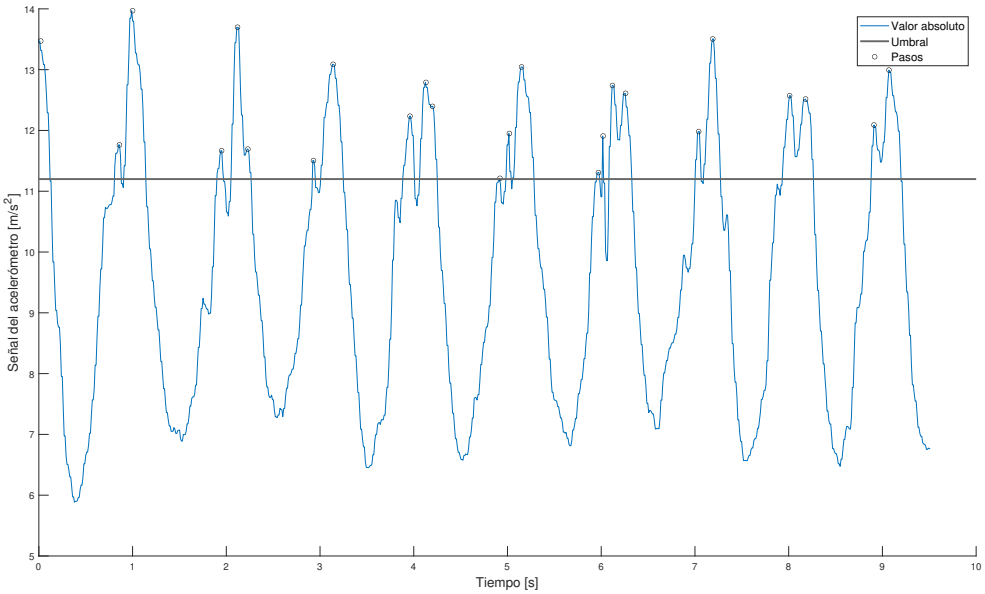

Figura 3.4: Valor absoluto de la señal obtenida del acelerómetro, umbral de detección de pasos y pasos detectados

Cabe destacar que, tal y como se documenta en diversas implementaciones de algoritmos de PDR como [Jimenez et al., 2009], [Wang et al., 2018], [Guo and Uradzinski, 2018], este umbral depende directamente de la forma de caminar del usuario y de las diferentes posiciones del dispositivo. En ese sentido, se ha establecido un umbral de 11,2 m/s^2 como valor general para la mayoría de los usuarios a partir de los resultados obtenidos en dichos trabajos.

A la vista de los eventos detectados en la Figura 3.3, se observa que debido al ruido existente producido por la sensibilidad del sensor, hay falsos positivos en las detecciones de los máximos. Para paliar este efecto, se opta por realizar un filtrado paso bajo de respuesta al impulso infinita que garantice una aproximación a la señal sinusoidal deseada.

Por definición, la ecuación que define este filtro digital para secuencias iterativas[2] es la siguiente:

$$y[i] = y[i-1] + \alpha(x[i] - y[i-1]) \tag{3.13}$$

donde i es el instante actual, $i-1$ el instante anterior, y la señal de salida, x la señal de entrada y α el factor de suavizado.

[2]El procesado de este filtro debe ser iterativo, de modo que ante sistemas LIT (lineales e invariantes en el tiempo) este filtrado es más eficiente. El caso de una señal que es aproximadamente sinusoidal cumple la condición de sistema LIT, por lo que los resultados se aproximan fielmente a la envolvente de las señales sinusoidales.

El valor del factor de suavizado se obtiene mediante la relación entre la constante de tiempo τ, el tiempo de muestreo T_s y la frecuencia de corte f_c que se selecciona para implementar el filtro.

$$\tau = \frac{1}{2\pi f_c} \rightarrow \alpha = \frac{T_s}{\tau + T_s} \tag{3.14}$$

Se plantean estas ecuaciones para filtrar el valor absoluto de la señal del acelerómetro. Para ello, se seleccionan los parámetros necesarios con el fin de obtener el valor de α.

Los sensores de los dispositivos móviles pueden trabajar a diversas frecuencias de muestreo, pero en el caso que ocupa se selecciona $f_s = 100$ Hz, lo que da lugar a $T_s = 10$ ms. Por otro lado, la frecuencia de corte se fija en $f_c = 15$ Hz (basado en el estudio empírico de [Wang et al., 2018]). Con estos datos, se obtiene que $\alpha = 0{,}4851$.

Una vez se ha implementado este filtro, se obtiene la señal que se muestra en la Figura 3.5, donde se muestra una disminución en el número de falsos positivos.

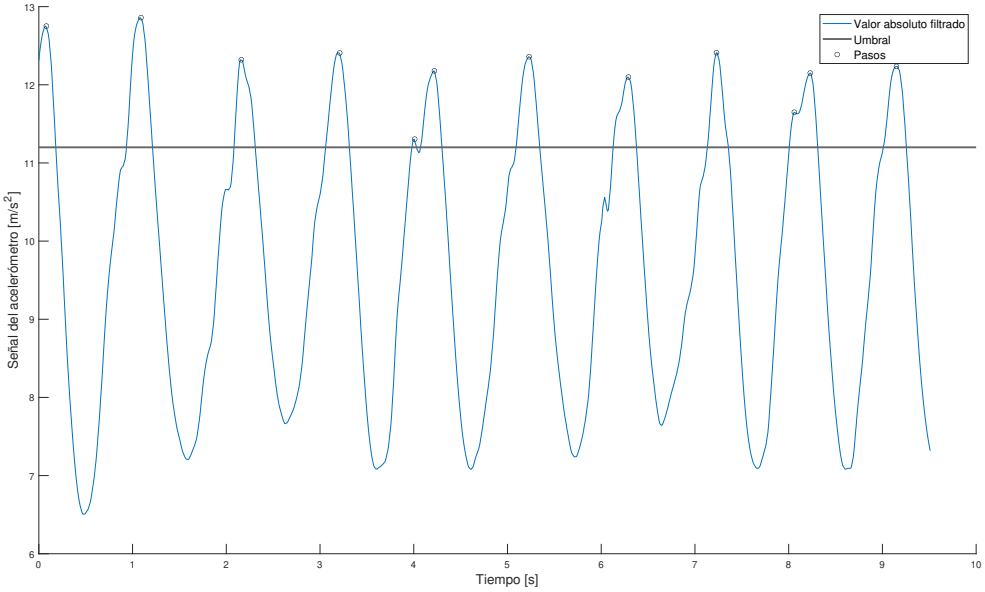

Figura 3.5: Valor absoluto de la señal obtenida del acelerómetro con filtrado paso bajo, umbral de detección de pasos y pasos detectados

En última instancia, se debe añadir un mecanismo de protección adicional frente a los falsos positivos, que se relacione con la solución de compromiso propuesta para el umbral de 11,2 m/s^2. Así, se diseña un algoritmo que implementa este mecanismo de protección, cuyo funcionamiento se basa en trazar ventanas que evalúan si en los estados anteriores se ha obtenido un pico (posible

máximo local) o no. En este caso, se habrá superado el umbral recientemente, de modo que se trata de un falso positivo. El diagrama de bloques de este algoritmo se ilustra en la Figura 3.6.

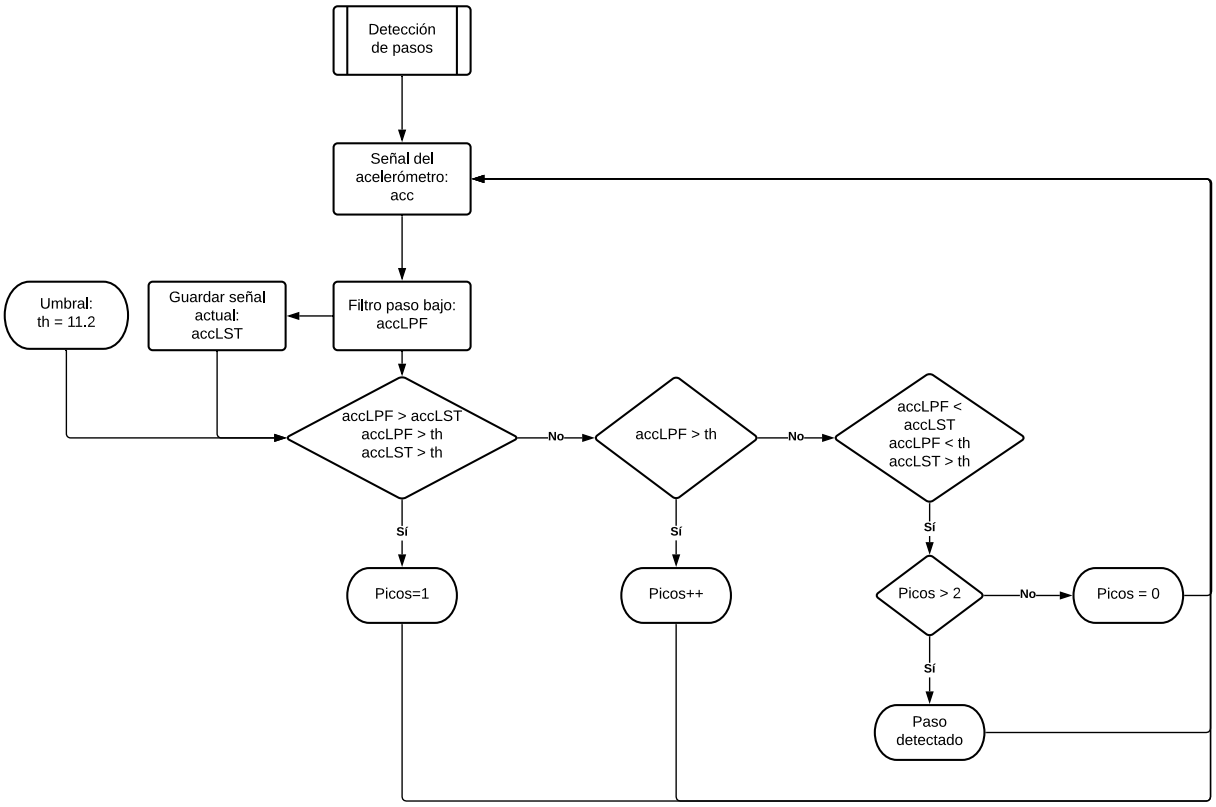

Figura 3.6: Diagrama de bloques del mecanismo de protección de falsos positivos mediante ventanas

Cuando se aplica este algoritmo, desaparecen los falsos positivos y el sistema incrementa su robustez frente a perturbaciones. En la Figura 3.7, se muestra la señal que se ha analizado sin falsos positivos.

3.3.2. Longitud del paso

La longitud del paso se confunde a menudo, o se utiliza indistintamente, con la longitud de la zancada. Sin embargo, técnicamente no son la misma medida. La longitud del paso es la distancia entre el golpe de talón de un pie y el golpe de talón del pie opuesto al caminar; la longitud de la zancada, por su parte, es la distancia entre dos golpes consecutivos de talón del mismo pie.

Existen diversos factores que intervienen en la longitud de los pasos: altura de la persona, velocidad del movimiento, cadencia, longitud del pie, etc. A partir de los estudios realizados por la Universidad de Oklahoma [Thompson, 2002], se establecen estadísticos generales en hombres y mujeres para estos parámetros. En la siguiente tabla se presenta un resumen de este estudio:

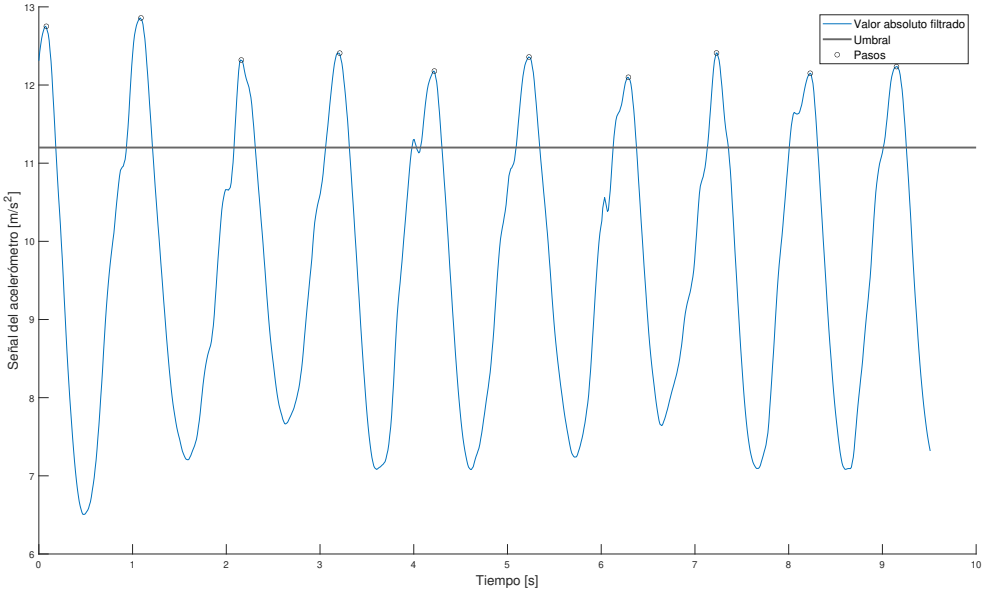

Figura 3.7: Detección de pasos con algoritmo de protección de falsos positivos

	Hombres	**Mujeres**	**Media general**
Longitud del paso (cm)	79	66	72,5
Longitud de la zancada (cm)	158	132	145
Cadencia (pasos/min)	117	117	117
Velocidad (m/s)	1,54	1,31	1,425

Tabla 3.2: Estadísticos del estudio de la Universidad de Oklahoma para la forma de caminar de las personas. Elaboración propia basada en [Thompson, 2002]

La longitud de los pasos se relaciona directamente con la ecuación general de PDR (2.1), donde $L_{k-1,k}$ representa la longitud del paso en cada actualización del algoritmo. La nomenclatura del subíndice $k-1, k$ se refiere, por tanto, a la distancia entre la ocurrencia del evento anterior ($k-1$, paso detectado anterior) y el evento actual (k, paso detectado actual). El valor de este parámetro puede ser constante o variable, dependiendo de la precisión requerida para la aplicación.

- Si el parámetro de la ecuación es variable, en cada iteración se ajusta a la longitud del paso que se ha dado, de modo que la distancia total calculada será semejante a la distancia total recorrida.
- Si el parámetro de la ecuación es constante, el error producido será mayor cuanto más diste la longitud del paso de la realidad. Cabe destacar que si se trabaja con valores

medios se puede generalizar esta constante sin conocer las características del usuario que está caminando.

En el proyecto GUIA, el despliegue de la red de luminarias se realiza en distancias de un máximo de 15 m entre sí. Como el algoritmo de PDR se activa cuando no se detecta un foco (Figura 3.2), la distancia que se recorre a lo largo del museo en esta condición produce errores asumibles, es decir, de órdenes de magnitud inferiores a estas distancias. Cuando se detecta un nuevo foco, la posición del usuario se actualiza con la posición del algoritmo basado en AoA. Por esta razón, se selecciona una longitud del paso constante. Si se aplica la longitud del paso constante a la ecuación general de PDR, se tiene que $L_{k-1,k} = L$. En este caso, se puede reescribir como:

$$\begin{cases} X_k = X_{k-1} + L \cdot \sin(\psi_{k-1,k}) \\ Y_k = Y_{k-1} + L \cdot \cos(\psi_{k-1,k}) \end{cases} \tag{3.15}$$

A la vista de la tabla 3.2, el valor de la constante de la longitud del paso es $L = 72{,}5$ cm. Hay que tener en cuenta que este estudio se realiza en situaciones generalizadas de la forma de andar de las personas. Dado que esta aplicación está destinada a su uso en museos, en esta situación la velocidad media de las personas es inferior al valor generalizado de la Tabla 3.2, de modo que se disminuye en un 25 % el valor de la longitud del paso (basado en el estudio empírico de [Wang et al., 2018]). Bajo esta condición, el valor seleccionado para la longitud del paso es $L = 58$ cm.

3.3.3. Rumbo del usuario

El ángulo de dirección es el parámetro que se utiliza para determinar el rumbo que sigue el usuario en su trayectoria al caminar.

Este parámetro se determina según el sistema de coordenadas del dispositivo (*body frame*) con respecto del sistema de coordenadas de la Tierra (*global frame*). Se asume una posición natural a la hora de sostener el dispositivo (apartado 3.1), de modo que el ángulo de dirección a determinar coincide con el ángulo de orientación.

En la ecuación de PDR (2.1), este parámetro se denomina $\psi_{k-1,k}$. Se refiere al ángulo que forma la dirección anterior con respecto de la dirección actual. Como se asume que el dispositivo está alineado con respecto del sistema de referencia, se puede decir que este ángulo depende únicamente del rumbo actual del usuario. Es decir, el ángulo de dirección deseado cumple que $\psi_{k-1,k} = \psi_k$.

Con todos los parámetros obtenidos, la ecuación de PDR se reescribe como:

$$\begin{cases} X_k = X_{k-1} + L \cdot \sin \psi_k \\ Y_k = Y_{k-1} + L \cdot \cos \psi_k \end{cases} \tag{3.16}$$

3.3.4. Implementación del algoritmo PDR en Kotlin

Una vez definidos los valores de la ecuación de PDR (3.16), se puede implementar el código en Kotlin que permite realizar las estimaciones de posicionamiento en aquellos instantes en los que no se capta un foco, de acuerdo con el diagrama de bloques ilustrado en la Figura 3.2. Se implementa una función para cada uno de los pasos seguidos en el apartado 3.3.

En primer lugar se necesitan los datos del acelerómetro y magnetómetro para determinar la aceleración y la orientación del dispositivo. Los datos de la orientación del dispositivo ya se han codificado para el algoritmo de AoA (apartado 3.2.3), mientras que los datos del acelerómetro se obtienen gracias a las recomendaciones de la documentación de Android [Android, 2019]:

Código 3.8: Valores del acelerómetro con la función `onSensorChanged`

```
private val accelerometerReading = FloatArray(3)

override fun onSensorChanged(event: SensorEvent) {
    if (event.sensor.type == Sensor.TYPE_ACCELEROMETER) {
        System.arraycopy(event.values, 0, accelerometerReading, 0, accelerometerReading.size)
    }
}
```

Se obtiene el valor absoluto de la aceleración en los tres ejes coordenados con la función `magnitudeAccelerometer` y un filtrado paso bajo de las características descritas en el apartado 3.3.1 (ecuación (3.13) con $f_c = 15$ Hz) con la función `lowPassFilter`:

Código 3.9: Función `magnitudeAccelerometer`

```
/**
 * Se calcula el valor absoluto de [input]
 *
 * @param input variable del tipo [FloatArray] que contiene los componentes
 * del vector
 * @return el valor absoluto de [input]
 */
private fun magnitudeAccelerometer(input: FloatArray) : Float {
    val x = input[0]
    val y = input[1]
    val z = input[2]
    return sqrt(x * x + y * y + z * z)
}
```

Código 3.10: Función `lowPassFilter`

```
private val alpha: Float = 0.4851F // 15 Hz

/**
 * Se calcula un filtro paso bajo de las variables de entrada en
 * funcion de la entrada [input] y la salida en el estado anterior
 * [oldOutput]
 *
 * @param input entrada del filtro
 * @param oldOutput estado anterior de la salida filtro
 * @return la entrada [input] filtrada
 */
```

```
12 private fun lowPassFilter(input: Float, oldOutput: Float): Float {
13     return oldOutput + alpha * (input - oldOutput)
14 }
```

Se codifica la función de detección de pasos `stepDetection`, que devuelve `true` si el usuario ha dado un paso. Esta función sigue la lógica descrita por el diagrama de flujo de la Figura 3.6.

Código 3.11: Función `stepDetection`

```
1  private val threshold : Float = 11.2F // Umbral de deteccion de pasos
2
3  /**
4   * Se detecta si el usuario ha dado un paso para obtener las muestras
5   * del PDR.
6   *
7   * @param accelerometerReading lecturas del acelerometro
8   * @return true si se ha dado un paso, false si no se ha dado un paso
9   */
10 private fun stepDetection(accelerometerReading: FloatArray) : Boolean {
11     step = false
12     val accelerometerMagnitude = magnitudeAccelerometer(accelerometerReading)
13     val accelerometerFiltered = lowPassFilter(accelerometerMagnitude,
14         lastAccelerometerFiltered)
15
16     if ((accelerometerFiltered > lastAccelerometerFiltered) &&
17         (accelerometerFiltered > threshold) &&
18         (lastAccelerometerFiltered < threshold)){
19             counterPeaks=1
20     }
21     else if (accelerometerFiltered > threshold){
22         counterPeaks++
23     } else if ((accelerometerFiltered < lastAccelerometerFiltered) &&
24         (accelerometerFiltered < threshold) &&
25         (lastAccelerometerFiltered > threshold)){
26
27             if (counterPeaks > 2) {
28                 step = true
29                 reset = false
30                 counterPeaks = 0
31             } else {
32                 step = false
33                 counterPeaks = 0
34             }
35     }
36
37     lastAccelerometerFiltered = accelerometerFiltered
38     return step
39 }
```

Se establece la longitud del paso con la función `stepLengthValue` (apartado 3.3.2). Cuando se ha producido un evento (un paso), se establece un valor de $L = 58$ cm. En caso contrario, se establece una longitud de paso de $L = 0$ cm, es decir, que el usuario no se ha movido.

Código 3.12: Función `stepLengthValue`

```
1 private var stepLength : Float = 0.58F // Longitud del paso en metros
```

```
2
3  /**
4   * Guarda el valor de la longitud del paso si se ha detectado un paso
5   *
6   * @param step variable de comprobacion de pasos
7   * @return 0 si no se ha detectado paso y stepLength si se ha detectado
8   */
9  private fun stepLengthValue(step: Boolean) : Float {
10     return if(step) { stepLength } else 0.0F
11 }
```

Se establece el valor del ángulo de orientación con la función `headingDetermination`. Como se detalla en el apartado 3.3.3, este ángulo coincide con el ψ obtenido en el algoritmo basado en AoA.

Código 3.13: Función `headingDetermination`

```
1  /**
2   * Determinacion del angulo de direccion del usuario con respecto del Norte
3   *
4   * @param orientationAngles vector de orientaciones del dispositivo
5   * @return valor del angulo de direccion
6   */
7  private fun headingDetermination(orientationAngles: FloatArray): Float {
8      return orientationAngles[0] // Radianes
9  }
```

Finalmente, se implementa la función que resuelve la ecuación de PDR (3.16) de forma iterativa.

Código 3.14: Función `pdrAlgorithm`

```
1  /**
2   * Algoritmo de PDR con ecuacion general:
3   *
4   * 1. Detectar pasos
5   * 2. Establecer la longitud del paso
6   * 3. Determinar el angulo de direccion
7   * 4. Resolver la ecuacion general
8   * 5. Guardar el estado actual
9   *
10  * @param accelerometerReading lecturas del acelerometro
11  * @param orientationAngles vector de orientaciones del dispositivo
12  * @return la posicion actual del peaton
13  */
14 private fun pdrAlgorithm(
15     accelerometerReading: FloatArray,
16     orientationAngles: FloatArray
17 ) : FloatArray {
18
19     // Condicion de restablecimiento de la posicion incial
20     if(reset){
21         position[0] = 0F
22         position[1] = 0F
23         stepLength = 0F
24         oldPositionX = 0F
25         oldPositionY = 0F
26         step = false
27         displayCurrentValues() // valores reinicializados
```

```
28      }
29
30      // Obtener los parametros de la ecuacion
31      step = stepDetection(accelerometerReading)  // Deteccion de pasos
32      stepLength = stepLengthValue(step)      // Dar longitud del paso
33      angle = headingDetermination(orientationAngles)  // Determinar angulo de direccion
34
35      // Resolver la ecuacion general
36      position[0] = oldPositionX + stepLength * sin(angle)
37      position[1] = oldPositionY + stepLength * cos(angle)
38
39      // Guardar el estado actual
40      oldPositionX = position[0] // X_k-1 = X_k
41      oldPositionY = position[1] // Y_k-1 = Y_k
42
43      return position
44  }
```

Capítulo 4

Resultados

Una vez se han llevado a cabo las implementaciones descritas en el Capítulo 3, se plantea un conjunto de pruebas y análisis de la funcionalidad del algoritmo.

El método de comprobación aplicado en este proyecto se basa en dos fases: configuración de las medidas y resultados.

1. Configuración de las medidas: se plantea un escenario de *ground truth*[1] y un sistema de medidas desde el dispositivo móvil. A esta configuración también se le denomina *setup*.
2. Resultados: Una vez realizado el *setup* de medidas, se obtienen y representan los resultados. Esta fase sirve para describir los resultados obtenidos desde un punto de vista puramente objetivo.

En el apartado 4.1 se obtienen los resultados de las correcciones en la posición de objetos detectados para medidas de posicionamiento basadas en AoA. En el apartado 4.2 se obtienen los resultados de las estimaciones de la posición del usuario mediante PDR. Finalmente, en el apartado 4.3 se obtienen los resultados de la fusión de ambos algoritmos para manifestar las diferencias y mejoras frente al uso de un solo algoritmo.

[1]La verdad fundamental o *ground truth* es la información que se sabe que es real o verdadera, proporcionada por observaciones y medidas directas, es decir, por pruebas empíricas.

4.1. Resultados obtenidos en las correcciones en la posición de objetos detectados para medidas de posicionamiento basadas en AoA

En este apartado se realizan pruebas para determinar la funcionalidad del algoritmo basado en AoA. En este caso, se obtiene la precisión del algoritmo en función del error cometido en la distancia con respecto del *ground truth* y el área de cobertura del foco.

4.1.1. Configuración de las medidas

Medidas estáticas con la estación total y el receptor

El *setup* para estas pruebas consiste en realizar medidas altamente precisas y asumir que estas son el *ground truth*. Esta exactitud se consigue mediante el uso de una estación total (Figura 4.1(a)) y un prisma de seguimiento (Figura 4.1(b)), ambos de la firma Leica. El funcionamiento de estos dispositivos se basa en obtener un punto de referencia en el espacio y capturar puntos de medida (con el uso del prisma o con un láser) que están referenciados respecto de ese punto.

(a) Estación Total Leica MS60 (b) Prisma Leica GRZ101

Figura 4.1: Instrumentación de medidas precisas basadas en una estación total

El uso de estos dispositivos para medidas tiene unas prestaciones del orden de uno a dos milímetros en rangos de distancia de 1,5 a 10000 m, según especifica el fabricante (hoja de características en el Anexo A). Esta precisión es notablemente superior a la que cabe esperar en la aplicación de este proyecto, que se espera del orden de centímetros (a partir de los trabajos

relacionados en el estado del arte, capítulo 2). Por este motivo, se utilizan los resultados de dicha instrumentación como *ground truth* del proyecto en las medidas.

En el caso que ocupa, los pasos que se siguen para configurar el escenario de pruebas son los descritos a continuación:

1. Se obtienen las coordenadas de la estación total en un sistema de referencia elegido. En este caso, se referencia con respecto del Norte de la Tierra con el fin de realizar medidas en el espacio (x, y, z).
2. Se miden las coordenadas tridimensionales en el sistema de coordenadas elegido del centro del foco.
3. Se coloca el prisma en el receptor (dispositivo móvil).
4. Se mide la posición del receptor en estático:

   En este conjunto de medidas se va a demostrar el funcionamiento del algoritmo ante distintas variaciones en los ángulos de posición del dispositivo. La finalidad de esta prueba es obtener los errores producidos por el algoritmo comparándolos con los valores del *ground truth*.

   Para esta prueba, se sitúa el dispositivo móvil en una posición inicial conocida (medida por la estación total) bajo el foco inteligente. De esta manera, se permite tener un máximo rango de movimiento en el receptor. Se realizan barridos en los tres ángulos de Euler:

   - Ángulo de orientación cada 90°, en el rango $[-90°, 180°]$. Se selecciona este rango con el fin de cubrir todas las posibles orientaciones.
   - Ángulo de elevación cada 4°, en el rango $[0°, 12°]$. Se selecciona este rango de valores debido a la naturaleza de la posición de la mano de los usuarios, que habitualmente elevan la pantalla del dispositivo hacia arriba.
   - Ángulo de inclinación cada 4°, en el rango $[-8°, 8°]$. Se selecciona este rango de valores debido a la naturaleza de la posición de la mano de los usuarios, que en este caso asume que se sostiene con la mano derecha, la izquierda o ambas. El movimiento natural consiste en inclinar el dispositivo hacia dentro según la mano con la que se sostiene.

   El conjunto de todas estas capturas da lugar a un total de $4 \times 4 \times 5 = 80$ medidas.

Medidas para detectar el área de cobertura.

En este apartado se determina cuál es el área de cobertura o sombra del dispositivo móvil. Este área depende de diversos factores como el ángulo de visión del dispositivo (determinado por el sistema de lentes), la altura a la que se sostiene el dispositivo, la altura a la que se coloca el emisor o los ángulos de posición del dispositivo, entre otros. Con el fin de simplificar

las medidas y a partir de la finalidad de esta aplicación, este área se define de forma empírica mediante las posiciones más alejadas (en el plano $x-y$) en las que se detecta el foco inteligente. Se coloca el dispositivo móvil coplanario al suelo y se toman capturas en continuo de las posiciones detectadas.

4.1.2. Resultados de las correcciones basadas en AoA

Medidas estáticas con la estación total y el receptor

Los resultados obtenidos en la prueba realizada en estático para los diferentes ángulos de posición se muestran en la Figura 4.2. Se han ilustrado los diferentes ángulos de orientación establecidos en las medidas $(-90°, 0°, 90°, 180°)$. En azul y con una marca circular "o", se representan las capturas realizadas con la estación total (*ground truth*), mientras que en rojo y con una marca "x"se representan las capturas realizadas por el dispositivo. Además, el foco se dibuja como una circunferencia amarilla, de manera que se ilustre el punto de referencia del emisor.

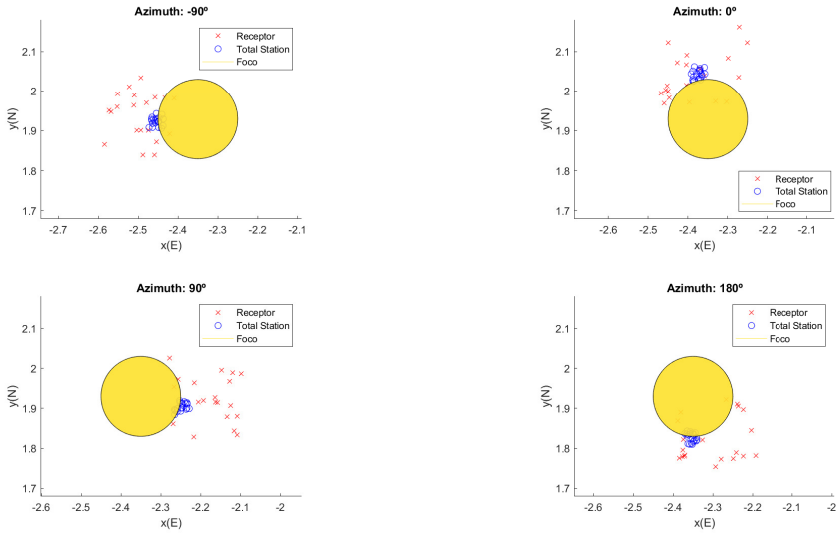

Figura 4.2: Medidas estáticas con variaciones en los ángulos de posición

Como se puede ver gráficamente, existe una dispersión en los valores de posición obtenidos al aplicar el algoritmo de corrección frente a la estabilidad de la nube de puntos formada por las posiciones del *ground truth*. El error producido por las correcciones se calcula a partir de esta dispersión. Para obtener el error, se obtiene la distancia euclídea para cada uno de los puntos con respecto del valor obtenido para la misma posición por la estación total como:

$$d_{ts,rx} = \sqrt{(x_{ts} - x_{rx})^2 + (y_{ts} - y_{rx})^2} \tag{4.1}$$

donde $d_{ts,rx}$ es la distancia euclídea entre la coordenada del receptor rx y la estación total ts, (x_{rx}, y_{rx}) son las coordenadas del receptor y (x_{ts}, y_{ts}) son las coordenadas de la estación total. En este caso, la distancia se mide únicamente sobre $x - y$ dado que la altura se considera conocida y constante (como se detalla en el apartado 3.2.1).

En las Tablas 4.1, 4.2 y 4.3 se recoge el análisis estadístico de los resultados obtenidos sobre el ángulo de orientación, inclinación y elevación, respectivamente. En concreto, se obtienen el valor máximo, el valor medio, la varianza y la desviación típica sobre la distancia (ecuación 4.1).

Azimut (°)	**max($\mathbf{E_d}$) (m)**	**$\bar{\mathbf{E}}_\mathbf{d}$ (m)**	**$\sigma^2(\mathbf{E_d})$ (m)**	**$\sigma(\mathbf{E_d})$ (m)**
-90	0,1389	0,0522	0,0895	0,0299
0	0,1798	0,0854	0,1625	0,0403
90	0,1727	0,1064	0,1899	0,0436
180	0,1830	0,0935	0,2567	0,0506

Tabla 4.1: Resultados en el algoritmo de corrección basado en AoA para la orientación (*azimut*)

Roll (°)	**max($\mathbf{E_d}$) (m)**	**$\bar{\mathbf{E}}_\mathbf{d}$ (m)**	**$\sigma^2(\mathbf{E_d})$ (m)**	**$\sigma(\mathbf{E_d})$ (m)**
-8	0,1407	0,0896	0,1388	0,0372
-4	0,1726	0,0859	0,1343	0,0366
0	0,1727	0,1020	0,1701	0,0412
4	0,1566	0,0828	0,1926	0,0439
8	0,1830	0,1030	0,2522	0,0502

Tabla 4.2: Resultados en el algoritmo de corrección basado en AoA para la inclinación (*roll*)

Pitch (°)	**max($\mathbf{E_d}$) (m)**	**$\bar{\mathbf{E}}_\mathbf{d}$ (m)**	**$\sigma^2(\mathbf{E_d})$ (m)**	**$\sigma(\mathbf{E_d})$ (m)**
0	0,1727	0,0801	0,1486	0,0385
4	0,1608	0,1025	0,1613	0,0402
8	0,1830	0,0953	0,2217	0,0471
12	0,1798	0,0927	0,1709	0,0413

Tabla 4.3: Resultados en el algoritmo de corrección basado en AoA para la elevación (*pitch*)

A la vista de los resultados obtenidos en las tres tablas, se observa que los errores máximos cometidos en los tres ángulos posibles (orientación, inclinación y elevación) son de valores similares, menores a 20 cm.

Medidas para detectar el área de cobertura

Las medidas realizadas en este escenario sirven como método para definir la cobertura que es capaz de ofrecer cada uno de los focos inteligentes que, tal y como se describe en la configuración, depende de diversos parámetros y variaciones. En esta prueba se ha caminado en varias direcciones bajo el foco, con el fin de encontrar aquellos puntos en los que este no se detecta.

En la Figura 4.3 se muestra el conjunto de puntos obtenidos por el dispositivo cuando capta el foco marcado en rojo con una "x", el foco inteligente en amarillo y el área de cobertura o sombra en gris, que tiene un radio de 1,3 m. Se representa una circunferencia para el área de cobertura para facilitar la geometría de la cobertura real dadas las múltiples variables que intervienen en su forma.

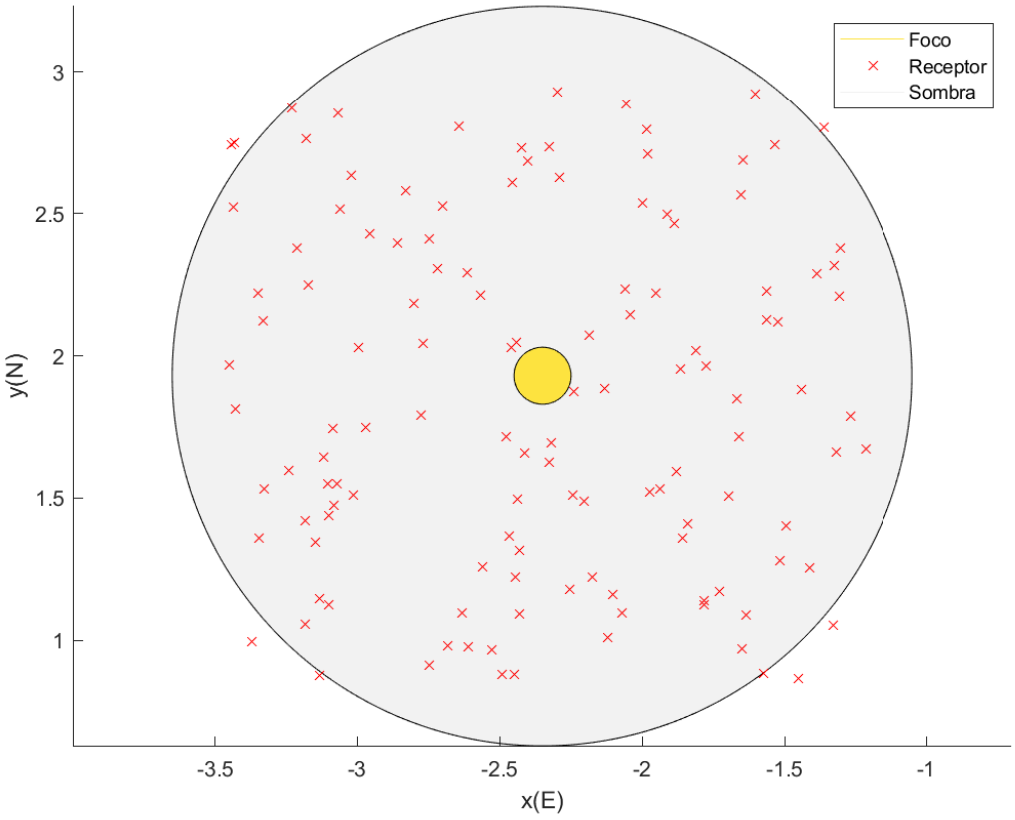

Figura 4.3: Área de cobertura de la detección del foco inteligente

4.2. Resultados obtenidos en las estimaciones de la posición del usuario mediante PDR

Este apartado tiene como objetivo demostrar la fiabilidad de las estimaciones de la posición basadas en el algoritmo de PDR desarrollado en el apartado 3.3 de este TFM.

4.2.1. Configuración para las medidas

El *setup* propuesto para la obtención de los resultados de este algoritmo se basa en establecer un camino (trayectoria) a seguir previamente determinado que se compara con los datos capturados por el dispositivo. En este caso, se establecen dos escenarios posibles: recta y paralelogramo.

Medidas para la trayectoria de una recta

Para este escenario se propone realizar una trayectoria en línea recta desde un punto de origen ($Inicio$) hasta un punto final (Fin), cuya distancia es conocida y expresada en metros. Se propone realizar el experimento con una distancia de 5 m varias veces. En la recta, se cuenta con marcas de referencia que sirven para determinar los errores cometidos en las distancias por el algoritmo, siendo este el *ground truth*.

Como el algoritmo de PDR se basa en desplazamientos que tienen en cuenta la orientación del dispositivo con respecto del Norte, el *ground truth* se representa en función de dicha orientación.

Medidas para la trayectoria de un paralelogramo

Para este escenario se propone realizar una trayectoria que siga a un paralelogramo. La finalidad de esta prueba es obtener los errores producidos en cambios de orientación abruptos (alrededor de 90°).

En el contexto de los museos (proyecto GUIA) este escenario es el más común en recorridos basados en una ruta previamente definida.

4.2.2. Resultados de las estimaciones basadas en PDR

Medidas para la trayectoria de una recta

En la Figura 4.4 se obtiene, con líneas discontinuas y distintos colores, la trayectoria descrita en cada una de las iteraciones (trayecto desde el punto de *Inicio* hasta el punto de *Fin*). Además, la línea del *ground truth* tiene unas marcas representadas por una "x"que sirven para establecer una referencia en la determinación de los errores.

Si se toma como referencia dichas marcas y se compara con cada uno de los pasos dados en cada trayecto, se pueden obtener los errores del PDR en estimaciones en líneas rectas.

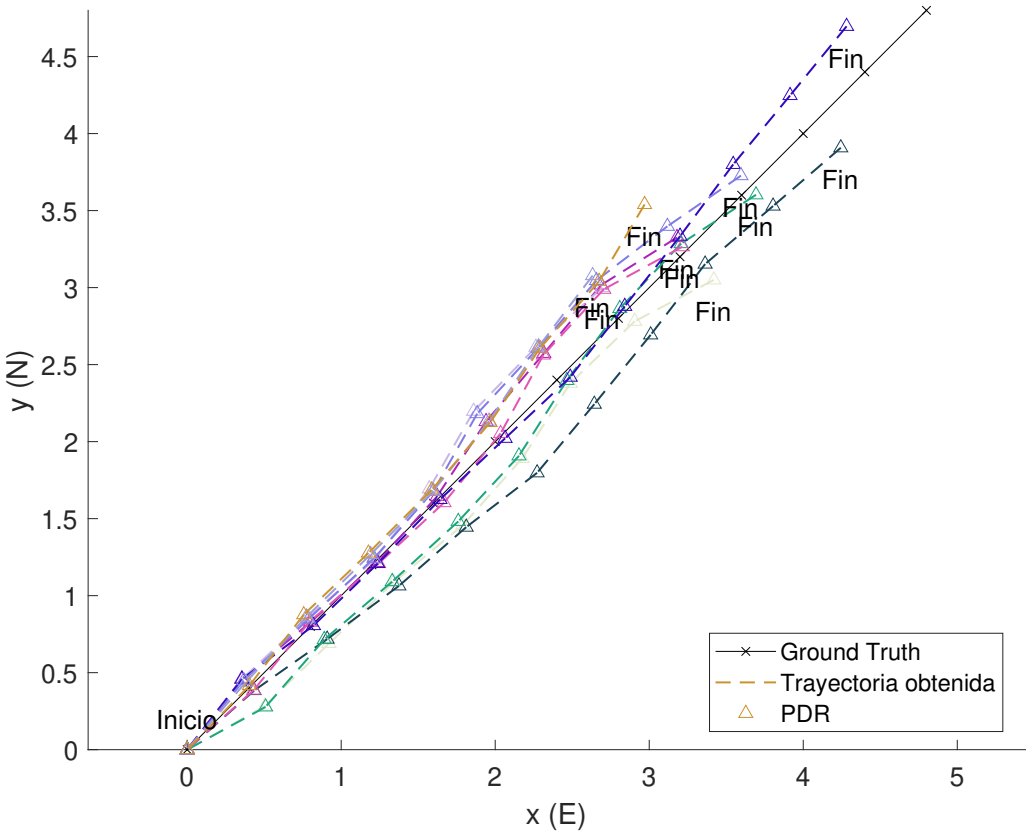

Figura 4.4: Resultados obtenidos en las pruebas de PDR sobre una recta de 5 m

En la Tabla 4.4 se muestran los errores máximos, la media, la varianza y la desviación típica en términos de distancia, para cada trayecto. Como se observar en la columna de los errores máximos, el máximo error en las iteraciones realizadas es de 24 cm.

Trayecto	**max($\mathbf{E_d}$) (m)**	**$\bar{\mathbf{E}}_\mathbf{d}$ (m)**	**$\sigma^2(\mathbf{E_d})$ (m)**	**$\sigma(\mathbf{E_d})$ (m)**
1	0,1990	0,1287	0,0026	0,0507
2	0,2439	0,1115	0,0053	0,0726
3	0,1068	0,0784	0,0006	0,0238
4	0,1058	0,0745	0,0010	0,0313
5	0,1261	0,0828	0,0010	0,0321
6	0,1212	0,0847	0,0010	0,0311
7	0,1026	0,0708	0,0005	0,0222
8	0,1179	0,0893	0,0007	0,0266
9	0,0839	0,0568	0,0007	0,0258
10	0,1015	0,0815	0,0005	0,0222

Tabla 4.4: Errores por trayecto en el algoritmo de PDR para la prueba de la línea recta

Medidas para la trayectoria de un paralelogramo

En la Figura 4.5, se muestra la trayectoria obtenida por el algoritmo de PDR. Como se puede observar, cada vez que se da una vuelta la posición se desplaza con respecto del *ground truth*. Esto se debe a la existencia de errores acumulativos debidos a la falta de referencia de los sistemas basados en unidades inerciales, es decir, al no disponer de referencia alguna la trayectoria obtenida se va desviando de la prevista conforme se avanza.

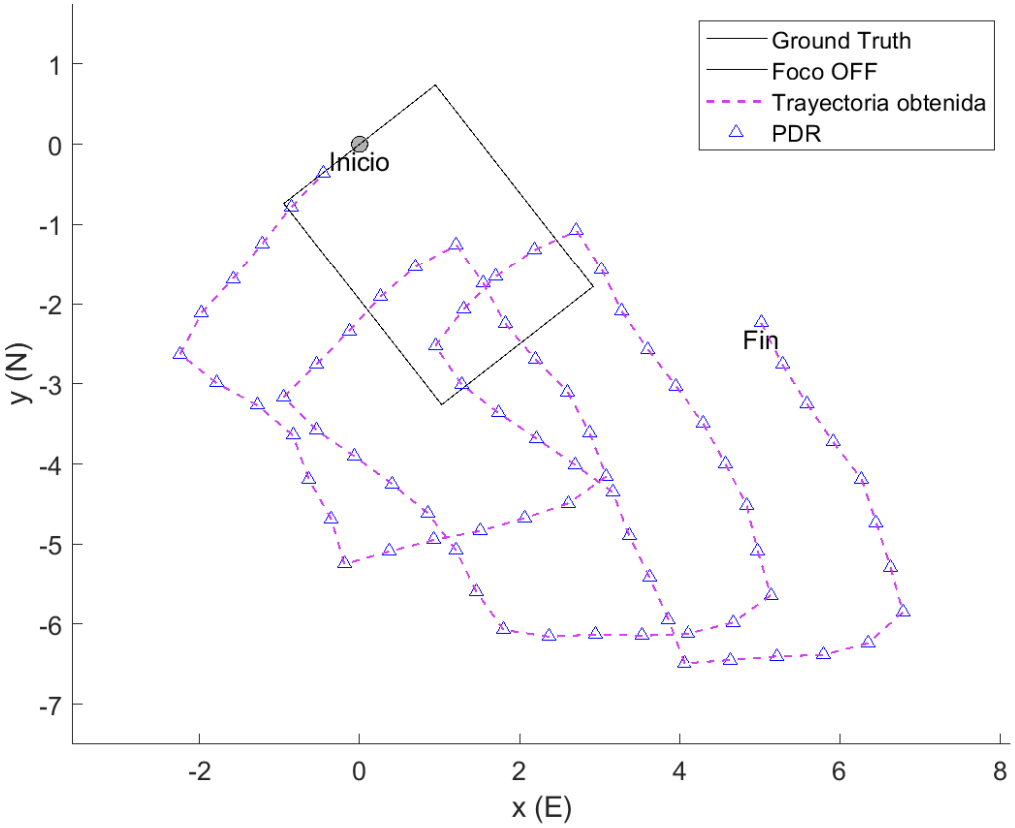

Figura 4.5: Resultados obtenidos en las pruebas de PDR sobre un paralelogramo

Si se establecen las cuatro esquinas del paralelogramo como puntos de referencia, se puede comparar cada una de las esquinas obtenidas por el algoritmo de PDR con el fin de obtener los errores acumulativos de cada una de las vueltas. En el siguiente cuadro se muestran los errores máximos, la media, la varianza y la desviación típica que se produce en cada esquina y por cada vuelta.

Vuelta	**max($\mathbf{E_d}$) (m)**	**$\bar{\mathbf{E}}_\mathbf{d}$ (m)**	**$\sigma^2(\mathbf{E_d})$ (m)**	**$\sigma(\mathbf{E_d})$ (m)**
1	2,3780	2,3478	0,014	0,0369
2	4,4528	3,0812	0,8801	0,9381
3	5,6130	4,4227	1,7074	1,3067

Tabla 4.5: Errores acumulativos en el algoritmo de PDR para la prueba del paralelogramo

4.3. Resultados de la fusión de los algoritmos propuestos

El objetivo de este apartado es representar los resultados obtenidos en un escenario real. En este escenario se fusionan los dos algoritmos desarrollados en la aplicación, donde la ejecución de cada algoritmo depende de los movimientos arbitrarios que realiza el usuario. Estos movimientos activan un algoritmo u otro y cada uno de ellos actúa en consecuencia.

Las pruebas evalúan la fusión de ambos algoritmos, de modo que se apaga o enciende la luminaria inteligente según se quieran o no aislar los algoritmos. En el caso de tener el foco encendido, actúan ambos algoritmos según el diagrama de la Figura 3.1. Si el foco está apagado, actúa únicamente el algoritmo de PDR, ya que el algoritmo de AoA solo actúa en el caso de que se detecte un foco.

4.3.1. Configuración para las medidas

El escenario para las pruebas comparte el *setup* con el configurado en los dos apartados anteriores (4.1 para AoA y 4.2 para PDR).

La trayectoria a seguir la forma un paralelogramo de 2,40 m de ancho y 3,20 m de largo. La luminaria inteligente se establece como posición inicial y se decide ubicar en el centro de uno de los lados cortos del paralelogramo. La orientación del paralelogramo tiene una rotación de 38° respecto del Norte de la Tierra. La Figura 4.6 ilustra el escenario de pruebas que se toma como *ground truth* cuando el foco está apagado, mientras que la Figura 4.7 ilustra cuando el foco está encendido (con la sombra de detección acorde con los resultados obtenidos en el apartado 4.1).

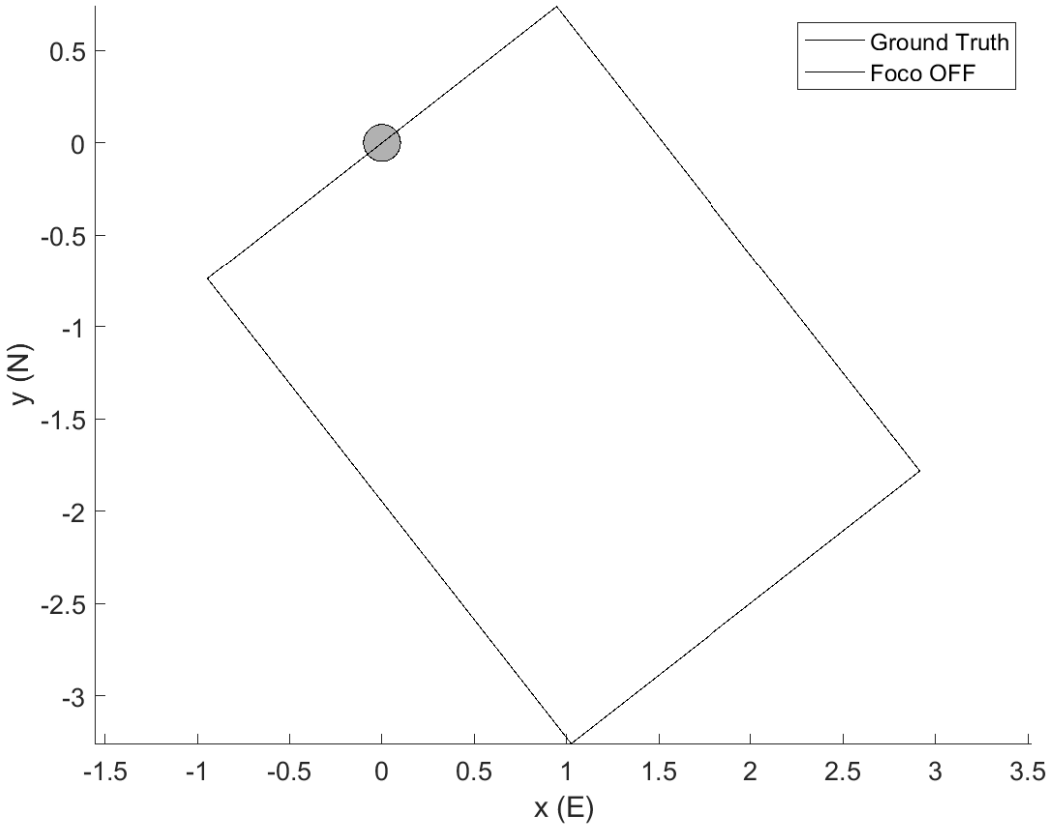

Figura 4.6: *Ground Truth* del paralelogramo con el foco apagado

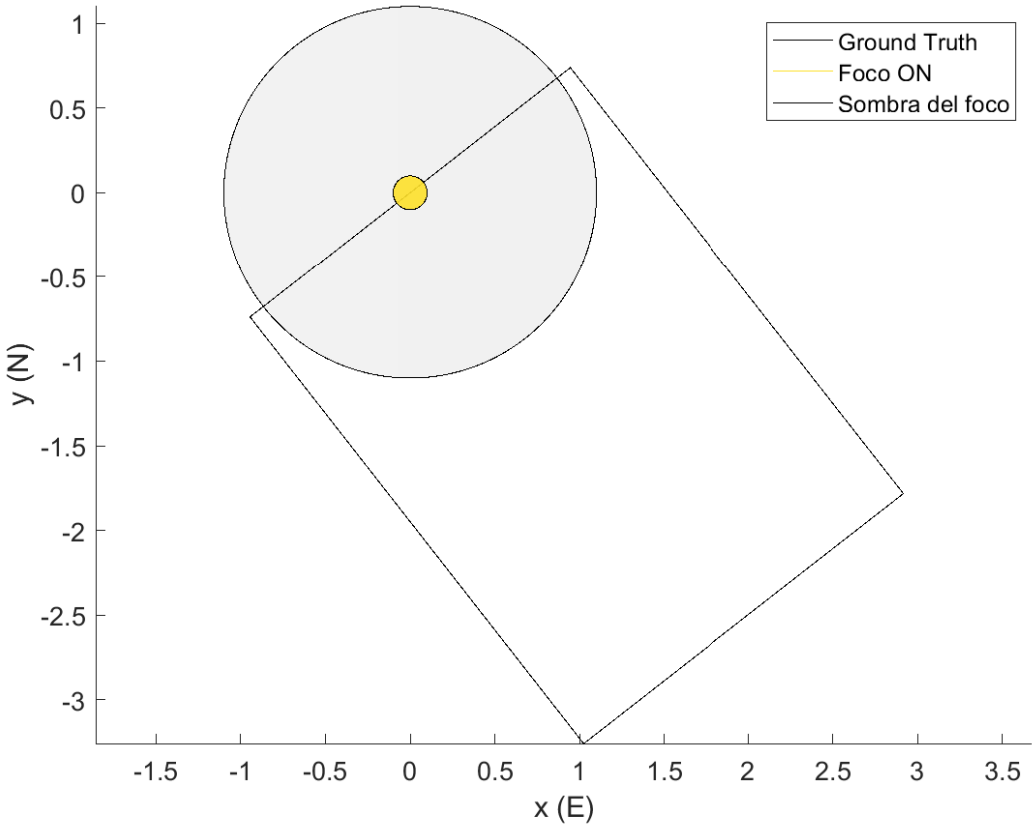

Figura 4.7: *Ground Truth* del paralelogramo con el foco encendido

4.3.2. Resultados de la fusión de los algoritmos

En este caso, se comparan los resultados cuando hay fusión de algoritmos. Estas prueba consiste en recorrer tres vueltas en el sentido horario sobre el *ground truth* detallado en la Figura 4.6. En primer lugar, se realiza una prueba con el foco apagado[2] y, a continuación, con el foco encendido.

En la Figura 4.8 se muestra el resultado obtenido cuando el foco está encendido. El gráfico se interpreta de la siguiente forma:

- El usuario comienza a caminar desde el punto *Inicio* siguiendo la trayectoria del paralelogramo (*ground truth*) y se dispone a dar tres vueltas con la aplicación abierta hasta llegar al punto final (*Fin*).

[2]Esta prueba es la misma que la realizada en el apartado 4.2. Los resultados de la prueba con el foco apagado se muestran en la Figura 4.5.

- Cuando el usuario está en el área de cobertura del reconocimiento del foco, el algoritmo que se utiliza es la corrección basada en AoA. Estos instantes se representan con el símbolo "+".
- Cuando el usuario está fuera del área de cobertura, el algoritmo que se utiliza es la estimación basada en PDR. Estos instantes se representan con un triángulo y cambia de color cada vez que se pasa por debajo del foco.
- En aquellos instantes que se cambia del algoritmo basado en PDR al algoritmo basado en AoA, se traza una línea roja que representa la corrección (en términos de distancia) de la posición entre ambos algoritmos. Es decir, como el algoritmo basado en PDR tiene errores acumulativos, cada vez que se reconoce el foco se corrige la posición a la obtenida por el algoritmo basado en AoA, que es más preciso.

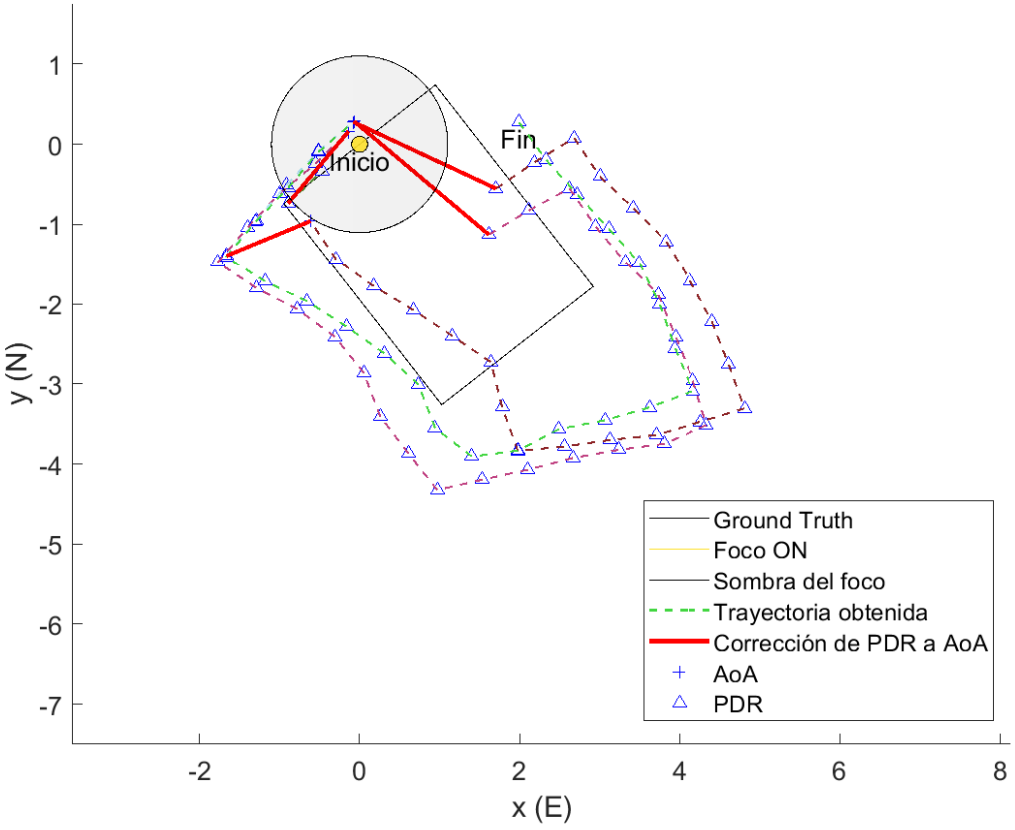

Figura 4.8: Resultados obtenidos en las pruebas de fusión de algoritmos en tres vueltas en el sentido horario con el foco encendido

Si se establecen las cuatro esquinas del paralelogramo como puntos de referencia, se puede comparar cada una de las esquinas obtenidas por el algoritmo de PDR fusionado con el algoritmo

de AoA. El objetivo de este análisis es comparar los resultados obtenidos al fusionar ambos algoritmos con los obtenidos únicamente con el algoritmo basado en PDR. En el siguiente cuadro se muestran los errores máximos, la media, la varianza y la desviación típica que se produce en cada esquina y por cada vuelta.

Vuelta	$\mathbf{max(E_d)}$ **(m)**	$\mathbf{\bar{E}_d}$ **(m)**	$\sigma^2(\mathbf{E_d})$ **(m)**	$\sigma(\mathbf{E_d})$ **(m)**
1	2,2401	1,6321	0,3978	0,6308
2	2,4330	1,4490	0,7819	0,8843
3	1,8047	1,1685	0,2055	0,4533

Tabla 4.6: Errores acumulativos en la fusión de algoritmos para la prueba del paralelogramo

Capítulo 5

Discusión

En este capítulo se discuten los resultados en el contexto del presente TFM. Se estructura la discusión de los resultados según la organización del Capítulo 4: resultados para el algoritmo basado en AoA, resultados para el algoritmo basado en PDR y resultados para la fusión de algoritmos.

- **Resultados para el algoritmo basado en AoA**

  En el estudio de estos resultados se han realizado pruebas para determinar la precisión de las correcciones del algoritmo basado en AoA y para determinar el área de cobertura de reconocimiento del foco inteligente.

  A la vista de los resultados (Tablas 4.1, 4.2 y 4.3), el error producido en estas correcciones da máximos menores a 20 cm en los tres ángulos de trabajo (orientación, inclinación y elevación), mientras que el valor medio es menor a 10 cm. Además, en términos de desviación típica del error de la distancia, se dan valores menores a 5 cm, lo que significa que el error obtenido no es muy disperso entre los diversos ángulos de medida. Estos valores se obtienen en comparación al *ground truth* establecido con la estación total, que es capaz de posicionar objetos con errores de milímetros.

  En el contexto del proyecto, en el que el usuario está en movimiento en un museo, se entiende que estos valores de error, del orden de 20 cm cuando se capta el foco y de 2 m cuando se estima la posición entre focos, son aceptables. El propio usuario no es capaz de percibir estos errores en la aplicación, ya que en esta se representa un mapa georreferenciado con la posición actual y la escala de visualización es 1000 veces inferior a la realidad (la escala de reducción habitual es de 10 m a 1 cm). Por tanto, en aquellos momentos en los que la aplicación trabaja bajo el algoritmo basado en AoA, se puede asumir que el posicionamiento es preciso.

  Por otro lado, el área de cobertura que se ha obtenido en las pruebas de laboratorio da un radio de 1,3 m con respecto del centro del foco (Figura 4.3). Cabe destacar que, tal y como se ha comentado (apartado 4.1), los resultados varían en función de la posición del dispositivo o la altura del foco. Así, se afirma que en los instantes en los que el usuario

está en el área de cobertura del foco, se obtiene dicho posicionamiento con una precisión de 20 cm, que en este contexto se considera altamente preciso.

- **Resultados para el algoritmo basado en PDR**

  En los estudios basados en PDR se han realizado pruebas para determinar los errores acumulativos debidos a la estimación basada en el algoritmo que se ha desarrollado. La acumulación de estos errores se debe a que cada vez que se actualiza la posición, la única referencia que se tiene es la posición anteriormente obtenida, de modo que todos los errores producidos en dicho instante anterior se acumulan en el actual, y así sucesivamente. Los errores se han determinado en dos situaciones distintas: caminar sobre una línea recta y caminar sobre un paralelogramo.

  En el caso de seguir la trayectoria en línea recta, los resultados presentan errores máximos menores a 25 cm, mientras que el valor medio es inferior a 13 cm. Además, la desviación típica del error de la distancia es menor de 7 cm. Estos resultados tienen órdenes similares a los obtenidos con el algoritmo basado en AoA, que se considera preciso.

  En el caso de seguir la trayectoria de un paralelogramo, los resultados obtenidos presentan errores notablemente mayores. En concreto, el error máximo es de 5,6 m, mientras que el valor medio es de 4,4 m. Cabe destacar que este error máximo se debe a la propia acumulación previamente descrita, que en este caso es incremental desde la primera vuelta hasta la tercera. No obstante, a la vista de la Figura 4.5, se entiende que el algoritmo funciona como era de esperar, ya que cada vez que se produce un cambio en el ángulo de orientación de 90°, la trayectoria cambia acorde a este valor. El parámetro que hace que la acumulación sea mayor es la longitud del paso (apartado 3.3.2), que es constante: a mayor número de pasos, mayor es el error acumulado por la diferencia entre la longitud del paso real y el valor establecido.

  En el contexto del proyecto, se asume que los resultados obtenidos en este algoritmo son válidos, ya que el objetivo de este TFM es combinar ambos algoritmos (como se estudia en el siguiente apartado).

- **Resultados para la fusión de algoritmos**

  En el estudio de estos resultados se han realizado pruebas para determinar las mejoras debidas a la fusión de los dos algoritmos. La prueba compromete al peor de los casos, que como se ha visto consiste en dar tres vueltas sin parar entorno a un paralelogramo.

  La Tabla 4.6 recoge estos resultados de la misma forma que en el caso anterior. En esta ocasión, el error máximo en términos de distancia, con respecto de las esquinas del paralelogramo, es de 2,4 m. El valor medio, por su parte desciende hasta 1,6 m. con desviaciones típicas de 0,8 m. Estos valores, comparados con los recogidos en la Tabla 4.5, manifiestan las mejoras al fusionar estos algoritmos, ya que hay una diferencia de 3 m entre el error máximo de ambos algoritmos, con la particularidad de que no existen errores acumulativos. En definitiva, cada vez que el usuario da una vuelta y pasa por debajo del foco la posición anterior del algoritmo de PDR se actualiza con la del algoritmo de AoA, de tal forma que se produce una corrección sobre los errores acumulativos de las estimaciones.

En el contexto del proyecto GUIA, la mayor parte de las situaciones se corresponden a esta combinación entre algoritmos. Como se ha detallado en el apartado 1.1, se va a desplegar una red de focos inteligentes en el museo. En conclusión, la combinación de ambos algoritmos es válida para contribuir y mejorar la aplicación desarrollada en el proyecto, ya que se cumplen los objetivos previstos: corregir la posición detectada en función de la orientación del dispositivo (objetivo 1) y estimar la posición del usuario cuando no se capta un foco inteligente (objetivo 2).

Capítulo 6

Conclusiones

En este capítulo se recogen las conclusiones finales sobre el desarrollo del proyecto, con las lecciones aprendidas y el cumplimiento de los objetivos establecidos en el presente TFM (apartado 6.1); las posibles líneas de investigación que se pueden explorar, ya sea en el ámbito del proyecto GUIA como en el resto de sistemas basados en VLP (apartado 6.2); y el seguimiento de la planificación establecida en el Diagrama de Gantt (Figura 1.1), a la vista la consecución de los hitos y objetivos planteados (apartado 6.3).

6.1. Conclusiones

A continuación, se listan las conclusiones del presente TFM:

- La implementación de ambos algoritmos cumple con los objetivos definidos en el apartado 1.2. A la vista de los resultados obtenidos (Capítulo 4), se manifiesta y garantiza que la contribución de ambos algoritmos en el desarrollo de la aplicación de posicionamiento del proyecto GUIA aporta un impacto positivo en la finalidad de la aplicación (posicionar y guiar a los usuarios en el museo).
- La fusión de algoritmos de posicionamiento mejora la precisión y calidad de los resultados obtenidos en términos de distancias, ya que proporciona estabilidad en los errores y mitiga alguno de ellos (como los errores acumulativos en las estimaciones basadas en PDR). Esto se comprueba en las comparativas realizadas en el Capítulo 5.
- Los resultados obtenidos son válidos en el contexto del proyecto GUIA. El error, en términos de distancia, que es del orden de 20 cm en aquellos instantes en los que se utiliza el algoritmo basado en AoA, y de 2,4 m en los que se utiliza el algoritmo basado en PDR (bajo la condición de fusionar ambos algoritmos), tiene unas prestaciones asumibles como precisas en el contexto de los museos.

En las aplicaciones de guiado y posicionamiento, el usuario no es capaz de percibir estos errores en la visualización de su posición actual. Como se ha comentado en el Capítulo 5, estos datos tienen una escala de reducción habitual de 1:1000. Aunque se puede cambiar esta escala, los valores de trabajo son siempre muy inferiores a los de la realidad.

- La contribución realizada con la fusión de algoritmos se puede extrapolar a aplicaciones de índole similar a este proyecto, en las que interviene el uso de los sensores inerciales de dispositivos móviles. Esto se detalla en el apartado 6.2 de este Capítulo.

6.2. Líneas de futuro

A continuación, se listan las posibles vías de investigación futuras:

- Se pueden introducir mejoras en los algoritmos desarrollados:

  Un claro ejemplo es el algoritmo basado en PDR, que considera la longitud del paso constante, el umbral de detección definido para todos los usuarios y el ángulo de orientación en base al estado actual. Gracias a profundizar sobre el análisis de las señales obtenidas en el acelerómetro, se puede diseñar un algoritmo que actualice estas tres variables de forma dinámica. Esto puede disminuir los errores acumulativos que se han obtenido en los resultados, con una notable mejora en la precisión del sistema de posicionamiento.

  Por otro lado, en el algoritmo basado en AoA se asume que el dispositivo tiene una altura constante con respecto del suelo. En ese sentido, se puede aprovechar el resto de sensores para obtener esta altura. Así, la ecuación que define el modelo matemático basado en *pinhole* tiene datos más realistas de la posición del receptor.

- Se puede analizar como afecta la colocación de las luminarias en el entorno. Esta línea de investigación permite conocer cuales son las ubicaciones óptimas y el número de sensores necesarios en el despliegue de la red de focos inteligentes. Aunque en el presente TFM únicamente se exploran las mejoras en el posicionamiento del receptor, el hecho de colocar adecuadamente el emisor en el espacio puede suponer mejoras en el funcionamiento de la fusión de algoritmos.

- Se pueden añadir algoritmos de posicionamiento adicionales. Existen algoritmos complementarios, como los basados en filtrado de Kalman o aprovechar las señales GNSS para mejorar la precisión y fiabilidad del sistema.

- Se puede extrapolar el desarrollo de estos algoritmos a otras aplicaciones de posicionamiento VLP. Aunque el receptor de esta aplicación es un dispositivo móvil, las técnicas desarrolladas en el presente TFM son aplicables a otros receptores que se equipen con una IMU y un mecanismo de detección de focos inteligentes.

- Otra línea de investigación es generalizar esta aplicación a todos los dispositivos, por medio del estudio de los rangos de error válidos en función de las características de sus sensores.

Es decir, estudiar la viabilidad de esta aplicación en dispositivos con sensores inerciales de diferentes características. Adicionalmente, se puede implementar la aplicación en el sistema operativo iOS de Apple.

6.3. Seguimiento de la planificación

En este apartado se presenta la consecución de los hitos establecidos en el apartado 1.4. El siguiente cuadro recoge la fecha de fin prevista para cada uno de los hitos y la fecha real de su cumplimiento.

Actividad	ID	Fecha de inicio	Fecha de fin	Fecha de fin real
Comienzo de la asignatura	HT1	15-sep.-21	16-sep.-21	16-sep.-21
Objetivos del TFM y organización de tareas	PEC1	15-sep.-21	26-sep.-21	23-sep.-21
Resumen y propuesta del título	PEC1.1	15-sep.-21	17-sep.-21	17-sep.-21
Motivación, objetivos y enfoque	PEC1.2	17-sep.-21	22-sep.-21	17-sep.-21
Planificación del proyecto	PEC1.3	17-sep.-21	25-sep.-21	22-sep.-21
Índice preliminar de la memoria	PEC1.4	22-sep.-21	26-sep.-21	23-sep.-21
Estado del arte	PEC2	27-sep.-21	10-oct.-21	9-oct.-21
Contexto y búsqueda de otros trabajos	PEC2.1	27-sep.-21	30-sep.-21	30-sep.-21
Estudio del marco teórico	PEC2.2	29-sep.-21	5-oct.-21	5-oct.-21
Enfoque y descripción teórica	PEC2.3	5-oct.-21	10-oct.-21	9-oct.-21
Desarrollo y resultados obtenidos	PEC3	11-oct.-21	3-dic.-21	3-dic.-21
AoA: Obtención de los parámetros del dispositivo	PEC3.1	11-oct.-21	16-oct.-21	13-oct.-21
AoA: Modelado matemático basado en pinhole	PEC3.2	17-oct.-21	24-oct.-21	20-oct.-21
AoA: Implementación del algoritmo en Kotlin	PEC3.3	25-oct.-21	4-nov.-21	6-nov.-21
AoA: Análisis de comportamiento con y sin detección	PEC3.4	5-nov.-21	10-nov.-21	16-nov.-21
PDR: Detección de pasos, longitud de pasos y rumbo	PEC3.5	11-nov.-21	21-nov.-21	22-nov.-21
PDR: Análisis del comportamiento entre focos	PEC3.6	22-nov.-21	3-dic.-21	25-nov.-21
Estudio de resultados obtenidos con ambos algoritmos	PEC3.7	25-nov.-21	3-dic.-21	3-dic.-21
Elaboración final de la memoria	PEC4	4-dic.-21	27-dic.-21	27-dic.-21
Conclusiones	PEC4.1	4-dic.-21	7-dic.-21	17-dic.-21
Líneas futuras del proyecto y repercusión	PEC4.2	4-dic.-21	6-dic.-21	20-dic.-21
Revisión bibliografía	PEC4.3	7-dic.-21	12-dic.-21	23-dic.-21
Maquetado final	PEC4.4	13-dic.-21	27-dic.-21	26-dic.-21
Elaboración de la presentación	PEC5	28-dic.-21	5-ene.-22	5-ene.-22
Maquetado diapositivas	PEC5.1	28-dic.-21	30-dic.-21	30-dic.-21
Grabación y montaje de la lectura	PEC5.2	2-ene.-22	5-ene.-22	5-ene.-22
Defensa del TFM	HT2	10-ene.-22	11-ene.-22	13-ene.-22

Tabla 6.1: Seguimiento de las actividades del TFM

Como se muestra en la Tabla 6.1, los hitos correspondientes a las dos primeras PEC y a las dos últimas, se han finalizado con anticipación a la fecha prevista. Sin embargo, las actividades de la tercera entrega (PEC3) han tenido ciertos retrasos con respecto de la previsión por la familiarización con el IDE Android Studio y el lenguaje Kotlin, el tiempo de aprendizaje para utilizar la estación total de Leica y las limitaciones de acceso al laboratorio del grupo de investigación GEINTRA a los días laborales.

Siglas

AoA

Angle of Arrival 12, 22, 23, 31, 36, 44, 45, 47, 49, 50, 59, 62–65, 67, 68

DR

Dead Reckoning 24, 29

GEINTRA

Grupo de Ingeniería Electrónica Aplicada a Espacios Inteligentes y Transporte 11, 69

GNSS

Global Navigation Satellite System 11, 19, 29, 68

GUI

Graphical User Interface 14

GUIA

Sistema de Guiado y Localización en Interiores usando la Iluminación de los Edificios 11, 13, 20, 21, 31, 38, 44, 55, 66, 67

IDE

Integrated Development Environment 14, 69

IMU

Inertial Measurement Unit 12, 13, 20, 21, 25, 26, 29, 68

IPS

Indoor Positioning Systems 11, 12, 19

LIT

Lineal e Invariante en el Tiempo 40

LOS

Line Of Sight 22

MEMS

Micro Electro-Mechanical Systems 25, 28, 29

NDK

Native Development Kit 14

PDR

Pedestrian Dead Reckoning 13, 16, 21, 29, 31, 38, 40, 43–45, 47, 49, 55–59, 62–65, 67, 68

PEC

Prueba de Evaluación Continua 14, 69

PSD

Position Sensitive Device 19, 20

RSS

Received Signal Strength 22, 23

RTLS

Real-Time Locating Systems 19

TDoA

Time Difference of Arrival 22, 23

TFM

Trabajo Final de Máster 9, 11–17, 19–21, 31, 32, 55, 64, 65, 67–69

ToA/ToF

Time of Arrival/Flight 22

VLC

Visible Light Communication 19

VLP

Visible Light Positioning 3, 4, 12, 19, 20, 67, 68

XML

Extensible Markup Language 14

Bibliografía

[Android, 2019] Android (2019). Android developers: Descripción general de sensores. https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es-419.

[Ashraf et al., 2020] Ashraf, I., Hur, S., and Park, Y. (2020). Smartphone sensor based indoor positioning: current status, opportunities, and future challenges. *Electronics*, 9(6):891.

[Collaboration et al., 2016] Collaboration, N. R. F. et al. (2016). A century of trends in adult human height. *Elife*, 5:e13410.

[Conesa et al., 2018] Conesa, J., Pérez-Navarro, A., Torres-Sospedra, J., and Montoliu, R. (2018). *Geographical and Fingerprinting Data for Positioning and Navigation Systems: Challenges, Experiences and Technology Roadmap*. Academic Press.

[De-La-Llana-Calvo et al., 2020] De-La-Llana-Calvo, Á., Salido-Monzú, D., Lázaro-Galilea, J.-L., Gardel-Vicente, A., Bravo-Muñoz, I., and Rubiano-Muriel, B. (2020). Accuracy and precision assessment of aoa-based indoor positioning systems using infrastructure lighting and a position-sensitive detector. *Sensors*, 20(18):5359.

[Do and Yoo, 2016] Do, T.-H. and Yoo, M. (2016). An in-depth survey of visible light communication based positioning systems. *Sensors*, 16(5):678.

[El-Sheimy and Youssef, 2020] El-Sheimy, N. and Youssef, A. (2020). Inertial sensors technologies for navigation applications: State of the art and future trends. *Satellite Navigation*, 1(1):1–21.

[Guo and Uradzinski, 2018] Guo, H. and Uradzinski, M. (2018). The usability of mti imu sensor data in pdr indoor positioning. In *2018 25th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*, pages 1–4.

[ISO/IEC, 2014] ISO/IEC (2014). Iso/iec 24730-1:2014, information technology - real-time locating systems (rtls) - part 1: Application programming interface (api). https://www.iso.org/obp/ui/#iso:std:59801:en.

[Jabborov and Cho, 2020] Jabborov, F. and Cho, J. (2020). Image-based camera localization algorithm for smartphone cameras based on reference objects. *Wireless Personal Communications*, 114(3).

[Jimenez et al., 2009] Jimenez, A. R., Seco, F., Prieto, C., and Guevara, J. (2009). A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu. In *2009 IEEE International Symposium on Intelligent Signal Processing*, pages 37–42. IEEE.

[Li and Ning, 2018] Li, Y.-S. and Ning, F.-S. (2018). Low-cost indoor positioning application based on map assistance and mobile phone sensors. *Sensors*, 18(12):4285.

[Li, 2016] Li, Z. (2016). *Indoor Positioning System Using Visible Light Communication and Smartphone With Rolling Shutter Camera*. PhD thesis, University of California, Riverside.

[Luo et al., 2017] Luo, J., Fan, L., and Li, H. (2017). Indoor positioning systems based on visible light communication: State of the art. *IEEE Communications Surveys & Tutorials*, 19(4):2871–2893.

[Mautz, 2012] Mautz, R. (2012). *Indoor positioning technologies*. PhD thesis, ETH Zurich, Department of Civil, Environmental and Geomatic Engineering.

[Rodríguez-Navarro et al., 2017] Rodríguez-Navarro, D., Lázaro-Galilea, J. L., De-La-Llana-Calvo, Á., Bravo-Muñoz, I., Gardel-Vicente, A., Tsirigotis, G., and Iglesias-Miguel, J. (2017). Indoor positioning system based on a psd detector, precise positioning of agents in motion using aoa techniques. *Sensors*, 17(9):2124.

[Salido-Monzú, 2015] Salido-Monzú, D. (2015). *Infrared ranging in multipath environments for indoor localization of mobile targets*. PhD thesis, Universidad de Alcalá.

[Shala and Rodriguez, 2011] Shala, U. and Rodriguez, A. (2011). Indoor positioning using sensor-fusion in android devices.

[Thompson, 2002] Thompson, D. M. (2002). Stride analysis. https://ouhsc.edu/bserdac/dthompso/web/gait/knmatics/stride.htm.

[Travagnin, 2020] Travagnin, M. (2020). Cold atom interferometry for inertial navigation sensors. *Technology Assessment: space and Defence Applications*.

[Wang et al., 2018] Wang, B., Liu, X., Yu, B., Jia, R., and Gan, X. (2018). Pedestrian dead reckoning based on motion mode recognition using a smartphone. *Sensors*, 18(6):1811.

[Wang et al., 2019] Wang, Z., Yang, Z., Huang, Q., Yang, L., and Zhang, Q. (2019). Als-p: Light weight visible light positioning via ambient light sensor. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1306–1314. IEEE.

[Woodman, 2007] Woodman, O. J. (2007). An introduction to inertial navigation. Technical report, University of Cambridge, Computer Laboratory.

Apéndice A

Hoja de características de Leica MS60

Leica Nova MS60 MultiStation

MEDICIÓN ANGULAR		
Precisión [1] Hz y V	■ Absoluto, continuo, cuádruple	1" (0.3 mgon)
MEDICIÓN DE DISTANCIAS		
Rango [2] / Precisión / Tiempo de medición	■ Prisma (GPR1, GPH1P) [2,3,5] ■ Normal (a cualquier superficie) [2,4,5,6]	1,5 m a >10,000 m / 1 mm + 1,5 ppm / típico 1,5 s 1,5 m a 2.000 m / 2 mm + 2 ppm / típico 1,5 s
Tecnología de medición	Digitalizador de forma de onda	Láser rojo visible coaxial, tamaño de punto 8 mm × 20 mm a 50 m
ESCANEAR		
Velocidad de escaneo / Máxima velocidad de escaneo	30,000 Hz	30,000 puntos por segundo
Máx. alcance [7] / Ruido (1 sigma)	■ Modo 30 kHz ■ Modo 8 kHz ■ Modo 1 kHz ■ Modo 1 Hz	60 m / 3 mm a 50 m 150m / 1,5mm a 50 m 300m / 1,0mm a 50 m 1.000m / 0,6mm a 50 m
Datos de escaneo	Nube de puntos en 3D incluyendo color verdadero (RGB), intensidad y SNR	
Duración del escaneo	■ Escaneo de bóveda completa 400 gon × 155 gon ■ Escaneo de banda 400 gon × 50 gon	Resolución 50 mm a 15 m, duración 12 min Resolución 12,5 mm a 50 m, duración 45 min
IMÁGENES		
Cámara gran angular y coaxial	■ Resolución / Tasa de refresco ■ Campo de visión (gran angular / coaxial)	CMOS 5 MP / hasta 20 fps 19.4° / 1.5°
MOTORIZACIÓN		
Motores directos basados en Tecnología Piezoeléctrica	Velocidad de rotación / Tiempo en cambiar a CI	Maximo 400 gon (360°) por s / norm. 2.9 s
PUNTERÍA AUTOMÁTICA - ATRplus		
Alcance de la puntería a prisma [2] / Alcance de seguimiento a prisma [2]	■ Prisma Circular (GPR1, GPH1P) ■ Prisma 360° (GRZ4, GRZ122)	1,500 m / 1,000 m 1,000m / 1,000 m
Precisión [1,2] / Tiempo de medición	Precisión angular del ATRplus Hz, V	1" (0.3 mgon)/ normalmente 2.5 s
POWERSEARCH		
Alcance / Tiempo de Búsqueda	Prisma 360° (GRZ4, GRZ122)	300m / normalmente 5s
LUCES GUÍA DE REPLANTEO (EGL)		
Rango de Trabajo / Precisión		5-150m / normalmente 5cm @ 100m
GENERAL		
Sistema operativo / Software de campo	Windows EC7 / Leica Captivate y sus aplicaciones admiten decisiones in situ y en tiempo real	
Procesador	TI OMAP4430 1GHz Dual-core ARM® Cortex™- A9 MPCore™	
Autoenfoque de objetivo	Aumentos / Rango Enfoque	30 x / 1.7m a infinito
Modo Medida altura Automática	■ Precisión de distancia ■ Rango de distancia	1,0 mm (1 sigma) 0,7 m a 2,7 m
Pantalla y teclado	5", WVGA, color, táctil, en ambas posiciones	37 teclas, iluminadas
Manejo	3 tornillos sin fin, 1 tornillo para enfoque, 2 teclas de autoenfoque, SmartKey configurable	
Alimentación	Batería intercambiable Li-Ion	Hasta 9 h, capacidad de carga interna
Almacenamiento de datos	Memoria interna / Tarjeta de memoria	2 GB / Tarjeta SD 1 GB o 8 GB
Interfaces	RS232, USB, Bluetooth®, WLAN	
Peso	MultiStation incl. batería	7,7kg
Especificaciones ambientales	■ Rango de Temperatura de Trabajo ■ Polvo & Agua (IEC 60529) / Lluvia racheada ■ Humedad	–20°C a 50°C IP65 / MIL-STD-810G, métodos 506.5 I y 507.5 95%, sin condensación

[1] Desv. Estándar ISO 17123-3
[2] Cubierto, sin bruma, visibilidad aprox. 40km; sin reverberación
[3] 1.5m a 3.000m para prismas de 360° (GRZ4, GRZ122)
[4] Objeto en sombra, cielo nuboso, Tarjeta Gris Kodak (90% reflectividad)
[5] Desv. Estándar ISO 17123-4
[6] Distancia > 500m: Precisión 4mm + 2ppm, Tiempo de medición típico de 4s
[7] Objeto en sombra, cielo nublado, visibilidad ininterrumpida, prisma estático, Tarjeta Gris Kodak (90% reflectividad)

Radiación láser, evitar la exposición directa a los ojos.
Láser clase 3R de acuerdo con IEC 60825-1:2014.

Integración con LOC8 – Lock & Locate
Para obtener más información, visite **leica-geosystems.com/LOC8**

Leica Geosystems AG
Heinrich-Wild-Strasse
9435 Heerbrugg, Suiza
+41 71 727 31 31

- when it has to be **right**

Leica Geosystems