CENTRO INTERNACIONAL DE ESTUDOS
DE DOUTORAMENTO E AVANZADOS
DA USC (CIEDUS)

TESIS DE DOCTORADO

INDOOR POSITIONING FOR SMARTPHONES WITHOUT INFRASTRUCTURE AND USER ADAPTABLE

Presentada por:

Germán Rodríguez García

Dirigida por:

Roberto Iglesias Rodríguez
Adrián Canedo Rodríguez

ESCUELA DE DOCTORADO INTERNACIONAL PROGRAMA DE DOCTORADO EN INVESTIGACIÓN EN TECNOLOXÍAS DA INFORMACIÓN

SANTIAGO DE COMPOSTELA
Septiembre de 2019

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

DECLARACIÓN DEL AUTOR DE LA TESIS
INDOOR POSITIONING FOR SMARTPHONES WITHOUT INFRASTRUCTURE AND USER ADAPTABLE

Don Germán Rodríguez García

Presento mi tesis, siguiendo el procedimiento adecuado al Reglamento, y declaro que:

1. *La tesis abarca los resultados de la elaboración de mi trabajo.*
2. *En su caso, en la tesis se hace referencia a las colaboraciones que tuvo este trabajo.*
3. *La tesis es la versión definitiva presentada para su defensa y coincide con la versión enviada en formato electrónico.*
4. *Confirmo que la tesis no incurre en ningún tipo de plagio de otros autores ni de trabajos presentados por mí para la obtención de otros títulos.*

En Santiago de Compostela, Septiembre de 2019

Fdo. Germán Rodríguez García

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

AUTORIZACIÓN DEL DIRECTOR/TUTOR DE LA TESIS
INDOOR POSITIONING FOR SMARTPHONES WITHOUT INFRASTRUCTURE AND USER ADAPTABLE

Dr Roberto Iglesias Rodríguez,

Dr Adrián Canedo Rodríguez,

INFORMAN:

Que la presente tesis, corresponde con el trabajo realizado por ***Don Germán Rodríguez García*** *bajo nuestra dirección, y autorizamos su presentación, considerando que reúne los requisitos exigidos en el Reglamento de Estudios de Doctorado de la USC, y que como directores de ésta no incurre en las causas de abstención establecidas en Ley 40/2015.*

En Santiago de Compostela, Septiembre de 2019

Fdo. Roberto Iglesias Rodríguez
Director/a tesis

Fdo. Adrián Canedo Rodríguez
Director/a tesis

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Not all those who wander are lost.

J. R. R. Tolkien

We cannot know what we can do in advance. The only way to find out is to go all out trying, thinking only of success.

Royal Robbins

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Acknowledgments

Thanks to a) *Ministerio de Economía, Industria y Competitividad* (Spanish Government) in the *Industrial PhD* 2014 program. b) *Ministerio de Economía, Industria y Competitividad*, AEI/FEDER (European Union) grant TIN2017-90135-R.

To the *Centro Singular de Investigación en Tecnoloxías da Información* (CITIUS), to the *Departamento de Electrónica e Computación*, and to the *Grupo de Sistemas Intelixentes* of the University of Santiago de Compostela.

To Fernando Estevez Casado, for his assistance and colaboration.

To Situm Technologies and to all my colleagues throughout these years.

To Dr Adrián Canedo and Dr Roberto Iglesias the supervisors of this thesis for their help, guidance and support.

September 2019

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Resumen

Las tecnologías de Localización en Interiores han cobrado especial importancia en los últimos años. Las personas cada vez pasamos más tiempo en interiores, y estos entornos son cada vez más grandes y complejos. En lugares como centros comerciales, hospitales o aeropuertos es fácil perderse o perder tiempo buscando como llegar a un lugar en concreto. La localización en estos entornos tiene una gran variedad de aplicaciones, como guiado de personas, monitorización o localización de activos entre otras.

En exteriores existen desde hace años soluciones de posicionamiento y guiado basadas el uso de GPS (Global Positioning System) o GNSS (Global Navigation Satellite System). Estas soluciones no funcionan en interiores debido principalmente a que los materiales de los edificios interfieren con la señal de los satélites, atenuándolas, bloqueandolas por completo o bien provocando fenómenos como el *multipath* que provocan errores en la posición estimada.

Como respuesta a este problema, en las últimas décadas han ido surgiendo diferentes tecnologías de Localización en Interiores. Se han desarrollado soluciones utilizando diferentes tecnologías y sensores, como infrarojos, ultrasonidos, cámaras de video, etc. siendo las que más éxito ha tenido las basadas en radiofrecuencia. La mayoría de estas propuestas, tanto académicas como comerciales, utilizan dispositivos hardware dedicados, y requieren la instalación de emisores a lo largo del edificio. El sujeto a localizar debe llevar con él un dispositivo receptor para escuchar las señales emitidas. A partir de estas lecturas y utilizando conocimiento previo sobre la distribución de los emisores, se estima la ubicación del sujeto mediante técnicas como localización por proximidad, triangulación o *fingerprinting*.

Utilizar hardware dedicado es costoso, dificulta la escalabilidad del sistema y su aceptación por los usuarios finales. Un usuario será más reacio a utilizar un sistema de posicionamiento si tiene que comprar harware específico. Afortunadamente, la gran mayoría de las personas ya lleva consigo un teléfono que contiene todos los sensores necesarios para imple-

mentar con éxito las técnicas clásicas de localización en interiores, como tarjetas receptoras WiFi o BLE. Dado que prácticamente todos los edificios cuentan con una buena infraestructura de red WiFi, se pueden reutilizar para el posicionamiento y no es necesario instalar hardware adicional, con el importante abaratamiento de costes. Los teléfonos móviles actuales, además, incorporan nuevos sensores, como el magnetómetro, acelerómetro o giróscopo, que se pueden utilizar para estimar el desplazamiento del usuario y así ayudar a mejorar la robustez y precisión de los sistemas de posicionamiento.

A pesar de que ya se ha realizado investigación sobre el posicionamiento en interiores con smartphones, todavía no existe una solución óptima y existen problemas pendientes de resolver:

- Para conseguir una mayor precisión es necesario instalar hardware adicional. Muchas soluciones comerciales requieren la instalación de una gran cantidad de emisores, como por ejemplo beacons Bluetooth, en el edificio. Hasta cierto punto, a mayor densidad de emisores, mayor precisión. Por supuesto, esto también implica mayores costes de despliegue y mantenimiento.

- Además, la mejora de rendimiento que se puede obtener utilizando sólo observaciones basadas en un sensor está limitada por las características físicas de la propia tecnología (los infrarojos no atraviesan paredes, las cámaras necesitan línea directa de visión con el sujeto y les afectan las condiciones de luminosidad, la radiofrecuencia sufre de atenuación de señal, propagación multicamino,etc.), además del propio ruido del sensor. Estas limitaciones pueden superarse combinando la información de diferentes sensores.

- Utilizando los sensores inerciales del teléfono es posible estimar el desplazamiento del usuario. Combinando observaciones con el movimiento del sujeto, es posible relacionar temporal y espacialmente medidas secuenciales de los diferentes sensores para así mejorar la precisión y robustez de las posiciones estimadas. Obtener una estimación precisa del desplazamiento del usuario a partir de las señales recibidas por los sensores inerciales de un teléfono móvil no es trivial. Las personas utilizamos los teléfonos de maneras muy diferentes. Los llevamos en la mano, en el bolsillo, en una mochila, los usamos para hablar, etc. Esta variedad en el modo de uso del teléfono o de la posición en que se lleve afectan enormemente a la forma y características de las señales percibidas. Muchas soluciones funcionan restringiendo el modo de uso o posición del dispositivo, obligando a llevarlo en determinado lugar, como en la mano, en la cintura,

etc. Este tipo de restricción no es deseable ya que el usuario debería poder utilizar su teléfono libremente, incluso moviéndolo de sitio, sin que su uso afecte a la calidad del posicionamiento.

- Además, el usuario puede llevar a cabo diferentes actividades, como andar, subir escaleras, mover el teléfono, caminar, o desplazarse en un vehículo. Todas estas actividades afectarán de distinta forma a las señales percibidas por los sensores.

- En entornos interiores como parkings subterráneos, es frecuente que las personas se desplacen en vehículos. El posicionamiento en interiores en vehículos implica dificultades adicionales respecto al caso de caminar:

  - Estimar el desplazamiento del vehículo a partir de los sensores de un teléfono es complejo. Teóricamente, la velocidad se podría estimar integrando la señal del acelerómetro. Sin embargo las vibraciones, el ruido de los sensores de baja calidad de los teléfonos, los cambios de inclinación del terreno, hacen que esta solución no funcione en la práctica.
  - La velocidad de un vehículo es mayor que la de una persona caminando. Errores en la estiamción del movimiento implicarán un error de mayor distancia.
  - Generalmente, en parkings subterráneos, existen menos puntos de acceso WiFi y la cobertura es más limitada que en otro tipo de entornos.

En esta tesis abordamos el problema del posicionamiento en interiores con teléfonos móviles con el objetivo de maximizar la precisón de las posiciones estimadas mientras minimizamos la infraestructura necesaria. Además, para que el sistema sea útil para el usuario final, debe funcionar en situaciones reales. Esto implica que debe ser robusto a diferentes usuarios (cada usuario tendrá su forma de caminar), permitir el uso normal del teléfono (usarlo para hablar, llevarlo en la mano o en el bolsillo,etc.) y funcionar mientras se llevan a cabo diferentes actividades.

Para conseguir estos objetivos hemos abordado los siguientes hitos:

1. Hemos desarrollado un sistema de posicionamiento en interiores que calcula la posición del usuario fusionando la información que proviene de los diferentes sensores presentes en un teléfono móvil. Combinando la información de múltiples fuentes obtenemos mayor precisión y robustez de la que obtendríamos utilizando los distintos sensores por

separado. Este sistema combina observaciones sobre la posición global del usuario con la estimación de su desplazamiento y orientación. Este sistema es escalable, fácil de desplegar y es lo suficientemente flexible como para permitir añadir en el futuro nuevas fuentes de información sin dificultad. Además el sistema desarrollado se ejecuta en el propio teléfono en tiempo real.

2. Hemos desarrollado un módulo de estimación inercial capaz de identificar el movimiento del usuario. Hemos diferenciado dos casos principales, posicionamiento de peatones y de vehículos. Nuestra propuesta es suficientemente robusta como para soportar diferentes modos de uso, posiciones del teléfono y actividades. Para lograrlo hemos abordado los siguientes puntos:

   a) Hemos desarrollado un módulo para estimar la orientación del teléfono en el espacio utilizando los sensores inerciales del teléfono. Con el objetivo de obtener el desplazamiento y dirección del usuario primero necesitamos estimar la orientación del teléfono respecto al sistema de coordenadas de la Tierra. Utilizamos el sistema de coordenadas ENU (East North Up) definido por un eje vertical que apunta hacia arriba, y dos ejes tangenciales a la superficie terrestre que apuntan al norte y al este.

      Estimar la orientación del teléfono es lo mismo que estimar la rotación que define el cambio del sistema de coordenadas del teléfono respecto al sistema de coordenadas de la tierra. Para estimar esta rotación combinamos la información de los sensores inerciales del teléfono (acelerómetro y giróscopo). El giróscopo permite obtener una estimación precisa a corto plazo de los giros que realiza el teléfono, pero no proporciona información absoluta de su orientación y tiene cierto error, que al integrarse, se acumula rápidamente (drift). Por su parte, el acelerómetro, mide tanto las aceleraciones producidas por el movimiento del teléfono como la causada por la fuerza de gravedad. Esta información puede utilizarse para estimar la orientación del teléfono respecto al eje vertical. Combinando la información de ambos sensores se puede obtener una estimación de los cambios de orientación del dispositivo eliminando el drift en el eje vertical.

      El drift en los ejes Norte/Este se podría eliminar incluyendo la información del magnetómetro, pero en interiores son frecuentes las interferencias electromagnéticas causadas por maquinaria e instalaciones eléctricas, que pueden introducir un

error mayor que el propio drift del giróscopo. Es por esto que el drift en el eje Norte/Este, en nuestro caso, lo eliminaremos posteriormente al combinar la información del resto de sensores. Si bien, también emplearemos el magnetómetro de forma ocasional para mejorar la velocidad de inicialización del sistema.

b) Hemos evaluado y desarrollado diferentes técnicas para estimar el desplazamiento del usuario. Para esto hemos diferenciado dos modos principales de desplazamiento: andando o en un vehículo.

   i. Estimación de desplazamiento andando:
   Existe una amplia bibliografía sobre la estimación del desplazamiento de personas a partir de sensores inerciales, sin embargo, las soluciones existentes están lejos de ser ideales. Las propuestas clásicas asumen que los sensores se situan en una posición fija del cuerpo del usuario, como los pies o la cadera. Esto simplifica la tarea de estimación de distancia pues las aceleraciones a los que estaría sujeto el terminal estarían relacionadas directamente con el desplazamiento del usuario. Sin embargo, para el caso de posicionamiento con teléfonos móviles, no es razonable asumir que el teléfono va a estar en una posición fija.
   El enfoque habitual para calcular el desplazamiento de una persona caminando usando sensores inerciales, pasa por detectar los pasos dados y asignarle una estimacion de distancia a cada paso. Cuando una persona camina llevando un teléfono en la mano, y en absencia de otros estímulos, la señal percibida por el acelerómetro muestra un patrón reconocible, casi sinusoidal, en el que cada paso dado produce un pico seguido por un valle. Existe una amplia bibliografía respecto al reconocimiento y conteo de pasos, pero la experimentación solamente se centra en casos de personas caminando. Hemos observado que cuando el usuario camina, un simple detector Pico Valle es suficiente para obtener rendimientos elevados, pero el principal problema viene a la hora de diferenciar los pasos reales de movimientos del teléfono que no estén relacionados con caminar. O lo que es lo mismo, discriminar falsos positivos.
   Para solucionar este problema hemos adoptado y evaluado dos enfoques diferentes:

   A. Identificación de pasos basada en forma la forma de la señal. Hemos

desarrollado y evaluado un algoritmo de reconocimiento y conteo de pasos basado en la forma de la señal del acelerómetro producida cuando un usuario da un paso.

En este algoritmo, hay una fase previa de entrenamiento por parte del usuario, en la que se graban datos de pasos reales en diferentes posiciones y se genera uno o varios patrones de paso ideal. Posteriormente, se compara la señal del acelerómetro con estos patrones para identificar si se ha realizado un paso o no. Este algoritmo se adapta al usuario y son capaz de identificar pasos llevando el teléfono en diferentes lugares, utilizandolo para hablar, etc.

B. Identificación de pasos basada en características. En este otro caso construimos un clasificador a partir del conjunto de características más representativo extraído de la IMU (*Inertial Measurment Unit*) del teléfono, con el objetivo de identificar la actividad de caminar. Estas características pertenecen tanto al dominio del tiempo como al de la frecuencia.

Además de identificar los pasos dados por el usuario, para estimar la distancia total, hemos añadido una etapa de estimación de tamaño de paso. Según la bibliografía, el tamaño de paso depende principalmente de factores como la altura del usuario o la cadencia de paso. Hemos realizado experimentación al respecto, y finalmente hemos adaptado el tamaño de paso dinámicamente en función de la frecuencia de paso. A pesar de que conocer la altura del usuario sería útil para ajustar el tamaño de paso, descartamos utilizarla para evitar ser intrusivos con el usuario.

Hemos creado un extenso dataset para evaluar los algoritmos de conteo de pasos. Este dataset consta de las señales de los sensores inerciales con los pasos etiquetados con su timestamp uno a uno. Hemos hechos especial énfasis en realizar actividades tanto caminando como sin caminar, para poder evaluar correctamente la detección de falsos positivos.

ii. Estimación de desplazamiento en vehículo:

Además de estimar el desplazamiento de personas caminando, hemos contemplado el caso de uso el posicionamiento para conductores en un garaje. Teóricamente la velocidad del vehículo podría estimarse a partir de la aceleración percibida por el acelerómetro. En la práctica, esto no es factible, dado

que esta señal tiene mucho ruido (los acelerómetros de los teléfonos no son lo suficientemente precisos, el teléfono puede moverse, etc.). Como solución, hemos optado por estimar probabilidad de movimiento del vehículo, en base a las aceleraciones percibidas. Conociendo esta probabilidad y asumiendo que el rango de velocidad permitido en un garaje subterráneo está limitado, podemos obtener una aproximación del desplazamiento del vehículo.

3. Hemos integrado diferentes fuentes de información como observaciones en nuestro sistema de posicionamiento.

    a) *Fingerprinting* de radiofrecuencia. Utilizamos técnicas de *fingerprinting* para estimar la posición del usuario a partir de las señales escaneadas WiFi y BLE. En una fase previa construimos mapas de señal del edificio, que reflejan con que intensidad se recibe la señal de diferntes puntos de acceos WiFi o beacons BLE, desde cada punto del edificio. Posteriormente, comparando las señales percibidas con estos mapas, podemos hacer una estimación de la zona del edificio en la cual está el usuario.

    b) Brújula. Añadimos información de la brújula al sistema para mejorar la estimación de la dirección del usuario. Esto resulta especialmente útil en las fases de inicialización del sistema.

    c) Mapa de zonas transitables. Usamos un mapa para definir zonas transitables del edificio y así mejorar la precisión del sistema.

4. Experimentación sistema completo en un entorno real con usuarios caminando. Hemos desplegado el sistema en un entorno real y hemos evaluado su rendimiento. Se ha escogido un edificio de grandes dimensiones con áreas complejas como espacios abiertos, pasarelas, escaleras mecánicas,etc. con el objetivo de validar el funcionamiento del sistema en un espacio lo suficientemente genérico como para recrear la mayoría de situaciones que nos podemos encontrar. Se han realizado pruebas con diferentes dispositivos en diferentes posiciones y el sistema ha probado ser robusto y preciso.

5. Experimentación con el sistema completo en un entorno real en vehículo. Hemos desarrollado una aplicación para el guiado de conductores a una plaza de garaje. Hemos instalado el sistema en un parking subterráneo de varias plantas y hemos realizado varios experimentos para evaluar la calidad del posicionamiento. Los resultados de esta

experimentación son positivos y el sistema ha probado ser útil para el guiado de personas a una plaza de garaje.

El sistema de Localización en Interiores desarrollado ha probado ser robusto, adaptable a diferentes entornos, usuarios, casos de uso y fácilmente escalable. Finalmente, cabe destacar que los resultados de la invetigación reflejada en esta tesis han servido para mejorar la tecnología de posicionamiento de la empresa Situm Technologies, integrándose en su solución comercial, mejorando la precisión y minimizando la infraestructura necesaria para el correcto funcionamiento del sistema. Esta tecnología está actuaknebte instalada en cientos de edificios por todo el mundo y está siendo utilizada con éxito diariamente por miles de personas.

Además, los resultadosde esta investigación se han reflejado en dos publicaciones en revistas internacionales indexadas en JCR y una publicación en un congreso internacional.

En el transcurso de esta tesis, se han detectado una serie de mejoras potenciales de cara al futuro. Estas mejoras abarcan diferentes aspectos, desde la mejora de la estimación inercial, el proceso de calibración para la generación de los mapas de señal, o la incorporación de nuevos modos de funcionamiento. Algunos de los puntos detectados se enumeran a continuación:

1. Mejorar la estimación inercial. Como hemos explicado, estimamos la longitud del paso usando el período de zancada. Esta variable no es la única que afecta a la distancia recorrida. Aunque la odometría estimada tiene suficiente precisión para la mayoría de los casos, todavía hay margen de mejora. Utilizando la información de trayectorias pasadas del usuario, o incluso con fuentes de información adicionales (como el GPS, si hay trayectorias exteriores), sería posible crear modelos de longitud de paso ajustados a cada usuario, lo que puede mejorar aún más el rendimiento del sistema.

2. Mejorar la estimación de la orientación del usuario o dirección de desplazamiento. Nuestro sistema detecta la orientación del usuario en función de la orientación del teléfono. Esto puede suponer un problema, por ejemplo, si el usuario camina y gira el teléfono, el sistema interpretará que ha realizado un giro. Después de varios segundos la información del resto de los sensores terminará corrigiendolo, pero mientras tanto, la precisión del posicionamiento se degradará temporalmente. Detectar estos eventos y separar los giros del smartphone de los del usuario ayudaría a mejorar la precisión en estos casos.

3. Añadir nuevos sensores como fuentes de información. Hemos explorado el uso de sensores adicionales para mejorar el rendimiento del sistema en ciertos casos. Sensores

como el barómetro, presente en algunos teléfonos, pueden ayudar a estimar la planta en la que se encuentra el usuario y reducir el tiempo de convergencia. También hemos experimentado con la odometría visual para mejorar la estimación del desplazamiento. Esto puede ser especialmente útil para el posicionamiento de vehículos en entornos industriales.

4. Posicionamiento en interiores y exteriores. Hemos explorado también la posibilidad de utilizar la información del GPS para permitir el posicionamiento tanto en interiores como en exteriores. Esto hará posible la transición entre diferentes edificios, e incluso puede mejorar la precisión del sistema en áreas al aire libre como azoteas o patios interiores.

5. Simplificar los procesos de generación de mapas de señal. Actualmente generamos los mapas de señal caminando con el teléfono mientras este escanea las señales WiFi o BLE y marcamos en el mapa del edificio la posicion actual. Este proceso, aunque simple, puede llegar a ser tedioso en edificios grandes. En el futuro, se podrían aplicar técnicas como SLAM (Simultaneous Localization and Mapping) para generar estos modelos automáticamente.

6. Actualizar los mapas de señal. Con el tiempo, los mapas de señal pueden degradarse debido a cambios en la infraestructura del edificio. Se pueden añadir o quitar nuevos puntos de acceso, lo que con el tiempo puede hacer que la posición sea menos precisa. La aplicación de técnicas como SLAM ayudaría a actualizar constantemente estos modelos, detectando los puntos de acceso perdidos o añadidos e incorporándolos a los mapas del edificio.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Contents

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 1

INTRODUCTION

Nowadays people spend most of their time indoors [1]. Indoor spaces, have turn into large buildings where people can develop all kind of activities, from work to leisure, without having to go out at all. Our usual environment includes big shopping centers, subway networks, hospitals, educational centers, etc. These buildings can be huge and are frequently crowded, so it is easy to get lost or waste a lot of time looking for a particular place. In order to solve this issues, indoor positioning has gained great importance lately. With an indoor positioning system people can know where they are, and the best route towards their destination. In addition, they can provide useful information to the building managers, allowing them to extract statistics of visits, time spent in different areas, etc. Indoor positioning is also useful for tracking assets or workers. For example, in the security sector, it provides a useful tool that that allows security guards to be found quickly in case of emergency.

Since Global Navigation Satellite System (GNSS) does not work indoors, other approaches have taken over. Technologies based on different sensors such as infrared, ultrasound [2], vision based [3], magnetic field, audible sound or radiofrequency, have been explored. Radiofrequency (RF) based technologies are the ones that have become more popular lately [4]. These technologies usually require the location of transmitters along the building and that the subject to be located carries a receiver. It is necessary to know the distribution of the transmitters or to have a signal map in order to be able to relate the signals received with the space. When the receiver detects one or more signals, its position is estimated using the prior knowledge of the distribution of the transmitters or signals in the building. The most common approaches for position estimation using radiofrequency are proximity, triangulation or fin-

gerprinting [4]. Different RF technologies exist, from the expensive and highly accurate Ultra Wide Band (UWB) to the most popular and inexpensive WiFi and BlueTooth.

Most of the classic indoor positioning solutions relied in the use of dedicated hardware. These solutions were usually expensive and required complex installations. Using dedicated hardware is undesirable since it involves extra costs and is more difficult to implement widely. The final user will be more reluctant to use a location system if he needs to purchase or carry additional hardware. Fortunately, the vast majority of people already carry an intelligent device most of the time. Smartphones, tablets and wearables are already fully installed in our society. This devices have enough processing power and high-end sensors to allow the reception and processing of information from the environment. They have WiFi and BLE (Bluetooth Low Energy) reception cards which allow the detection of WiFi access points (APs) and Bluetooth beacons from the environment which are extensively used for indoor positioning. They also include inertial sensors like accelerometer and gyroscope that can be used to estimate the motion of the user and thus improve the robustness of the location systems. These characteristics and the pervasiveness of the smartphones makes them the ideal platform for indoor location.

Even though there has been a lot of research about indoor positioning with smartphones, there are still some issues that must be solved:

In order to achieve high positioning accuracy, the usual approaches still require installing additional hardware. For example, many commercial solutions rely on installing a high amount of beacons in the area. To some extent, the higher the density of beacons, the greater the precision. Of course, installing a large number of beacons also implies higher deployment and maintenance costs. In addition, the maximum position accuracy achievable from using only sensor based observations has a limit. This limit comes from the physical inherent characteristics of the sensor technology (infrared can't go through walls, camera based require direct vision, RF signal attenuation, multipath, etc. [5]) and the noise of the sensor [6].

Another problem with using smartphones for indoor location is that they can be used in a great variety of ways. They can be kept on the pocket or in bag, they can be hold on the hand or used to talk. A great advantage of these devices is that they have inertial sensors, which can be used to estimate the movement of the terminal and, therefore, the displacement of the user. However, the position in which they are carried implies important changes in the perceived signal, as well as the movements that are made with the terminal while using it normally. Many solutions require the device to be placed in an specific location [7, 8, 9], or carried in

a predefined way. This is undesirable, because the user should be able to use the device as desired without affecting the quality of positioning.

Additionally the user can perform different activities, like walking, going up and down stairs, sitting, driving, etc. The signals perceived from the inertial sensors would be different in every case as well as the motion of the person. In this thesis we consider both the case of pedestrians walking indoors, and drivers who travel in vehicles in an indoor car park.

1.1 Goal of the Thesis

The main goal of this thesis is to provide a robust indoor positioning solution for smartphones that maximizes location accuracy while minimizes the required infrastructure.

At the same time we will make a system that works in real world situations. It must be robust to different users, as well as allowing the free use of the terminal, without restricting its position. It should work in different indoor environments as well as performing typical activities such as going up / down stairs, walking, using the phone without moving, etc.

Our proposal will make use of the sensors of the smartphone in order to estimate and update the position of the user. It will combine absolute position measurements, taking advantage of the already existing infrastructure of the building (WiFi, BLE), with the information of the inertial sensors of the smartphone (accelerometer, gyroscope, magnetometer) in order to determine the displacement of the user. Combining this information will allow us to relate the measurements both temporally and spatially and thus obtain greater precision without the need to increase the infrastructure.

In order to achieve these goals, we have addressed the following milestones:

1. We have developed an indoor positioning system that calculates the user's position by fusing the information from the different sensors present in a mobile phone. By combining the information from multiple sources, we achieve greater accuracy and robustness than of the one that we would obtain using the different sensors separately. This system combines observations on the global position of the user with the estimation of its displacement and orientation. Our system is scalable, easy to deploy and flexible enough to allow the addition new sources of information in the future. In addition, the developed system runs on the phone itself in real time.

2. We have developed an inertial estimation module capable of identifying the user's movement. We have considered two main modes of displacement: walking and in a

vehicle. We do not want to restrict how the user uses the phone, so our solution works regardless of the position in which the phone is carried (hand, pocket, bag) and the activity performed such as walking, going up and down stairs, driving, etc. It is also robust to different users regardless of their sex or age. In order to achieve this we have addressed the following points:

a) We have developed a module to estimate the attitude of the phone (its orientation in the space) using the phone's inertial sensors. In order to do so we combined the information of the gyroscope, which provides accurate short term rotation information but ends up accumulating error (drift), with that of the accelerometer which allows to fix the position of the telephone with respect to the vertical axis. We also use the magnetometer in order to provide an estimation of the orientation of the device with respect to the north. However this information is unreliable indoors, because there are frequently elctromagnetic interferences caused by the machinery and the electrical installations of the building.

b) We have evaluated and implemented different techniques for estimating the user displacement. We differentiated two main scenarios: pedestrian ad vehicle displacement estimation.

   i. Pedestrian displacement estimation:
   The usual approach to calculate the displacement of a person walking using inertial sensors, involves detecting the steps taken and estimating the step length. Identify steps is relatively simple in constrained conditions (telephone in a fixed position and the user walking). When a person walks with a telephone in their hand, and in the absence of other stimuli, the signal perceived by the accelerometer shows a recognizable pattern, almost sinusoidal, in which each step taken produces a peak followed by a valley. However, if we do not restrict the position of the telephone and allow the user to make use of it in a normal way, while performing different activities, it becomes a complex task.
   We have addressed this issue with two completely different approaches:

      A. Shape based step detection. In this approach we identify the shape that a step produces in the acceleration signal and, through a learning process,

we obtain a pattern against which compare the new signal in order to identify the new steps.

B. Feature based step detection. In this approach we built a classifier from the most representative feature set extracted from the smartphone IMU in order to identify whether the user is walking or not. This classifier works with feature vectors extracted from the sensor signal, both in time and frequency domain.

With the information of the steps taken we carried out an study in order to identify which parameters affect most the step length and developed and algorithm to estimate it dynamically.

We created an extensive dataset in order to evaluate our step detection proposals. This dataset includes the sensor signals of a phone carried by multiple users while performing different activities. The singularity of this dataset is that, unlike in the existing public datasets, the steps are labeled one by one with the timestamp of the moment in which they are taken. This is especially important to evaluate the main problem of step counting algorithms which are false positives in sequences with activity, but without displacement.

ii. Vehicle displacement estimation:
We studied and developed a displacement estimation mode for vehicles in underground car parks. Theoretically the speed of the vehicle could be estimated integrating the acceleration perceived by the accelerometer. In practice, this is not feasible, since this signal has a lot of noise (phone accelerometers are not accurate enough, the phone can move, etc.). As a solution, we have chosen to estimate the probability that the vehicle is moving, based on the perceived accelerations. Knowing this probability and assuming that the speed range allowed in an underground garage is limited, we can obtain an approximation of the displacement of the vehicle.

3. We integrated different sources of information as observations in our positioning system (WiFi, BLE, compass and a occupancy grid map), and observed that the combination of all of them allows us to obtain an accurate and robust estimation of the user position.

4. We performed an extensive experimental study of our Indoor Location system both for the case of pedestrian Indoor Location and Indoor Location for drivers. We deployed

our system in different environments: a large shopping center, and a multifloor car park. In both scenarios our system proved to be accurate and robust enough to provide positioning and guidance.

The thesis is organized as follows:

- Chapter 2. We describe the algorithms used to fuse the information of the different sensors. We present the generic global architecture of our proposal whose particular parts we will describe in more detail in the following chapters.
- Chapter 3. We describe the algorithms used to obtain a rough estimation of the absolute position of the device in the building. These will be the observations of our system.
- Chapter 4. We describe the inertial framework of the system. We explain how we process the inertial sensors of the telephone to pass from raw measurements of the telephone to orientation and displacement estimates applicable to the user.
- Chapter 5. We describe the complete system for drivers in a parking garage.
- Chapter 6. We describe the complete system for pedestrians.
- Chapter 7. We describe the conclusions of this thesis and the future work.

CHAPTER 2

INDOOR LOCATION

As we have said in the previous chapter, the goal of this thesis is to obtain an indoor positioning system for smartphones that maximizes the accuracy while minimizes the required infrastructure. In addition, for this system to be useful, it must work in real world situations. It should be robust to different activities and uses of the terminal, it should estimate the position correctly when the subject is walking, standing, going up or down stairs or even in a vehicle. It must be robust to different positions, handheld, in the pocket, in a backpack, etc. It must work correctly in different environments, it should not depend on specific characteristics of the building and be easily deployable in any environment.

Many of the classic approaches to indoor positioning rely on a single sensor. For example, estimating the position of the subject triangulating WiFi signals. Nevertheless, the position accuracy achievable from using only single sensor based observations has a limit. This limit comes from the physical inherent characteristics of the sensor technology (infrared can't go through walls, camera based require direct vision, RF signal attenuation, multipath, etc. [6, 5]) and the noise of the sensor [6]. It is known that fusing the information from different sensors is more robust than using only one [6].

On the one hand, the physical characteristics of different sensor can complement each other. For example, the signal emitted by a WiFi access point covers a wide area of space that even can cross some walls, while an infrared signal covers a smaller area and it is only received if there is a direct line of sight. For localization purposes, listening to a WiFi signal, you can estimate a position in a wide area, but the accuracy of that position will be lower than that obtained with an infrared sensor. On the contrary, in order to positioning only with infrared

it would be necessary to populate densely with emitters the area. In addition, combining the information of different sensors can solve the problem of having blind spots. As an example, there may be an area where no WiFi access point is heard but there is a BLE signal.

In our case, we use the different sensors of the smartphone fusing the information received to estimate the user's position. We scan WiFi and BLE signals, as observations, and combine the accelerometer, gyroscope and magnetometer signals to estimate the motion of the subject.

Combining observations with the motion of the subject allows us to relate (both temporally and spatially) sequential sensor measurements and thus be able to achieve greater accuracy.

To this extent, probabilistic position estimation has been used in robotics for years with great success [10, 11, 6, 12]. In this paradigm, there are one or more hypothesis of a state with its own uncertainty. The state is modified with observations, which provide information of the environment, and with information of motion or control which indicates the movement of the subject. Both observations and motion estimations have their own uncertainty. Improving the accuracy of observations or the motion estimation will affect the precision of the position.

2.1 State of the art

Fusing information from multiple sources allows to obtain position estimates more accurate and robust than those that would be obtained using only one sensor. Within the fusion of sensors there are different alternatives, such as interval calculus, fuzzy logic, etc.[13] The most popular of these is the probabilistic fusion, especially the methods based on Bayesian Filtering [13, 6].

In the Bayes Filter the variables or state to be estimated can not be observed directly. The belief distribution is estimated from the control, in our case the motion of the subject, and from the observations, readings from the sensors that provide information about the environment.

Within the Bayesian fusion paradigm there are multiple approaches, of which some the most popular are the Kalman Filter (KF) and the Particle Filter (PF).

The Kalman Filter main drawback is that it only works with linear problems. Extendend Kalman Filter (EKF) solves this limitation by linearising the estimation problem in order to apply the Kalman Filter. However the probability density function is approximated by a Gaussian which may distort the underlying real distribution and make the filter diverge [14].

In the Particle Filter, each particle represents a hypothesis of the state to be estimated, which in our case is the user's pose. The filter iterates in two main stages: prediction and

update. In the prediction stage, the particles are displaced using the information of movement of the user. In the update stage the observations, measurements of the sensors, are used to weight the particles. Particles that are coherent with these observations will have a greater weight than the others. After several iterations, one or several particle clusters will be formed in the areas where the user may be located.

The particle filters have important advantages over other alternatives such as the Kalman Filter. They allow to represent multimodal probability distributions. This is frequent in localization because there may be multiple plausible positions in a given moment of time. Particle Filters are highly scalable. Unlike with the Kalman Filter, in which adding new measurements is complex, in the Particle Filter there can be easily added as many observations as required.

2.2 Architecture

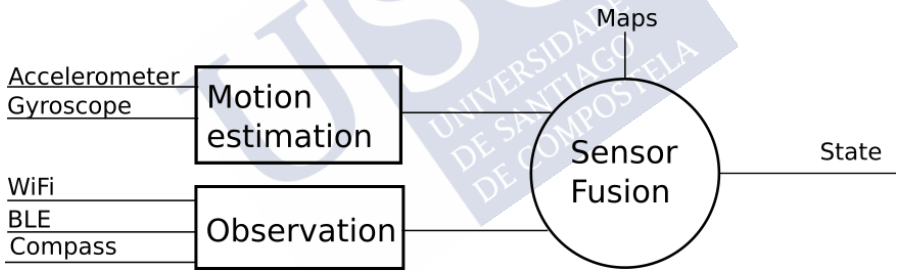


Figure 2.1: Global system architecture

Figure 2.1 represents the global architecture of the system. On the one hand we have the inertial sensors of the smartphone (accelerometer, gyroscope) which we use for the estimation of the movement. This is an isolated block, so we can modify the motion models and adapt the displacement estimation depending of the activity performed, use mode, etc. without affecting the rest of the system.

On the other hand we have the measurements received from the sensors of the smartphone, which in our case are comprised of WiFi, BLE and Compass readings. WiFi and BLE readings are used together with prior knowledge of the building such as maps or signal maps to get rough estimations of the global position of the subject. These estimations together with the compass are the observations.

The estimation of the movement and the observations are combined using a particle filter to obtain, finally, an estimate of the state. This state is the user's pose.

This architecture allows us to add or remove sources of information easily, maintaining the robustness of the system and allowing us to adapt it to different situations.

2.3 Particle Filter

In this section we describe a particle filter used to determine the position of the user at each instant. This particle filter merges information provided by inertial sensors, WiFi access points (APs), Bluetooth beacons, and an occupancy map.

The Particle Filter estimates at each instant the probability with which the user is in a certain state s_t given the set of all previous actions $U_{t:1} = \{u_t, u_{t-1}, \ldots . u_1\}$ and measurements $Z_{t:1} = \{z_t, z_{t-1}, \ldots, z_1\}$. In our case the state s_t represents a position of the user on a floor map:

$$s_t = \begin{pmatrix} x_t \\ y_t \\ \theta_t \\ floor_t \end{pmatrix} \tag{2.1}$$

where (x,y) is the 2-dimensional position, θ_t is the heading of the user, and $floor_t$ is the current floor.

The actions $u_t, \forall t$ represent the movement of the user at every instant t. In particular u_t is described as the distance traveled by the user and the direction of movement, i.e. $u_t = [d_t, \Delta\theta_t]$. The *Motion Estimation* module shown in Fig. 2.1, will use the inertial sensors of a mobile phone to estimate these actions $u_t = [d_t, \Delta\theta_t], \forall t$.

Finally, the measurements $Z_t = \{z_t^1, \ldots, z_t^{n_t^z}\}$, are the sensor readings or observations perceived at an instant of time t from the n_t^z different sources of information (e.g. signals received from a set of Bluetooth Low Energy transmitters (BLE)).

Thus, as it was just pointed out, the particle filter estimates the probability distribution for the state space at each instant. This probability distribution is usually called *belief function* $bel_t(s)$, and it is often represented as:

$$bel(s) = p(s|z_t, u_t) \tag{2.2}$$

where $bel(s_t)$ represents the probability of the user being in state s_t. This belief distribution can be recursively estimated [6]:

$$bel_t(s) \propto p(z_t|s) \int p(s|s_{t-1}, u_t) bel_{t-1}(s) ds_{t-1} \tag{2.3}$$

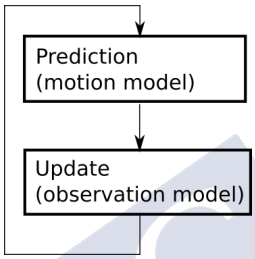


Figure 2.2: Prediction and update iterative cycle.

This equation (2.3) involves two stages: *prediction* and *update* (figure 2.2). The first stage, *prediction*, estimates the belief distribution $\hat{bel}_t(s)$ considering the previous one $bel_{t-1}(s)$ and the current movement of the user u_t.

$$\hat{bel}_t(s) = \int p(s|s_{t-1}, u_t) bel_{t-1}(s) ds_{t-1} \tag{2.4}$$

The term $p(s|s_{t-1}, u_t)$ is the *motion model*. It describes the probability with which the user evolves from state s_{t-1} to s when it performs the action u_t.

The *Update* stage corrects the predicted belief distribution taking into account the last sensor measurements.

$$bel_t(s) \propto p(Z_t|s_t) \cdot \hat{bel}_t(s) \tag{2.5}$$

$p(Z_t|s_t)$ is called *observation model*, and it represents the probability of receiving the measurements Z_t when the user is in state s_t. The joint probability $p(Z_t|s_t)$ is not easy to estimate directly. That is because there are sensors that may not be present in certain device and even if they are, their readings may not be received synchronously. Because of that, and assuming that the sensors are conditionally independent we can approximate $bel_t(s)$ with the following equation:

$$bel_t(s) \propto \hat{bel}_t(s) \cdot \prod_{k=1}^{n_t^z} p(z_t^k|s_t) \tag{2.6}$$

In chapter 3 we will describe the processing of the raw sensor data required to obtain the motion model, and in the 4 we will describe the observation model.

Once we have the *observation* and *motion* models, we can apply a particle filter to estimate the position and orientation of the subject. Particle Filters approximate a belief distribution $(bel(s_t))$ using a set of n^p variables called particles $X_t = \{X_t^i = (s_t^i, \omega_t^i), \forall i \in 1,\ldots,n^p\}$. Each particle keeps an hypothesis about the current state of the system (s_t^i) and a weight w_t^i that somehow, reflects the probability of that hypothesis being true (hypothesis plausibility). Hence the summation of the weights of the whole set of particles must be equal to one. Both, the hypothesis s_t^i, and the weight ω_t^i will vary over time. Taking this into account, Particle Filters represent the belief $bel(s_t)$ as:

$$bel(s_t) \approx \sum_{i=1}^{n^p} \omega_t^i \cdot \delta(s_t^i - s_t) \tag{2.7}$$

Where $\delta(s_t^i - s_t)$ is the Dirac's delta function centered at s_t^i. Starting from a set of random particles with the same value for their weights, the algorithm iterates over four stages: prediction, update, resampling and pose estimation.

1. Prediction. At this stage the hypotheses kept by all the particles are updated according to the motion model:

$$s_t^i \sim p(s|s_{t-1}^i, u_t) \forall i \in \{1,\ldots,n^p\} \tag{2.8}$$

Since $s_{t-1}^i = (x_{t-1}^i, y_{t-1}^i, \theta_{t-1}^i, floor_{t-1}^i)$, $\forall i$, represents the hypothesis of each particle, these hypotheses are updated according to the following expression:

$$s_t^i = \begin{pmatrix} x_{t-1}^i + d_t \cdot cos(\theta_{t-1}^i + \Delta\theta_t) + \mathscr{E}x_t \\ y_{t-1}^i + d_t \cdot sin(\theta_{t-1}^i + \Delta\theta_t) + \mathscr{E}y_t \\ \theta_{t-1}^i + \Delta\theta_t + \mathscr{E}\theta_t \\ floor_{t-1}^i \end{pmatrix} \tag{2.9}$$

Where $\mathscr{E}x_t$, $\mathscr{E}y_t$ and $\mathscr{E}\theta_t$ represent zero mean Gaussian noise and are randomly drawn from normal distributions $\mathscr{E}x_t \sim N(0,\sigma_{xt})$,$\mathscr{E}y_t \sim N(0,\sigma_{yt})$, $\mathscr{E}x_t \sim N(0,\sigma_{\theta t})$.

2. Update. At this stage the weights of the hypothesis stored in the particles are updated using the observation model:

$$\omega_t^i \sim p(Z_t|s_t^i) \cdot \omega_{t-1}^i = \omega_{t-1}^i \prod_{k=1}^{n_t^z} p(z_t^k|s_t^i) \tag{2.10}$$

Finally, the updated weights are normalized so that their total sum equals 1.

3. Resampling: This stage will help to avoid the fall in local minima. Particle Filters suffer from a phenomenon called sample depletion [12], this means that after some iterations all particles will probably have a negligible weight except one. An approximation of the amount of depletion can be obtained:

$$N_{eff} = \frac{1}{\sum_{i=1}^{n^p} (w_t^i)^2} \tag{2.11}$$

when N_{eff} falls under a threshold, it is advisable to perform a resampling, where a new set of particles is created from the current one. In our case, we will use the Low Variance Resampling technique [6].

4. Pose Estimation: After some iterations, particle clouds will converge around one or more plausible locations, from which we can estimate the position of the subject. Basically we group the particles in clusters[15] (according to their hypothesis s_t^i) and then we select the cluster with the highest total weight (the weight of a cluster is computed as the sum of the weights of all the particles included in such cluster and that are over a predefined threshold). After this, the pose of the subject is computed as the average of all the particles included in the selected cluster.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 3

MOVEMENT ESTIMATION

In the previous chapter (Chapter 2) we described the particle filter that we use to estimate the pose of the user. We have explained that we combine the movement of the user (*motion model*) with the observations of its global position (*observation model*). In this chapter we will describe how we estimate the movement of the user from the data of the inertial sensors of the smartphone.

3.1 Motion model

In order to be able to known how the user moves through space, we must process the inertial data and estimate whether or not the device has moved over time, and how much. More formally, we are trying to model the relationship between accelerometer and gyroscope data

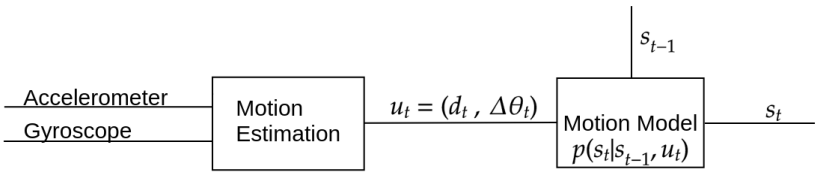


Figure 3.1: Motion model

with the displacement (d) and change of direction ($\Delta\theta$) of the user over time. We must keep in mind that the displacement and direction of the phone may not be aligned with that of the user.

This relationship is normally called the motion model, represented in figure 3.1. This figure represents that the input of the system is the data from the smartphone sensors: accelerometer, gyroscope and magnetometer. From these data we estimate the movement of the user at each instant of time $u_t = (d_t, \Delta\theta_t)$. The estimation of these variables is carried out in the module *motion estimation*. The motion model defines the probability of transition from the state s_{t-1} to the state s_t, when the motion u_t occurs. In the Particle Filter framework, the motion model is represented as $p(s_t|s_{t-1}, u_t)$, as shown in figure 3.1 . Note that we have already defined this equation in the previous chapter when we defined the prediction stage of the particle filter (chapter 2, equation 2.4).

To compute the motion model we work with an stochastic representation of the transition of the position from one state s_{t-1} to the next one s_t due to $u_t = [d_t, \Delta\theta_t]$ (where d_t is the displacement, i.e., distance traversed amongst two time instants, and $\Delta\theta_t$ is the change of the heading of the user amongst these two timestamps):

$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \\ floor_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \\ floor_{t-1} \end{pmatrix} + \begin{pmatrix} d_t cos(\theta_{t-1} + \Delta\theta_t) \\ d_t sin(\theta_{t-1} + \Delta\theta_t) \\ \Delta\theta_t \\ 0 \end{pmatrix} + \vec{\varepsilon}_t \tag{3.1}$$

where $\vec{\varepsilon}_t$ is a zero-mean Gaussian noise vector with which we represent the fact that it is not possible to achieve a precise computation of the displacement and change in heading due to the noise and bias of the sensors, vibrations, etc:

$$\vec{\varepsilon}_t \approx \begin{pmatrix} N(0, \sigma_{xt} = \lambda_x \cdot d_t) \\ N(0, \sigma_{yt} = \lambda_y \cdot d_t) \\ N(0, \sigma_{\theta t} = \lambda_\theta \cdot \Delta\theta_t) \\ 0 \end{pmatrix} \tag{3.2}$$

where $N(0, \vec{\sigma})$ is a Gaussian function with average 0 and standard deviation $\sigma(\lambda)$. The parameters λ_x,λ_y and λ_θ are constants in the range [0,1] adjusted experimentally to represent the percentage of error associated to the displacement (d_t) and change of heading ($\Delta\theta_t$).

It is also important to realize that this motion model does not change the estimation of the current floor. The change of floor will be detected in another stage, based on the observations. We will discuss that later on chapter 4.

According to Eq. (3.1), we see that we must estimate the displacement d_t and heading $\Delta\theta_t$ in order to approximate its state. We will use the information provided by the inertial sensors (IMU) of the mobile phone to estimate both d_t and $\Delta\theta_t$. This estimation is performed in the module *Motion estimation* of the figure 3.1.

3.2 Smartphone Sensors

As shown in figure 3.1 and equation 3.1, we need to process the raw sensor readings in order to obtain the displacement (d_t) and rotation ($\Delta\theta$) of the user. In this section we describe the characteristics of the sensors present in most smartphones: accelerometer, gyroscope and magnetometer.

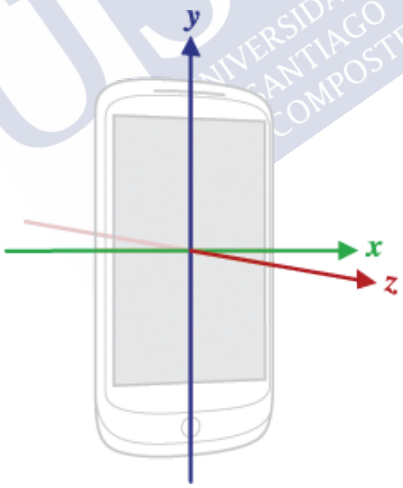


Figure 3.2: Android device sensors axes [16].

- *Accelerometer*: The accelerometer measures the acceleration forces to which the smartphone is subjected. Nowadays, smartphones have triaxial accelerometers, that measure the acceleration to which the terminal is subjected in each one of the axes of the tele-

phone, as it is shown in the figure 3.2. We can use the accelerometer readings for two main goals:

- *Orientation estimation:* It is important to remember that we do not know how the device is carried or its orientation. Given that we do not want to impose any restriction on the way in which the user carries the smartphone, we cannot make assumptions about if the smartphone is in the pocket, handheld, being used to talk, etc.

  Since accelerometer measures both the linear accelerations to which the phone is subject and the acceleration of gravity we can use this information in order to estimate the orientation of the smartphone with respect to the *up* axis. We achieve this by identifying how the gravity affects the different axes of the smartphone in the absence of other accelerations.

- *Displacement estimation:* We also can use these readings in order to estimate the displacement of the device. In ideal conditions we could integrate twice the acceleration in order to obtain the device displacement. However, in the real world this is not feasible since sensors have noise, that would accumulate through time exponentially. Nevertheless, there are other alternatives. For example, when an user is walking carrying a smartphone, the acceleration to which the device is subjected presents a periodic and recognizable pattern which can be detected. By identifying and counting the steps, and having an estimation of the step length, it is possible to provide an estimation of the total displacement. Of course, this is not so simple, given that the received signal will be different depending on the person, location and orientation of the smartphone, activity carried out, etc. It may even be the case of detecting patterns similar to those of walking while the user is not moving. Finally there are also cases in which a user will move without walking, such as in a vehicle. We have also addressed this case as we will see in section 3.5.

- *Gyroscope:* The gyroscope measures the angular velocity in each of the three axes of the smartphone (figure 3.2). Gyroscopes are frequently used to estimate rotations by integration. This allows to achieve accurate estimations of the rotation in short time intervals, but in the long term, the error accumulates (drift). In addition, these rotations are relative to the device, which may not be aligned with the user. Also, given that these

measurements are relative rotations, they do not provide an absolute orientation and it is required to known the initial orientation of the device in order to estimate it. We will solve this issue using a map and the particle filter as we will explain in section 4.

- *Magnetometer:* Finally, the magnetometer measures the electromagnetic field present in the environment. This includes the Earth magnetic field, which points approximately towards the north. This provides information about the absolute orientation, which can be very useful for our purpose. However, the magnetometer is also prone to detect electromagnetic interferences which are very frequent indoors, and may produce errors in the detection of the true north. In addition the smartphone itself can create a electromagnetic disturbance which may lead to additional errors.

3.3 Rotation Estimation ($\Delta\theta$)

In this section we describe how we estimate the change in orientation, $\Delta\theta$, from the inertial sensor readings.

3.3.1 Reference Frames and Orientation Representation

As we pointed out before, we need to know the orientation in the space of the smartphone. We must be aware of the existence of two reference systems:

1. *Sensor Frame:* A local reference system linked to the phone (also known as body frame). This local frame is defined relative to the device's screen.

2. *Earth Frame:* inertial reference system, the axes of which always point towards the same points (with respect to Earth).

   In the case of the inertial-Earth frame, we work with a frame analogous to the East North Up (ENU) coordinate system [17], in which the x-axis points toward the East, the y-axis points towards the North Magnetic Pole and the z-axis is pointing in the opposite direction of the gravitational force. The accelerometer and gyroscope readings are provided in the body frame, and therefore it is convenient to project them into the inertial-Earth frame in order to estimate the movement of the person who carries the mobile.

The attitude of the smartphone is its orientation with respect to the inertial-Earth frame. There are several ways to represent the attitude of an object in space. Euler Angles, Rotation Matrix, Angle-Axis or Quaternions are some of the most popular representations.

In order to represent this orientation, we use quaternions [18, 19] because of their many advantages over other representations. A quaternion is a four-dimensional vector that represents the relative orientation between two coordinate frames B and A, as a rotation of an angle θ around a three-dimensional axis r:

$$ {}^{A}_{B}q = [q_0\ q_1\ q_2\ q_3] = \left[cos\frac{\theta}{2}, r_x sin\frac{\theta}{2}, r_y sin\frac{\theta}{2}, r_z sin\frac{\theta}{2} \right], \tag{3.3} $$

where ${}^{A}_{B}q$ is the normalized quaternion that represents the orientation of a frame B relative to a frame A [19]. Following this notation, we will use ${}^{S}_{E}q_t$ to refer to the current value of the quaternion that represents the orientation of frame E (Earth frame), relative to the frame S (Sensor frame). This quaternion represents the current orientation of the mobile phone.

3.3.2 Device orientation estimation

Once we have decided how we are going to represent the device orientation, we need to choose an algorithm to estimate it. Theoretically, knowing the initial orientation of the device we could integrate the angular rate from the gyroscope in order to estimate this orientation. However, in the real world, sensor noise makes that the error accumulates provoking drift in the estimated orientation.

Because of this drift, the usual approaches involve the combination of information from multiple sensors in order to eliminate or minimize it.

To this extent we use an Extended Kalman Filter (EKF) [20] to estimate the quaternion that represents the attitude of the device. The EKF is an recursive estimator that calculates the internal state of dynamical system integrating measurements from noisy sources. In our case it will work with a *process model* which is represented as the evolution of the attitude of the smartphone due to the rotation of the mobile detected with the gyroscope (${}^{S}\omega_t$). The EKF differs form the Kalman Filter in which it can work with non linear models [21]. It works with statistical models of how the state evolves and how the observations relate to this state by minimizing the mean-squared error. It works in two stages, prediction and update.

- *Prediction:*

  In the prediction stage, it obtains the *a priori* estimation from the previous state and a known control or motion input.

  In our case, in the prediction stage, we integrate the angular rate from the gyroscope (${}^{S}\omega_t$) in order to obtain an *a priori* estimation (${}^{S}_{E}\hat{q}^{-}_{t}$).

  Since the gyroscope measures the rate of angular velocity it can be used to determine the orientation:

$$ {}^{S}_{E}\hat{q}^{-}_{t} = {}^{S}_{E}\hat{q}^{+}_{t-1} + \frac{1}{2}({}^{S}_{E}\hat{q}^{+}_{t-1} \otimes {}^{S}\omega_t)\Delta t \tag{3.4} $$

  where $\otimes$ is the quaternion product, ${}^{S}_{E}\hat{q}^{-}$ is the *a priori* estimate of the state, before the observations z_t are processed, whereas ${}^{S}_{E}\hat{q}^{+}$ is the a posteriori estimate. Therefore, given an initial orientation, the information provided by the gyroscope can be integrated to determine the device's change in position. Nevertheless, in our case we do not know such initial orientation, therefore, as we described in chapter 2 we will overcome this limitation using a particle filter.

- *Update:*

  In the update stage, the *a priori* estimation is corrected using measurements or observations about this state in order to obtain the *a posteriori* estimation. In our case, we correct the initial estimation using the accelerometer readings. We estimate the gravity from the acceleration in order to use it as an *anchor* to avoid drift in the vertical axis.

  The gyroscope has a high rate of error, its data drifts over time, is unstable, and low angular velocities might not be properly registered. Because of all this, and to compensate all these errors, the EKF uses a *measurement model* to compute the posterior estimate ${}^{S}_{E}\hat{q}^{+}$, Eq. 3.4. In particular, this observation model works on the basis that when the magnitude of the accelerometer signal ($\| {}^{S}a_t \|$) is close to Earth's gravity (g), which would mean that the mobile phone is not being affected by other forces, the accelerometer should measure only Earth-gravity in the local device frame.

  In this case, the projection of the unit gravity vector in the Earth frame, $\vec{z_G} = \overrightarrow{G} / \| G \| = \{0,0,0,1\}$, into the local reference system (body frame), should coincide with the information detected by the tri-axial accelerometer signal, after its normalization ${}^{S}\hat{a} = \{0, \frac{a_x}{\|{}^{S}a\|}, \frac{a_y}{\|{}^{S}a\|}, \frac{a_z}{\|{}^{S}a\|}\}$.

This projection of $\vec{z_G}$ into the local reference system can be computed as:

$$^{S}\vec{up} = {}^{S}_{E}\hat{q}_t^{-*} \otimes \vec{z_G} \otimes {}^{S}_{E}\hat{q}_t^{-} \tag{3.5}$$

where ${}^{S}_{E}\hat{q}_t^{-*}$ is the conjugate of ${}^{S}_{E}\hat{q}_t^{-}$ [19]. According to this, when z_G is rotated using the current estimation of the quaternion ${}^{S}_{E}\hat{q}_t^{-}$ (Eq. 3.4 and 3.5), we should obtain the same values as those provided by the normalized accelerometer signal ${}^{S}\hat{a}_t$. In fact, this difference amongst the predicted values of the accelerometer and the ones observed is precisely what the EKF uses to correct the a priori estimate of the quaternion:

$${}^{S}_{E}\hat{q}_t^{+} = {}^{S}_{E}\hat{q}_t^{-} + K_t[{}^{S}_{E}\hat{q}_t^{-*} \otimes \vec{z_G} \otimes {}^{S}_{E}\hat{q}_t^{-} - {}^{S}\hat{a}_t] \tag{3.6}$$

where K is the Kalman gain, a dynamic parameter that weights the importance between the predicted state ${}^{S}_{E}\hat{q}_t^{-}$ and the information carried out by the observations ${}^{S}\hat{a}_t$, which depends on the process noise and the observation noise [6].

As we just pointed out, one of the reasons why we estimate the quaternion with the EKF, is to get the changes in orientation of the device. In this case, to extract its heading θ_t the components of the quaternion estimated by the EKF. We use the yaw-pitch-roll (YPR) order of rotation[22].

$$\theta'_t = atan2(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)) \tag{3.7}$$

Where $atan2(y,x)$ is a function present in many programming languages that computes the arctangent of (y/x) taking into account the signs of the arguments to determine the correct quadrant. Note that we will not work with ${\theta'}_t$, which represents the heading assuming a known initial value. Instead our motion model (Eq. (3.1)), works with the increments $\Delta\theta$:

$$\Delta\theta = \theta'_t - \theta'_{t-1} \tag{3.8}$$

In order to retrieve the true initial orientation, and sort out this uncertainty about the initial heading, we will work with the particle filter that merges the information provided by the inertial sensors of the mobile phone with other sources of information (Chapter 2).

3.3.3 Compass (θ^c)

As we have explained in the previous section, we fuse the information of the accelerometer and gyroscope in order to estimate the rotations of the device ($\Delta\theta$). We have explained that we do not use the magnetometer in the estimation of $\Delta\theta$ because of the inaccuracies of the magnetometer readings, due to the presence of electromagnetic interferences in the environment and caused by the device itself.

However knowing the orientation of the device with respect to the north is still very useful for localization, especially in the context of the particle filter, for the estimation of the orientation in absence of movement. Because of that, we do estimate the orientation respect to the north, but we do not use it in our motion model (due to the high error rate). Instead we will use it later in the observation model, as we will explain later (chapter 4).

In order to estimate θ^c we use another EKF, like the one described in section 3.3.2, but in this case we also use magnetometer readings along with the accelerometer readings in the update stage. Thus the electromagnetic field will be included as an additional update in the EKF. This update is similar to that of the accelerometer defined in equations (3.5) and equation (3.13).

The state of this EKF is the quaternion q^c. First we project the unit vector that points towards the North in the Earth reference frame $\vec{z_N} = \{0,0,1,0\}$, into the local reference frame:

$$ {}^S_E\hat{q}_t^{c-*} \otimes \vec{z_N} \otimes {}^S_E\hat{q}_t^{c-} \tag{3.9} $$

It is important to keep in mind that this vector is supposed to point towards the geographic North Pole. It is a vector that points towards the North tangent to the Earth surface.

This is not directly comparable to the readings of the magnetometer. In the absence of electromagnetic interferences, the magnetometer detects the Earth magnetic field, which is different from the Geographic North. It has an *declination* difference in the horizontal plane, and an *inclination* (it points towards the Down axis). We will ignore the *declination* difference for simplicity, since it is generally negligible in comparison to sensor noise of the smartphone sensors. However we will deal with the *inclination* before integrating the reading into the EKF in order to avoid an error in the vertical axis.

After normalization we obtain an unit vector that points toward the magnetic north in the

sensor reference frame:

$$^{S}\vec{N}^{mag} = \{0, \frac{mag_x}{\| mag \|}, \frac{mag_y}{\| mag \|}, \frac{mag_z}{\| mag \|}\} \tag{3.10}$$

We want to integrate it in the EKF as an observation to update the orientation, without affecting the *up* axis. Thus we apply the following transformations.

First, we use the $\vec{up}'$ vector defined in equation (3.5) and $^{S}\vec{N}^{mag}$, in order to obtain a unit vector that points toward the Geographic East and is orthogonal to both vectors.

$$^{S}\vec{E} = ^{S}\vec{N}^{mag} \times \vec{up} \tag{3.11}$$

Then we approximate the vector that points towards the Geographic North:

$$^{S}\vec{N}^{G} = ^{S}\vec{up} \times ^{S}\vec{E} \tag{3.12}$$

where $^{S}\vec{N}^{G}$ is the vector that points towards the Geographic North (ignoring the *declination*).

At this point, and remembering the equation (3.9), we can integrate $^{S}\vec{N}^{G}$ into the EKF in the same way that we did with the accelerometer in equation (3.13):

$$^{S}_{E}\hat{q}_t^{c+} = ^{S}_{E}\hat{q}_t^{c-} + K_t[^{S}_{E}\hat{q}_t^{c-*} \otimes \vec{z}_N \otimes ^{S}_{E}\hat{q}_t^{c-} - ^{S}\vec{N}^{G}{}_t] \tag{3.13}$$

Finally we estimate the orientation respect to the north θ_t^c.

$$\theta_t^c = atan2(2(q_0^c q_3^c + q_1^c q_2^c), 1 - 2(q_2^{c\,2} + q_3^{c\,2})) \tag{3.14}$$

where θ_t^c is orientation of the device with respect to the Geographic North. We will explain in chapter 4, section 4.3 how we integrate this observation in the Particle Filter.

3.4 Pedestrian Displacement Estimation (d_t)

In section 3.1 we have defined our motion model and explained that it depends on the change of heading, $\Delta\theta_t$ and the user displacement, d_t. In section 3.3 we described how we estimate $\Delta\theta$, thus it only remains to explain how we estimate d_t.

The usual approach to estimate the displacement of a person walking using inertial sensors, involves detect and count the steps taken. When a person walks with a telephone in their hand, and in the absence of other stimuli, the signal perceived by the accelerometer

shows a recognizable pattern, almost sinusoidal, in which each step taken produces a peak followed by a valley. This is easily identifiable, if the person is indeed walking, however the person can perform other activities not related to walking that produce a similar pattern on the accelerometer signal.

In this section we describe how we estimate, from the raw sensor data, the displacement (d_t) of a person walking. This involves two stages, first detecting and counting steps, and finally the estimation of the distance travelled.

We have addressed the step recognition and counting stage applying two different approaches:

1. Shape based step detection: We train a classifier that uses directly the shape of the acceleration time series to detect characteristic patterns of walking. When a step is taken it produces a acceleration with a characteristic shape. This makes it possible to obtain one or more reference patterns of real steps that we can later compare to the acceleration signal in order to identify if a step has been taken.

2. Feature based step detection: In this approach we built a classifier from the most representative feature set extracted from the smartphone IMU in order to identify whether the user is walking or not. We will discuss this approach in section 3.4.3.

Finally, after detecting the steps taken, we estimate the step length dynamically as we will explain in section 3.4.6.

In the final version of the complete Positioning System we opted to use the Feature Based displacement estimation approach because it is more easily generalizable to different users. In addition to estimating the displacement of a person walking, we have also adapted our system to work in vehicles. The approach used to estimate the displacement in a vehicle will be explained in section 3.5).

3.4.1 Shape Based.

For this approach we decided to learn the step models in the mobile itself. These models should gather the specific characteristics of both, the mobile and the user.

The strategy we have developed works on three stages: first, our proposal will ask the user to walk normally. During this first stage, our application will collect some basic statistics of the information provided by the accelerometer and will collect samples corresponding to a set

of steps. From this initial set of steps our proposal will build a model (second stage), and the user will be able to test it and to decide whether the performance is good enough. Finally, a further improvement in the model can be carried out through a third and final stage in which the user records false positives (by simply moving the mobile in the arm but without walking).

In the following sections we will explain in more detail these stages.

Segmenting time series to detect steps

In this stage we collect data of the user walking normally and segment the acceleration signal in order to obtain a set of step candidate. Usually, the movement of walking gives rise to a very characteristic vertical acceleration signal which looks very similar to sinusoidal waves [23]. This is the reason why in this stage we will use a simple version of is known as peak detection [24] to identify steps. For us a step candidate is the acceleration signal obtained in a time interval and that might reflect the fact that the user has given a step. Therefore this algorithm chops the acceleration signal into segments that might reflect steps.

Obtaining a model

As it was pointed out at the beginning of section 3.4.1, our system will build a first model starting from a set of N initial potential steps collected while the user moves. Since most probably the walking speed of user will not be constant, we must assume that the number of data points in every gait cycle is not identical. Algorithm 1 shows how through the peak segmentation of the first stage we obtain a set of accelerations A for each step, as well as the time instants that these accelerations occurred. Therefore, to build a model, the time-scale of all the steps is normalized, i.e., $T'_j[i] = T_j[i]/max\{T_j[]\}, \forall j = 1,...,N$. After this normalization all acceleration samples have a time stamp in the interval $[0,1]$, being 0 the time associated to the first sample and 1 the times stamp corresponding to the last acceleration sample. The purpose of this is to achieve a solution that is, somehow, robust to differences in walking speed.

Once all acceleration time series have been normalized, we will look for a template, i.e., a representative step that can be used for template matching in order to decide when a step might have occurred. In order to determine the most representative step we need a similarity measurement. In this case we have used *Dynamic Time Warping* (DTW) and in particular FastDTW [25]. Dynamic Time Warping is suitable to match temporal series, basically it finds the optimal alignment between the samples corresponding to two different time series,

$counter = 0$;
$A = []$;
$T = []$;
$initial\_time = current\_time\_stamp$;
last_state="not_identified";
while process is not finished **do**
  Sc="neither_peak_nor_valley";
  if ${}^{e}a_z(t) >= max({}^{e}a_z(t-1), {}^{e}a_z(t+1)) \& ({}^{e}a_z(t) > Th_{peak})$ **then**
    Sc="peak";
  else if ${}^{e}a_z(t) <= min({}^{e}a_z(t-1), {}^{e}a_z(t+1) \& ({}^{e}a_z(t) < Th_{valley})$ **then**
    Sc="valley";
  end
  if *Sc="peak"* **then**
    last_state="peak"
  end
  if *Sc="valley" & last_state="peak"* **then**
    last_state="valley" ;
    steps=steps+1 ;
    $counter = 0$;
    $initial\_time = current\_time\_stamp$;
    Analise_wheter_step(A,T) ;
  end
  $A[counter] = {}^{e}a_z$;
  $T[counter] = current\_time\_stamp - initial\_time$;
  $counter = counter + 1$;
end

Algorithm 1: Detection of peaks and *step candidates*

minimizing the distance amongst them. Therefore it allows the comparison of time series even if one of them may be warped non-linearly by stretching or shrinking it along its time axis [25].

There is another issue we had to deal with, and that is related with whether it is convenient or not normalizing the time series, i.e. working with steps made up by the same number of samples. According to [26] length normalization reduces the recognition accuracy in application domains where the length of the compared time series matters for their classification. In our case we do not want such a dependency since it might lead to solutions that are too sensitive to gait frequency. Another study [27] claims that making the sequences to be of the same length has no detrimental effect on the performance of DTW. Due to all this, we have

opted for normalizing the time series that are going to be compared, i.e., when they are of different lengths, one of them must be reinterpolated.

The step that is going to be used as template for the matching processes, is determined automatically from the N potential steps collected while the user moves:

$$rep = argmin_{j\in[1,N]}\{max_{\forall k=1,\ldots,N}\{dtw_{j,k}\}\} \tag{3.15}$$

where $dtw_{j,k}$ represents the warping distance amongst time series (steps) j and k. According to equation (3.15), The pattern that is going to be used as template is the step that minimizes the distance to the furthest of the N-1 remaining steps in the collection. The matching radius ρ_{rep} is determined as $\rho_{rep} = max_{j\in[1,\ldots,N]}\{dtw_{j,rep}\}$. Any new pattern exhibiting a distance to the template rep lower than ρ_{rep} will be considered as a valid step, while otherwise it will no be considered as a step.

Improving the model

Finally, when one template is not enough, it is possible to improve the model after collecting false positives. In this case, the user is asked to move the arm trying to get our application counting steps but without walking at all. On this way, a second set of M false positives is collected. Using the two sets: 1) the N true positives and, 2) the M false positives, the model is improved (Algorithm 2). The application of this algorithm will issue a new model formed by the minimum number of templates (each one with its own radius), so that all the N initial steps are properly recognized but none of the M false positives is misidentified.

3.4.2 Shape Based. Results.

In this section we describe the results of the shape based approach for step detection. We have carried out two experiments in order to analyze its performance.

Experiment 1

In the first experiment the model is obtained from noisy data, i.e., the initial set of steps to obtain the model are collected under a wide variety of circumstances: the user walks with the mobile in his hand, swings it, simulates a phone call, keeps the mobile in the pocket and walks with it, gesticulates while walking, puts the mobile facing upwards in the palm of his hand, touches the screen, etc. This is done, without following any kind of regular pattern and

// Set of true positives
$TP = \{A_p[1], A_p[2], ..., A_p[N]\}$;
// Set of false positives
$FP = \{A_{fp}[1], ..., A_{fp}[M]\}$;
$templates = \{\}$;
// Initialize the set "left" as a copy of the set of true positives
$left = TP = \{A_p[1], A_p[2], ..., A_p[N]\}$;
while $cardinal\_of\{left\} \neq \emptyset$ **do**
  // Select the "true" positive with the highest number of neighbours belonging to the set TP
  // A true positive "A" is considered neighbour of B when the DTW-distance amongst A and B is less that the distance from A to any false positive
  $w = arg\,max_{i \in TP}\{cardinal\_of\{A_p[j] \in left \,|\, dtw_{i,j} < dtw_{i,l}, \forall l \in FP\}\}$;
  // Set the validity radius of w
  // Half way amongst the distance to the furthest neighbouring true positive and the distance to the closest false positive
  $R_w = \frac{max_{j \in left}\{dtw_{w,j} | dtw_{w,j} < dtw_{w,l}, \forall l \in FP\} + min\{dtw_{w,l}, \forall l \in FP\}}{2}$;
  // Increase the set of templates with w and its radius
  $templates = templates + \{(w, R_w)\}$;
  // Remove w and its neighbouring true positives from the set $left$
  $left = left - \{w\}$;
end

Algorithm 2: Improving the model

during 100 steps. Once this initial set of steps are collected for training, then first model is obtained and tested when the user follows a particular 5-stage-sequence: a) 20 steps walking with the mobile in the left hand, b) 20 steps swinging the mobile, c) 20 steps with the mobile in the ear (simulating a phone call), d) 20 steps during which the user keeps the mobile in the right pocket and walks with it in the pocket (the user does not stop walking at any moment), e) finally, after these 80 steps, the user stops and removes the mobile from the pocket (no steps should detected here) and checks the screen of the mobile to see how many steps have been detected. Obviously 80 steps is the desired output. Table 3.1 (first column) shows the output of the model when it is formed by only 1 template. In general we clearly see that our application

Table 3.1: Results obtained for five different experiments following the 5-stage-sequence described in the text.

steps counted	false positives	number of templates after refinement	steps counted
101 (80)	57 FP	22T	80(80)
96(80)	51 FP	21T	81(80)
103(80)	44 FP	15T	84(80)
95(80)	57 FP	20T	79(80)
95(80)	52 FP	22T	75(80)

counts a higher number of steps than the desired output. After this, and for each one of these experiments we carried out the third stage (model refinement). In this case initially the user collects false positives without walking and by moving the mobile in the arm sideways, up and down, keeping the mobile in the pocket, touching the screen, etc. The second column of table 3.1 shows the number of false positives collected in each experiment. With this set, the model is refined adding new templates (third column of table 3.1). This model is tested once again following the 5-stage-sequence aforementioned, the number of steps counted this time is shown in the last column of table 3.1. As we can see the results are very accurate, as the numbers of steps that have been counted with our application are very close to the real case.

Experiment 2

In this case during the initial collection of 100 steps the user walked with the mobile placed in the palm of his left hand.After this, then the model is obtained and tested (Table 3.2). We run 10 experiments. During the first five experiments we observe the output of the model when the user follows the 5-stage-sequence described in the previous subsection.In this case we observe that the model misses a significant number of steps (second column of Table3.2). Nevertheless the model is able to recognize perfectly the steps given by the user when the mobile is in his palm (columns 4 and 6 of Table3.2, the desired output is 50, we tested the model when is formed by 1 template (4 column), or after correction(column 6). Nevertheless, if we correct the model once again and test it with the 5-stage-sequence, the number of steps missed is even worse than before (column 8). The reason is due to the fact that the model is tuned to recognize a specific way of walking. Hence, when the model detects an step it is correct in all cases (low number of false positives, it filters out all the relatives movements of the arm), but it also misses too many steps when the user moves in a different way from what it is

Table 3.2: Results when the model is tuned to recognize a specific way of walking

	output	exp.	output(1)	false pos.	output(2)	false pos.	output(3)	templates
Exp 1	78 (80)	Exp 6	49(50)	44 FP	55 (50)	20 FP	46 (80)	9T
Exp 2	59 (80)	Exp 7	49(50)	15 FP	49 (50)	18 FP	40 (80)	5T
Exp 3	75 (80)	Exp 8	50(50)	166 FP	49(50)	97 FP	35 (80)	10T
Exp 4	78 (80)	Exp 9	50(50)	4 FP	49(50)	2 FP	63 (80)	2T
Exp 5	62 (80)	Exp 10	50(50)	6 FP	49(50)	9 FP	50 (80)	4T

expected and to what the model has been trained for (high number of false negatives).This can be useful for specific applications like guiding inside buildings, where the expected position of the mobile might be restrained, but for the general case is better to follow the procedure described in the previous experiment.

3.4.3 Feature Based.

In this section we describe our proposal for feature based distance estimation.

Since we want no limitations about *where* or *how* the user carries the mobile, our proposal will estimate whether the person is walking before actually counting the steps. This will allow filtering chunks of useless signal and thus discarding false positives common in these kind of approaches. Obviously, with the proposal described in this section, we tried to achieve a reliable and robust classification, able to achieve high performances regardless of how the device is being carried or its orientation. Nevertheless, we are aware that an idealistic 100% reliability is not feasible, which is why we also aim for a error-symmetric prediction (similar number of false positives and negatives). If this is true, these errors can cancel each other, so that the number of steps predicted by our proposal in a time interval will be very accurate. This will make our proposal very appropriate for indoor local navigation.

Figure 3.3 provides a schematic representation of our complete proposal for pedestrian displacement estimation. This figure represents the block *Movement Estimation* shown in the figure 3.1 from section 3.1. We can easily count the steps given by a person applying an Peak Valley detector to the accelerometer signal. This would work perfectly in ideal conditions in which the user walks normally with the smartphone in its hand, however, the normal smartphone use in real world situations can cause accelerometer signals similar to those that appear when walking, even when the user does not move. Because of this we need a stage of step validation in order to discard false positives.

Thus, we use an Enhanced Peak Valley in order to detect "step candidates", i.e., a segment of signal that might reflect a step given by the person carrying the mobile. In parallel, supervised learning and Bayesian filtering are being used to recognize the walking activity. Hence, a step is validated and counted in those circumstances in which the detection of a step candidate coincides with the positive identification of the walking activity in the same signal segment. Next, we provide details of each one of the parts that make up our proposal.

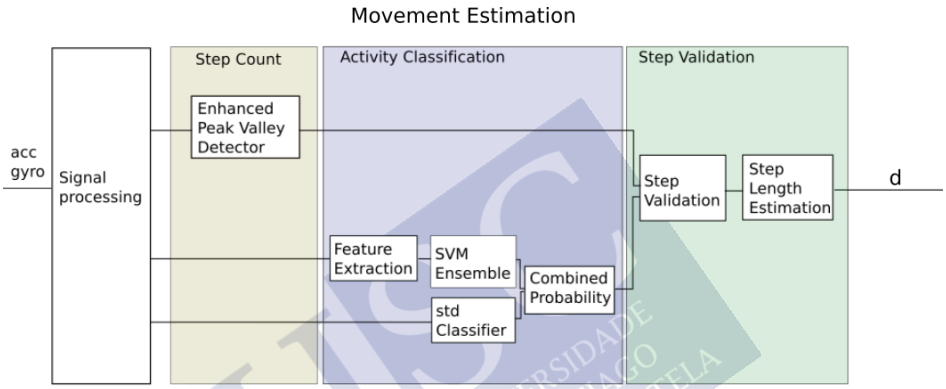


Figure 3.3: Our proposal to Pedestrian Dead Reckoning in a positioning system.

Lineal Vertical Acceleration

Accelerometer readings are highly dependant of the orientation of the device. In order to estimate the displacement we need to use a signal independent of the orientation of the device. Even though some approaches use the magnitude of the acceleration[28, 29] in order to achieve this orientation independence, it is also more failure prone in the presence of movements that have nothing to do with walking, thus increasing the rate of false positive detections. Because of that, we use the vertical linear acceleration instead.

Since we know the attitude of the phone (Section 3.3.2), we can now obtain the vertical component of the linear acceleration experienced by the mobile in the Earth reference system. To do so, we project the acceleration to the vertical axis, and remove the gravity force:

$$\vec{lacc}_t = {}^S_E q_t \otimes {}^S\vec{a}_t \otimes {}^S_E q_t^* - \vec{G}, \tag{3.16}$$

where ${}^S\vec{a}_t$ is the vector that arranges the accelerometer readings. The first term of the previous equation (Equation (3.16)) represents the projection of the information gathered by the ac-

celerometer into the Earth frame. $\vec{G} = (0,0,0,9.81)$ is the vector representation of the gravity force in the Earth frame. The vertical linear acceleration, $lacc_{z,t}$ (i.e., the vertical acceleration experienced by the mobile once the gravity has been removed), is the last component of the vector represented in Equation (3.16), $\vec{lacc}_t = (lacc_{0,t}, lacc_{1,t}, lacc_{2,t}, lacc_{3,t} = lacc_{z,t})$.

Inertial Signal Characteristics

At this point, we have isolated the vertical component of acceleration and eliminated the effect of gravity from it. When a person walks carrying a phone in its hand, and in the absence of other stimuli, the signal received (i.e., the vertical component of the acceleration in the Earth reference system) is often similar to a sinusoidal pattern shown in Figure 3.4a. Consequently, the signal shown in this figure can be considered as the characteristic signal generated by a person walking at a relatively low frequency [30]. In this signal, each step taken by the person is identified by the segment formed by a local peak followed by a valley. Hence, in this scenario, it might seem that a Peak Valley algorithm should be enough to identify the steps walked by a person. However, things usually are far from this ideal case, since there is a problem of perceptual aliasing, i.e., very similar signals to the one shown in Figure 3.4a can be obtained when the mobile phone is being used in a natural way, but when the person is not walking at all (Figure 3.4b). This makes the identification and segmentation of the signal into walking versus non-walking segments a difficult problem. Section 3.4.3 describes the way we have addressed this issue.

In order to achieve a satisfactory identification of steps, the vertical linear acceleration should be centered in zero as seen in Figure 3.4, but some devices may have a bias in the readings of the accelerometer that might prevent this centering. Non centered signals can lead to errors in the step detection algorithms. To ensure that this does not happen in our case, we apply a high pass filter to center the signal and thus remove any possible DC offset:

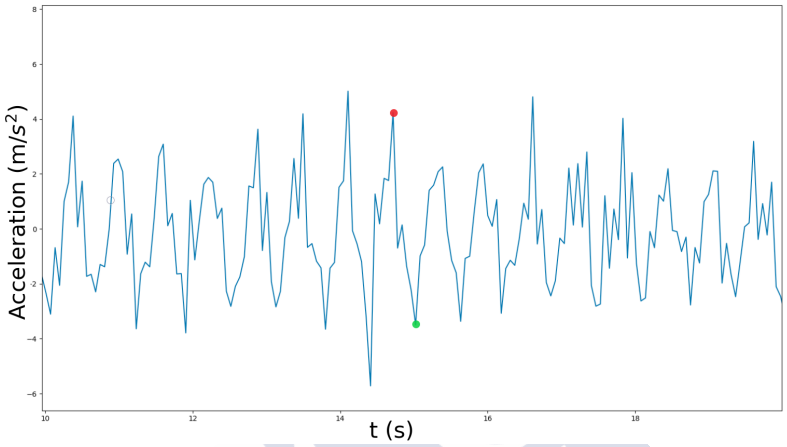
$$lacc^h_{z,t} = \alpha_h lacc^h_{z,t-1} + \alpha_h(lacc_{z,t} - lacc_{z,t-1}), \tag{3.17}$$

where $lacc_{z,t}$ is the input (vertical linear acceleration signal), $lacc^h_{z,t}$ the filtered signal, $\alpha_h = \frac{RC}{RC+\Delta t}$ and $RC = \frac{1}{2\pi f_c}$ and f_c the cutoff frequency (0.5Hz).

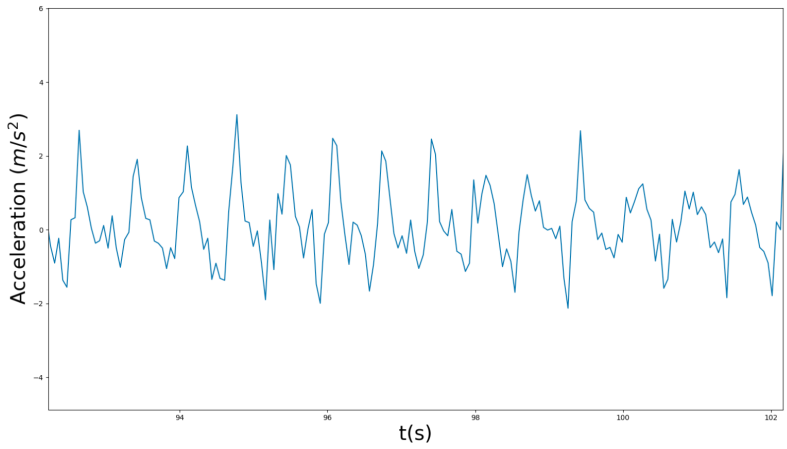
Finally, we apply a low pass filter to remove the noise from the signal:

$$lacc^l_{z,t} = \alpha_l lacc^h_{z,t} + (1-\alpha_l)(lacc^l_{z,t-1}), \tag{3.18}$$

where $lacc^h_{z,t}$ is the input (i.e., the already centered acceleration after applying Equation (3.17)), $lacc^l_{z,t}$ the filtered signal, $\alpha_l = \frac{\Delta t}{RC+\Delta t}$ and $RC = \frac{1}{2\pi f_c}$.



(a)



(b)

Figure 3.4: Vertical linear acceleration sampled at 16 Hz. (**a**) signal obtained when the user walking while holding its phone; (**b**) signal obtained when the mobile phone is being moved by the user but without walking.

Step detection

Step or gait detection can be achieved in different ways. One typical approach is to use a peak detector [31] to identify events, like heel strikes, where the impacts of the feet are reflected in the vertical acceleration signal. Other approaches involve exploiting the cyclic and repetitive nature of walking, and hence using properties such as the signal auto-correlation [32]. In our case, we have used an enhanced Peak Valley detector that locates local extremes in the signal. In this case, a step corresponds to a segment of signal in which there is a peak (local maximum which exceeds a threshold) followed by a valley (local minimum below a threshold). On the other hand, the time elapsed from the previous detected step to the new step-candidate must be above a valid walking period for it to be accepted. This is due to the fact that humans commonly walk within a low range of frequencies [30].

However, this peak valley algorithm is susceptible to detect any motion produced within the expected range of frequencies, and hence is prone to commit false positives and, on the other hand, usually has problems detecting changes in the walking speed [33]. Due to this, we consider the use of another module responsible for filtering out necessary, and, in real time, those parts of the signal which reflect some kind of movement in the mobile, but which have nothing to do with walking. This is a challenging task due to the high perceptual aliasing (i.e., the existence of many signals very similar but caused by different movements of the mobile or human actions).

Walking Recognition

In this section, we describe a module aimed at detecting when the user is walking, and thus discard noisy signals and filter out the false positives detected by the peak-valley algorithm described in the previous section. This module is strictly necessary to reach a symmetric error: on one side, the number of false positives will decrease since a step will be counted if, and only if, it is detected by the peak-valley algorithm and the activity of walking is recognized in the same signal segment by this module. On the other side, the inclusion of this module will have the collateral effect of increasing the number of false negatives, as a consequence of the misclassification of signals (although, as we will see, the rate increase of false negatives will be low). As we will see in the experimental results, the number of false positives and false negatives will tend to be very similar, thus canceling each other and making this proposal very suitable for pedestrian dead reckoning and indoor positioning.

Bayesian Filtering ($p(walk_t)$)

We use supervised learning and Bayesian filtering [34] to differ walking from non-walking sequences in the signal. Bayesian filtering involves the recursive application of a *prediction* and an *update* stages.

- *Predict:*

  The predictive probability distribution that defines whether the user is walking in an instant t, can be computed by the equation:

  $$\overline{bel}(walk_t) = \int p(walk_t|walk_{t-1})p(walk_{t-1}|Z_{1:t-1})dx_{t-1} \tag{3.19}$$

  where $Z_{1:t-1}$ is the history of measurements up to the time $t-1$.

  Therefore, as we can see, we work with a very simple dynamic model in which the only important element is 2×2 transition matrix $p(walk_t|walk_{t-1})$, the elements of which have been learned by an inductive process using a training data set. This transition matrix basically introduces a certain amount of hysteresis in the process: i.e., if the user was detected as walking in the previous time interval, there is a not null probability that the user is still walking at the current instant, despite the fact that the activity might not be identified in the current signal segment. Only after a certain time-lapse without recognizing the walking activity does this module set the probability of walking as null. Something similar happens in the contrary case.

- *Update:*

  Regarding the *update* stage, the Bayes' rule will be applied to estimate the posterior distribution of the state x_t, starting from the current measurement y_t:

  $$bel(walk_t) = \eta\, p(z_t|walk_t)\overline{bel}(walk_t) \tag{3.20}$$

  where η is a normalizing constant. We write z_t to refer to the current observation. In our case, we work with the overlapping sliding window, i.e., segments of signal and not instantaneous sensor readings. This means that z_t represents the observed signal ($lacc_z$) in a window, while z_t and z_{t-1} represent two different windows that overlap.

To compute the probability $p(z_t|walk_t)$, we will merge an ensemble of SVMs (Support Vector Machines) ($p_{svm}(z_t|walk)$) together with a logistic estimator that works over the standard deviation of the signal ($p_{std}(z_t|walk)$).

$$p(z_t|walk_t) = p_{svm}(z_t|walk_t)p_{std}(z_t|walk_t). \tag{3.21}$$

In the following subsections we explain how we estimate $p_{svm}(z_t|walk_t)$ and $p_{std}(z_t|walk_t)$.

SVM Ensemble $p_{svm}(z_t|walk_t)$

Regarding the SVM, we opted for an ensemble instead of a single SVM. We work with an ensemble because we have an unbalanced training dataset, with more walking sequences than non-walking sequences; therefore, we decided to use Bagging, oversampling randomly and with replacement from the minority class [35]. The outputs of all the SVMs that are part of the ensemble were merged to get a probability value:

$$p_{svm}(z_t|walk_t) \propto \frac{1}{n}\sum_{i=1}^{n} z^i_{svm}, \tag{3.22}$$

where n is the number of classifiers in the ensemble, and z_{svm} represents the output of the ith member of the committee. The inputs of these SVMs are feature vectors computed over windows of 2.5 s that overlap 0.5 s.

Using feature vectors instead of raw data can reduce the number of input elements and improve the generalization ability. We performed a study to identify which features were the most relevant for detecting walking sequences from accelerometer and gyroscope data. First, we collected all the features relevant to activity classification from the literature both in temporal and frequency domains [36, 37, 38]. Then, we analyzed the relevance of each feature with Recursive Feature Elimination (RFE) [39]. The selected features in the time domain are:

1. The Signal Magnitude Area (SMA).

   It is computed as:

   $$SMA = \frac{1}{w}\left(\sum_{i=1}^{w}|acc^i_x| + \sum_{i=1}^{w}|acc^i_y| + \sum_{i=1}^{w}|acc^i_z|\right), \tag{3.23}$$

   where acc_x, acc_y and acc_z are the acceleration signals perceived from each axis of the accelerometer, and w is the size of the window.

2. Maximum value for each axis of the accelerometer and gyroscope.

3. Standard deviation for each axis of accelerometer and gyroscope.

4. Correlation between axes y and z of the accelerometer.

   Finally, the formula for the correlation is:

$$corr(acc_y, acc_z) = \frac{cov(acc_y, acc_z)}{\sigma_{acc_y}\sigma_{acc_z}}, \tag{3.24}$$

   where $cov(acc_y, acc_z)$ is the covariance and σ_{acc_y} and σ_{acc_z} the standard deviation in each axis.

Regarding the features in the frequency domain, after applying the Fast Fourier Transform (FFT), we work with:

1. the FFT frequency bins up to 4 Hz for each axis of the acceleration signal.

2. the Dominant frequency in each axis which can be directly estimated from the FFT bin with the maximum value.

3. the Power Spectral Entropy (PSE) [40]:

$$PSE = -\sum_{i=1}^{N} p_i \ln p_i, \tag{3.25}$$

   where p_i is the power spectral density function:

$$p_i = \frac{\frac{1}{N}|X(w_i)|^2}{\sum_i \frac{1}{N}|X(w_i)|^2} \tag{3.26}$$

   and where $X(w_i)$ is the FFT of a signal, w_i is a frequency bin, and N is the number of frequency bins. This feature can be interpreted as a measurement of the uncertainty in the frequency domain.

Standard deviation estimator. $p_{std}(z_t|walk_t)$

Despite obtaining good performance with this ensemble of SVMs, due to the difficulty of the task, we decided to use a logistic estimator to reduce the number of misclassification. The results shown in [33] support the idea of applying a threshold over the standard deviation of

the acceleration to identify walking sequences in most scenarios. Therefore, we constructed an estimator of the probability $p_{std}(z_t|walk_t)$ as logistic function of the standard deviation of the vertical linear acceleration $lacc_z$:

$$p_{std}(z_t|walk_t) \propto \begin{cases} \frac{1}{1+e^{-k(\sigma_{lacc_z}-\beta_0)}}, & \text{if } \sigma_{lacc_z} < th, \\ \frac{1}{1+e^{k(\sigma_{lacc_z}-\beta_1)}}, & \text{otherwise,} \end{cases} \tag{3.27}$$

where σ_{lacc_z} is the standard deviation of the vertical linear acceleration (computed over sliding windows of one second), k and β_0 and β_1 are constants set experimentally that define the steepness and midpoints of the curves respectively, and th is a threshold set in the midpoint between β_0 and β_1. The idea behind this function is that typical walking sequences should have a standard deviation within a certain range of values. The logistic functions ensure smooth transitions.

Finally, we combined the results obtained with both the ensemble of SVMs, and the logistic function of the standard deviation $p(z_t|walk_t)$, as shown in equation (3.21).

3.4.4 Feature Based. Results.

We want to evaluate the performance of our proposal in realistic situations. To this end, we carried out the following experiments: First, we evaluated the step identification and counting when different people—mostly unrelated with our research to avoid bias—moved in a completely free manner, hence in a natural way, regarding both the walking manner and also the way they carried the mobile phone.

In order to carry out these experiments, and analyze the performance of the proposal described in this thesis, we require a way of obtaining the ground truth of the steps given by the user, i.e., the real number of steps walked by the person carrying the mobile. In some of the experiments, a visual counting might be enough, but, in some other experiments, where many users take part, a data set is collected, and where not only do we want to know the number of steps but also their timestamps—time values corresponding to each step—the visual counting is insufficient, prone to errors. In these cases, we had to make use of an automatic strategy described below.

Obtaining the Ground Truth

As we pointed out,we are going to collect data to analyze the number of steps given, due to which we need a ground truth. Most articles in the bibliography evaluate the performance of their algorithms only taking into account the total number of steps detected per experiment. Nevertheless, in our case, we opted for labeling every single step, in order to analyze the performance of each part of our proposal, as well as the canceling of false positives and false negatives thanks to the symmetry in the error. Hence, in our ground truth, each step is perfectly identified together with its timestamp, i.e., the moment in which the step is given. One option to obtain this ground truth could be manual counting. However, this method is unfeasible and prone to errors if we want to perform many experiments involving different people moving freely. Moreover, manual counting would not allow us to get the timestamps of the steps in an easy way. There are commercial step-counting solutions that perform well when the user walks, even though some of them are still susceptible to detect false positives [41], or they involve sensorized environments that constrain the freedom of movement of the user [42]. Nevertheless, we want to emphasize that, even though there might be commercial solutions available that would be valid to get the ground truth, we still opted for building our own mechanism due to two reasons: (1) our solution will require attaching inertial sensors to the legs (like some of the commercial solutions); as we will see, this will allow the detection of the movement of legs of the users unambiguously, and hence will provide a very reliable ground truth. On the other hand, (2) thanks to the use of our own sensors and software, we can get precise information about each single step walked by the user, and synchronize the readings provided by the sensors attached to the legs, with the signal that is being detected in the mobile carried by the person in the experiment and which we can run the proposals described in this thesis. This synchronization will allow a deep analysis of our proposals.

Attaching inertial sensors (accelerometers) to the legs, unlike the personal smartphone that can be carried on the hand, the pocket, the backpack, etc., removes uncertainty regarding when the person is walking. When these sensors detect the presence of important forces, we know that are due to the direct movement of the legs. Therefore, these sensors give precise information of when the user walks by the fact of being located on the legs. We have developed a system that uses two extra smartphone-based IMUs in the legs instead of any other low-cost integrated IMU. We attach smartphones on the legs instead of other low-cost IMU sensors, as the mobile phone can be considered a computer that can facilitate some tasks, such as real-time signal processing or synchronization. Hence, we tied these extra devices

with sports armbands to the legs of the people who participated in our experiments, as shown in Figure 3.5.

(a) (b) (c)

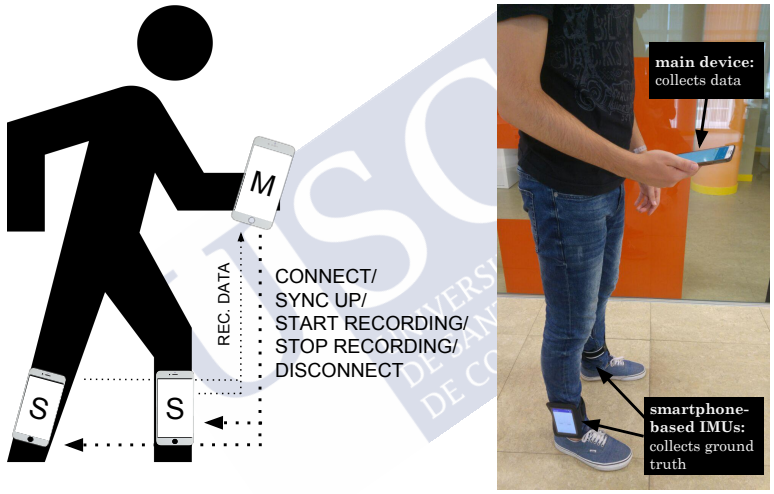
Figure 3.5: Sports armbands holding the mobiles of the legs. (**a**) Frontal view; (**b**) side view; (**c**) rear view.

We developed an Android application that was installed on all three smartphones. The devices worked in a synchronized manner storing information from the accelerometer, gyroscope and magnetometer sensors. The devices interact with each other following a master/slave communication protocol. Thus, the main mobile (the master) controls the other two devices in the legs (the slaves), as shown in Figure 3.6a. For all the devices to be synchronized as accurately as possible, all of them make a query to the same time server (`time.google.com`) using the library TrueTime [43] in order to get an atomic time, which allows later to compare information of different devices without problems of alteration of precedence.

Figure 3.7 shows a graphic representation of the ground truth over the signal of the vertical component of acceleration in the main mobile phone. Each peak-valley sequence in the ground truth signal is equivalent to one step, so it is easy to identify when the user is really walking and when the main device is experiencing accelerations due to actions different from walking.

The ground truth was calculated using only the data from the devices in the legs. First, we got two signals, one for each device fixed in each leg, using the acceleration module at each moment. In order to detect possible steps, we used a peak-valley algorithm [44], and then we matched the signals obtained with both mobiles on the legs. In this way, we can ideally detect each step in the signals coming from both legs, as shown in Figure 3.8a. Nevertheless,

depending on various conditions, such as the person performing the record, speed and way of walking, there are diverse situations that require special processing and attention: the signals detected with both legs might be shifted (Figure 3.8b), making it difficult to determine when it is the same or a different step. It is also possible to miss a peak in one of the legs due to an occasional weak signal. Finally, it can also happen that the peak of greatest intensity is recorded in the foot opposite to the one that has given the step. Nevertheless, we want to emphasize that our system has been designed to cope robustly with all these situations.



(a) (b)

Figure 3.6: Communication and synchronization between devices. (**a**) representation of the master-slave architecture; (**b**) volunteer obtaining data and its corresponding ground truth.

Given the large number of records performed in which the users walked under very different and greatly varied characteristics, we assume that there might be some steps missing in the ground truth. In order to somehow analyze to what extent this happens, we have also validated the ground truth through manual counting and visual inspection, counting steps of a significant part of the dataset (28%). This analysis allows us to limit the error committed in our ground truth in $\pm 2\%$, that is, between one and two steps per record.

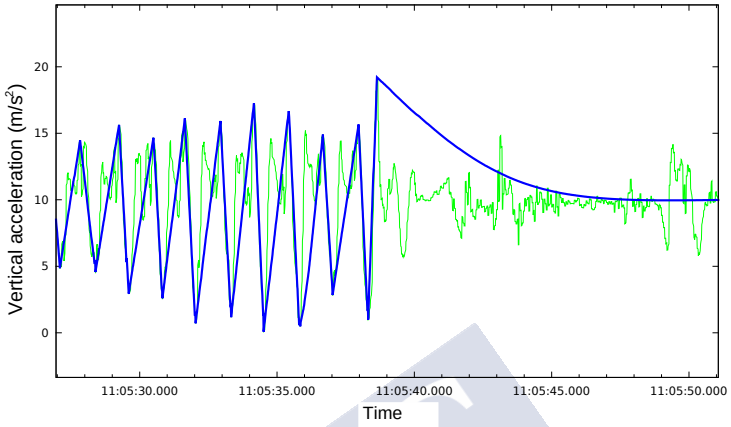


Figure 3.7: Graphic representation of the ground truth (thicker and darker line) over the signal of the vertical component of acceleration in the phone (thinner and clearer line).

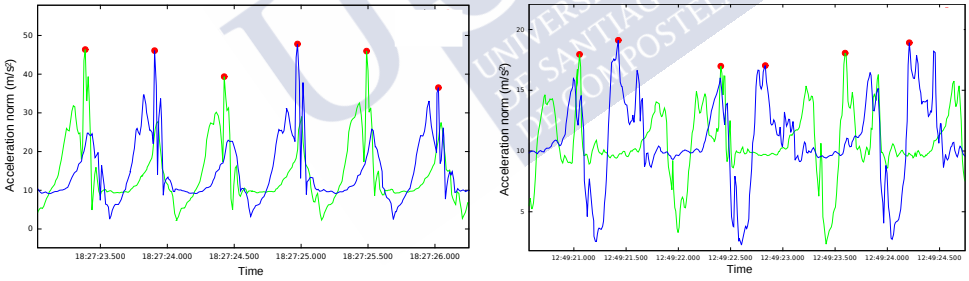


Figure 3.8: Peak detection (thick points) combining the signals of the two feet (blue and green lines) in an ideal situation (**a**) and a non ideal situation (**b**).

Step Detection and Walking Recognition Performance Analysis

In this first experiment, we wanted to evaluate the performance of our proposal when different people use it; to this aim, we have built a large dataset composed of a total of 140 records carried out by 75 different people. The vast majority of them (70, specifically) were volunteers with no links or connection with the research described in this thesis. We have proceeded in this way in order to ensure that the data were not biased. Thus, in each record, the participant

walked under natural conditions, freely or following some basic premises, while the inertial information of its movements was recorded and processed. Each volunteer walked, on average, about 2 min, giving around 110 steps. The dataset contains a variety of records that can be classified according to various criteria, such as the volunteer who performed the record, speed and way of movement: walking straight, climbing stairs, walking freely, etc. Another possible classification considers the position of the mobile, or the way in which it was carried: in the hand, in the pocket, in the backpack, etc. In all of the experiments in the dataset, the smartphone used was a BQ Aquaris E5 (BQ, Madrid, Spain). We labeled in this dataset each step taken with its timestamp using the method described in Section 3.4.4.

We have applied a matching algorithm to compare *step by step* the output of our proposal and the ground truth. We have analyzed the performance of the Peak Valley algorithm, also the walking recognition working as a classifier, and finally the complete system. Therefore, we have detailed information about true step detections (true positives), false positives, as well as false negatives. We can not compute the true negatives for the peak valley algorithm, since we speak about steps that do not appear in the ground truth and which are not detected by the Peak Valley algorithm. In the case of the walking recognition working as a classifier, the true negatives only reflect those false steps detected by the Peak Valley algorithm but discarded by the walking recognition working as a classifier.

We evaluated the performance of our algorithm dividing the dataset as follows:

1. Global performance, using the whole dataset.
2. Performance when the user walks freely with the phone in its hand.
3. Performance when the user walks freely with the phone in its pocket.
4. Performance in hybrid sequences, in which the user walks freely using the phone and changing its position (e.g., walks with it in the hand and then puts it back in the pocket).
5. Performance when the user barely walks but is using and moving the phone.

Table 3.3 shows the confusion matrices for each kind of experiment, while Table 3.4 shows the total amount of steps detected by the peak valley and by the complete system compared to the ground truth. To better understand these tables, we explain below the results obtained for the complete dataset (Table 3.3a and the first column of Table 3.4). The number of steps in the ground truth is 13,810. From these steps, the Peak Valley algorithm detects correctly

13,104, misses 706 real steps and counts 1640 incorrect steps. Therefore, the total amount of steps detected by the peak valley is 14,744, which are the candidate steps used as input for the classifier (the sum of the values in the confusion matrix of the classifier is equal to 14,744). From these candidate steps, the classifier detects 13,803 steps as correct, from which 12,544 are true positives and 1259 false positives. It also discards 941 steps, 381 of them being true negatives and 560 false negatives. The right side of the table shows the confusion matrix of the complete system. In this matrix, the steps detected as true (true positives + false positives) are the same as that in the classifier matrix. The false negatives are the sum of the steps missed by the peak valley (706) plus the the steps missed by the classifier (560). Finally, the number of true negatives is unknown, since it would be the sum of the 381 true negatives of the classifier plus the true negatives of the Peak Valley.

Peak Valley performs well when the user is actually walking but otherwise counts a high amount of false positives, as was expected. Our system discards part of the false positives, but it also misses some of the real steps, thus achieving an error symmetric approach. Our estimation of the steps missed by the ground truth leads us to conclude that both errors tend to be roughly the same. Since the error committed by the classifier is reasonable, the false positives and false negatives cancel each other out, thus achieving a high accuracy (as it is shown in Table 3.4). In this table, we can see how the total amount of steps counted by our complete system is significantly closer to the ground truth than the number counted with the Peak Valley algorithm only. Therefore, we think that this makes our proposal suitable for tasks such as Pedestrian Dead Reckoning and, in combination with other techniques, for indoor location.

Our system filters out noise in the signal, which is particularly relevant in sequences in which the user uses the phone without barely walking. However, it still has a significant amount of false positives, a problem that we will address in future research.

3.4.5 Final Reflection.

In this section we have carried out a detailed analysis of 2 different alternatives for identifying if the user is walking or not.

From these alternatives we have decided to use in our final system the feature based approach described in section 3.4.3. That is because shape based alternatives, while delivering high performance in our experiments, are more sensitive and less robust than feature-based alternatives. We have seen this in the operation of our system with the final users in the real

Table 3.3: Confusion matrices of the Peak Valley detector, the walking recognition working as a classifier over the candidate steps detected by the Peak Valley, and the Complete System, for each subset of data. Columns show the output of the system while the rows show the output of the ground truth (GT).

Complete Dataset									
		Peak Valley			**Classifier**			**Complete System**	
		True	**False**		**True**	**False**		**True**	**False**
GT	true	13,104 (85%)	706 (5%)	true	12,544 (85%)	560 (4%)	true	12,544 (83%)	1266 (8%)
	false	1640 (10%)	-	false	1259 (8%)	381 (3%)	false	1259 (8%)	-
Hand									
		Peak Valley			**Classifier**			**Complete System**	
		True	**False**		**True**	**False**		**True**	**False**
GT	true	5775 (92%)	213 (4%)	true	5630 (92%)	145 (2%)	true	5630 (90%)	358 (6%)
	false	311 (5%)	-	false	259 (4%)	52 (1%)	false	259 (4%)	-
Pocket									
		Peak Valley			**Classifier**			**Complete System**	
		True	**False**		**True**	**False**		**True**	**False**
GT	true	2848 (83%)	195 (6%)	true	2693 (83%)	155 (5%)	true	2693 (80%)	350 (10%)
	false	398 (12%)	-	false	320 (10%)	78 (2%)	false	320 (10%)	-
Hybrid									
		Peak Valley			**Classifier**			**Complete System**	
		True	**False**		**True**	**False**		**True**	**False**
GT	true	2547 (84%)	163 (5%)	true	2410 (84%)	137 (5%)	true	2410 (81%)	300 (10%)
	false	321 (11%)	-	false	255 (9%)	66 (2%)	false	255 (9%)	-
Not Walking									
		Peak Valley			**Classifier**			**Complete System**	
		True	**False**		**True**	**False**		**True**	**False**
GT	true	137 (26%)	22 (4%)	true	114 (23%)	23 (4%)	true	114 (30%)	45 (12%)
	false	368 (70%)	-	false	226 (45%)	142 (28%)	false	226 (59%)	-

world. In spite of having a complete dataset which comprises a high number of ways of transport of the mobile phone and different activities, the final system is used by thousands of people and, right now, it is impossible for us to contemplate how all of them will carry their phones or their peculiarities.

Table 3.4: Total steps in the ground truth, detected by the Peak Valley (PV) detector and detected by the whole system (PV with the walking recognition working as a classifier).

	Total Steps				
	Dataset	**Hand**	**Pocket**	**Hybrid**	**Not Walking**
GT	13,810	5988	3043	2710	159
PV	14,744	6086	3246	2868	505
PV + C	13,803	5889	3013	2665	340

3.4.6 Distance estimation (d_t).

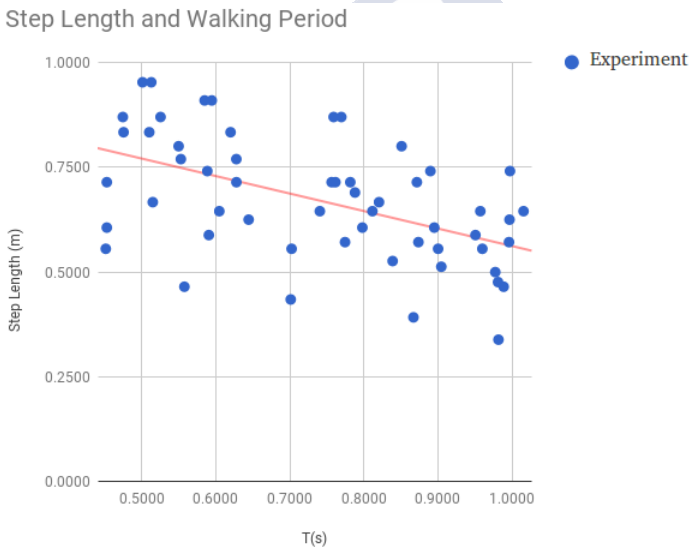


Figure 3.9: Relationship between walking period and step length.

At this point, we already have information about the steps the user is taking and an a probability of walking. However, we still need to estimate the length of the steps (l) in order to compute the displacement of the user.

Some authors estimate the step length as a function of the vertical acceleration (inverse pendulum) [30], however this requires the accelerometer to be placed in a fixed position at the hip. Other authors have demonstrated that the step size depends on two main variables, the height of the user and the walking frequency [45, 31]. In our case we have decided to use a

function that depends only on the step period, given that we have obtained good results, and this way we avoid being intrusive with the user asking him to enter personal data.

Due to the fact that there is a relationship between the step length and walking frequency [45], we decided to model the length of the step considering only the walking period [31]. Estimating this period from consecutive valid steps is straightforward. Nevertheless, for the matter of robustness, we work with the average of the last N detected steps. To get the relationship amongst the walking period and the step length, we asked several people to walk along a 20 m hallway at a constant speed. We instructed them to repeat this walk at different paces. For each record, we calculated the average step length (from the total distance and the number of steps walked). We also estimated the average walking period. Figure 3.9 plots both variables for the data obtained experimentally showing that there is a clear relationship between them. Finally, we applied Linear Least Squares to relate the step length with the walking period.

Thus we estimate the step length as a function of the walking period:

$$l = f(T) = \beta_0 + \beta_1 T \tag{3.28}$$

, where l is the resultant step length, T is the estimated walking period and β_n are constants that define the line fitted to the data shown in figure 3.9.

Thus, when the peak-valley (Section 3.4.3) detector detects a step, we estimate the distance traveled by simply applying:

$$d_t = l_t \cdot bel(walk_t), \tag{3.29}$$

where l_t is the length estimated for the step, and $bel(walk_t)$ is the probability of the user being walking estimated in Section 3.4.3 in Equation (3.20). Therefore, the system will count only steps when the peak-valley algorithm detects an step and the walking recognition confirms that the user is moving.

3.4.7 Results Distance Estimation

Besides the analysis of the performance of our proposal when recognizing steps, we also carried out experiments to evaluate the accuracy of the estimations made by our system regarding the distance walked by the users.

We run 20 experiments with 10 users in which they had to walk along a predefined path of 44 m. This path is shown in Figure 3.10a. During the experiment, there were marks on the ground every two meters to indicate to the users the path that they had to follow Figure 3.10b. The users walked following the route and carrying a phone that recorded the data. We run our system using these data and compared the distance estimated with our proposal versus the ground truth. It should be noted that the ground truth may have an error of approximately ± 1 m, due to some possible experimental artifacts: the last step taken by the user does not have to necessarily coincide with the last mark on the floor, and there can be a little deviation between the path followed by a person and the one marked on the ground, etc. The device used in these experiments was a BQ Aquaris E5.

Figure 3.11 and Table 3.5 show the results of the experiment. In Figure 3.11, we can see a boxplot with the distances estimated by our proposal. The blue line represents the ground truth. It can be seen that the median is slightly above the ground truth of 44 m and the percentiles Q1 and Q3 only deviate from it around 2 m. There are two outliers that identified a distance greater than 10 m above the real one. It is interesting to note that both belong to the same person.

Table 3.5: Statistical values corresponding to the distances estimated with our proposal.

	Distance (m)
Average	45.11
Standard Deviation	3.31
Maximum	53.8
Minimum	40.38

(a) (b)

Figure 3.10: (**a**) path followed by the people taking part in the experiment aimed for the analysis of the performance of our proposal at estimating the distance travelled by a person walking; (**b**) marks on the ground every 2 m placed to indicate the path that must be followed during the experiment.

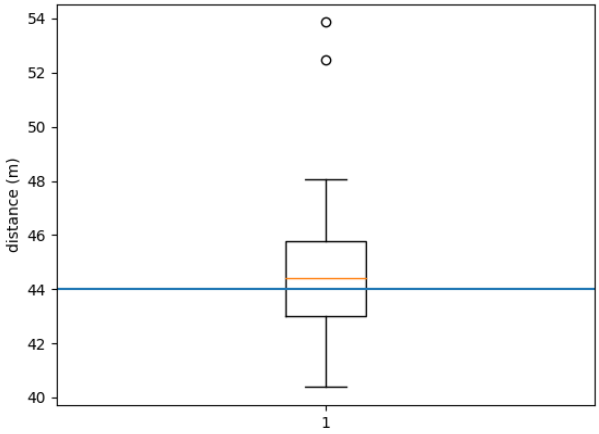


Figure 3.11: Boxplot with the distances estimated by our proposal for the 44 m long path.

As it was previously shown in Figure 3.9, there is a relationship between step length and the walking period, which is taken into a account by our model. Nevertheless, our proposal does not consider the influence of other variables like the user height, etc. The only way we foresee including those other variables is by means of personalized models that can integrate particular information of the user carrying the mobile. Nevertheless, considering the outcome of this experiment, our proposal performs well enough and it does not seem necessary to include this other information to improve the step size estimation.

Analysis of the Independence of Our Proposal to the Hardware Being Used

In this experiment, we will study the impact of the hardware—model of the mobile phone being used—in the performance of our proposal. Hardware differences generally involve different acquisition frequencies, differences in the amount of noise in the signal, and some devices can even have a bias in one or more of the sensor axes. These signal differences should not affect our algorithm thanks to the subsampling and filtering of the signal, the use of robust features, an ensemble of classifiers or Bayesian filtering, etc., with which our proposal should generalize well. However, we performed experiments with different smartphones, in order to analyze their impact on the outcome of our proposal.

We took different models of smartphones at the same time and walked while running our system. If our hypothesis is correct, the number of steps detected by each smartphone should be very similar for each experiment.

We performed four experiments using five terminals at the same time. In the first two, we carried all of the telephones in the same hand, and, in the next two, in the pocket. In all of them, we took 60 steps. The results can be seen in Table 3.6. For the experiments in the hand, there are hardly any variations. In those experiments in which the mobiles are placed in the pockets, experiments (3 and 4), there is a slightly higher variation regarding the number of steps counted with our proposal. Nevertheless, even in this case, the difference is not important, since the maximum difference between the steps counted for the different mobiles is three steps. Looking at the results, we can say that there are certain differences, especially when the mobile phone is in the pocket, but they should not represent a serious threat.

Table 3.6: Steps detected in four experiments running our proposal with different smartphones.

	Hand		Pocket	
	Experiment 1	**Experiment 2**	**Experiment 3**	**Experiment 4**
BQ Aquaris E5	58	57	58	60
Samsung Galaxy A2	58	58	59	59
Xiaomi Mi A2	58	58	59	62
Motorola Moto G6 plus	58	58	61	60
OnePlus 2	58	58	60	59

3.5 Displacement Estimation in a Vehicle (d_t).

In addition to the scenario of a person walking, we also considered that of a driver on a vehicle in a parking garage.

In order to estimate whether the vehicle is moving or not, we will isolate the vertical vibrations of the vehicle using the accelerometer signal. This has been proved to be a good indicator of the vehicle's movement [46]. Therefore, we estimate the vertical acceleration as we have done in the case of a person walking (Section 3.4.3, equation (3.16)).

To determine whether the vehicle is moving or not, i.e. the probability of movement, we work an simple heuristic rule that considers the last values of the vertical linear acceleration ($lacc_{z,t}$):

$$P_{movement} = \begin{cases} \frac{1}{th}\sigma_{lacc_z}, & \text{if } \sigma_{lacc_z}, < th \\ 1, & \text{otherwise} \end{cases} \tag{3.30}$$

Where $P_{movement}$ is the probability of movement, (σ_{lacc_z}) is the standard deviation of the vertical acceleration obtained on a window of 0.5 seconds and, finally, th is a threshold chosen empirically.

In general, we can assume that in parkings people drive within a limited speed range $[0, speed_{max}]$. Therefore, once we now the value of $P_{movement}$ it is possible to model the amount of displacement by applying a simple rule:

$$d = speed_{max} \cdot P_{movement} \cdot \Delta t \tag{3.31}$$

3.5.1 Results Vehicle Displacement and Orientation estimation.

We have analyzed whether our implementation of the EKF meets the required expectations, i.e., whether the quaternion estimated by the filter conveys enough information to determine when the car is moving (Eq. 3.30), as well as the change of orientation experienced by the car (Eq. 3.14).

With this purpose we have driven the car in an underground car park with the mobile attached to the car's dashboard. We have developed an Android app which used our EKF to compute at each instant the probability of the car moving and the change of orientation. In order to get the ground truth we recorded the experiment in video, from which we can extract precise information about whether the car is moving, as well as its heading. This video is synchronized with the mobile app, so that we can compare, at every instant, the outcome of the filter with the real information about the heading and movement of the car.

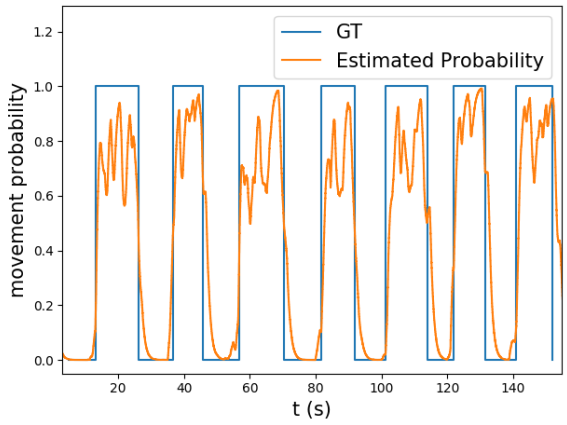


Figure 3.12: Performance of the movement estimation predicting whether the car is moving or not in an underground car park. The blue line represents the ground truth, and the orange line the probability of movement estimated with Eq. 3.30.

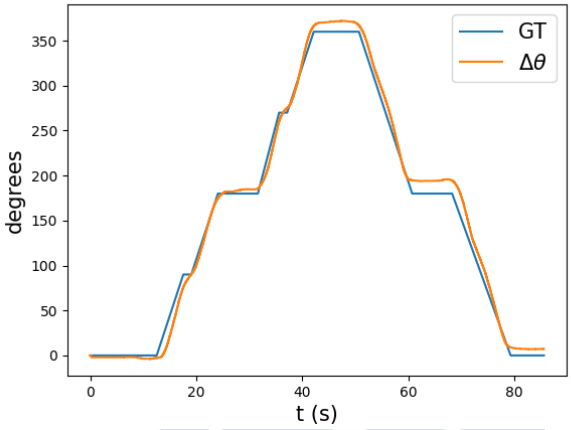
- *Movement Probability:*

  During a first part of the experiment the car stopped and moved several times on purpose (Figure 3.12). This Figure 3.12 shows the outcome of the experiment in an interval of almost three minutes. As we can see, the probability resembles quite precisely the

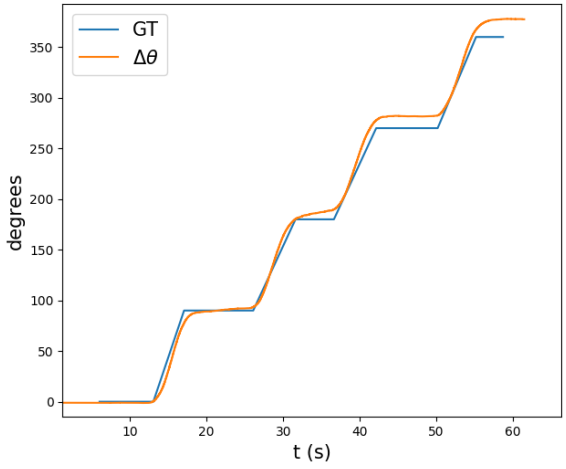
ground truth, i.e. there is no situation in which our estimation says that the car is moving when it is not, or the contrary. The probability of the car moving is always high when the car is effectively moving, or near zero when the car is stopped.

- *Orientation:*

  In a second part of the experiment, we computed the accumulated change of orientation $\Delta\theta$. Figure 3.13 shows how reliable the outcome of our estimation is, in comparison with the ground truth, for an interval of one minute. During the experiment the car was driven along different lanes thus requiring multiple turns. We must be aware that, as we will see in the next sections, this estimated change of orientation will be merged with our sources of information (by means of a particle filter), thus avoiding the drift in the inertial sensors of the mobile phone. Moreover, the change of orientation detected with our EKF will be corrected by the particle filter in intervals much shorter than 1 minute, therefore Figure 3.13 shows the performance of the EKF in situations that go much further than what our complete system will require.



(a)



(b)

Figure 3.13: Performance of the EKF when estimating the heading change, $\Delta\theta$, accumulated by a car moving along two different routes in an underground car park. The blue line represents the ground truth and the orange the output of the EKF.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 4

OBSERVATIONS

In chapter 2 we have explained that we use a particle filter to determine the position of the user at each instant. With this filter we combine information of the movement of the user, *motion model*, with observations of its position, *observation model*, in order to update and correct the initial distribution 2.3.

In chapter 3 we have described the motion model, and how we estimate the displacement and change of heading of the user from the sensors of the smartphone, both when a person is walking and when the motion is that of a vehicle.

In this chapter, we describe how we estimate the global position of the user (observations) from the sensors of the smartphone and how we integrate them in order to estimate our observation model. In our observation model we combine information from WiFi and Bluetooth Low Energy (BLE) signals, along with compass information and a occupancy grid map. WiFi and BLE provide a rough estimate of the devices position in the building (x, y, $floor$), the compass provides information of orientation respect to the north, (θ^N), and the occupacy grid map is used to discard estimations that fall out of the navigable areas.

4.1 Observation Model

The observation model defines the probability of receiving certain readings or observations from one or more sensors (Z_t), while being on a particular state s_t.

$$p(Z_t|s_t) \tag{4.1}$$

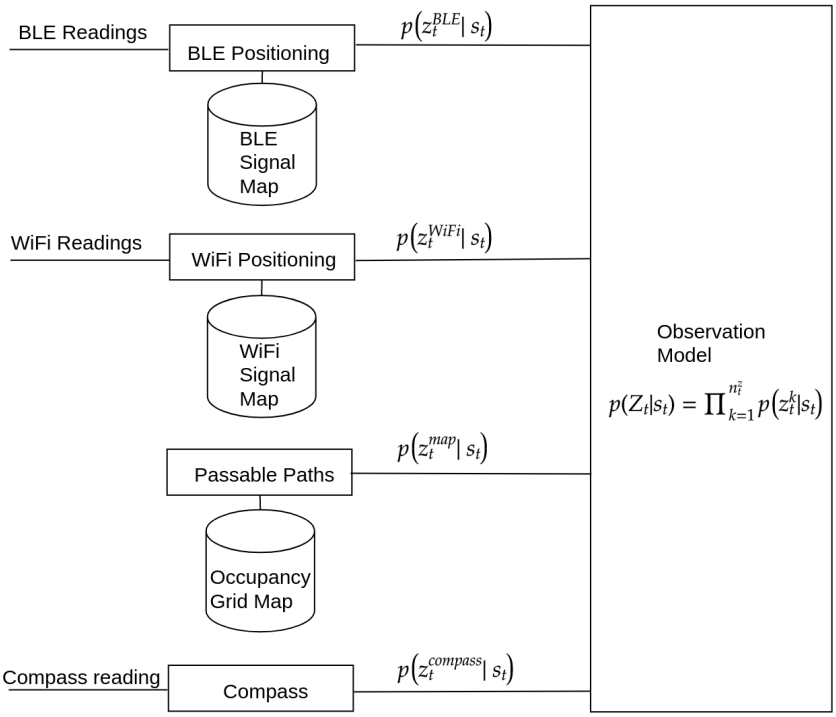


Figure 4.1: This figure represents the sources of information that we use as observations and the observation model

The most probable poses will be those with more similarity between the expected and received sensor data.

As shown in figure 4.1, we combine in our motion model information from WiFi, BLE, compass and an occupancy grid map. Given that these different sensor readings are not synchronous, one or more of these observations can be present at any given moment of time t. We use them in the update stage of the particle filter as they come.

Since our sources of information include different sensors, our observation model is the combination should be estimated as the joint probability of the measurements of all of them. However, given that the joint probability is not easy to estimate directly especially when the

sensor data is not synchronized, we approach it as follows [12]:

$$p(Z_t|s_t) = \prod_{k=1}^{n_t^z} p(z_t^k|s_t) \tag{4.2}$$

where n_t^z is the number of sensors available at an instant t, and z_t^k the observation for each one of these sensors. With these approach the number of sensors used as observations can be easily expanded or reduced as needed.

4.2 WiFi and BLE localization

Nowadays, in most buildings there are deployed a large number of WiFi access points (APs), in order to provide network coverage in the whole area of the building. In addition, there are often more than one network provider, such as in commercial environments, where each business usually has its own network. It is also frequent to receive WiFi signals from APs located in nearby buildings. The signal received from all these APs provide valuable information that can be exploited in order to estimate the location of the receiver. Each one of these APs has it own unique identifier and it is received with a certain power level from different locations of the building. A smartphone can listen to these signals, and with help of a radio map of the building, estimate the area in which it is most likely to be.

However, some smartphone models do not allow scanning WiFi networks, and some areas, such as car parks, have little WiFi coverage. In order to solve this issue, we can add BLE beacons for increasing the coverage area of the radio frequency signals and thus complement the WiFi signals. Beacons area less expensive and easier to deploy than WiFi APs. In addition some WiFi router models, are already capable of emitting a BLE signal analogous to that of the WiFi. Given that both WiFi and BLE are radiofrequency signals with similar characteristics, for the sake of simplicity we use them to estimate the location of the subject with the same algorithm. We consider them as different observations being those of WiFi z^w and those of BLE z^b, but the estimation of its probabilities is analogous. In this section we will describe the method that we use for both, and will refer to radiofrequency observations as z^r.

There are multiples approaches for radiofrequency positioning (proximity, triangulation, etc.) being the most successful those based on fingerprinting [4].

Fingerprinting techniques are based in the premise that, for each location in the space there is an identifiable signature, a set of measurements that are possible from that location.

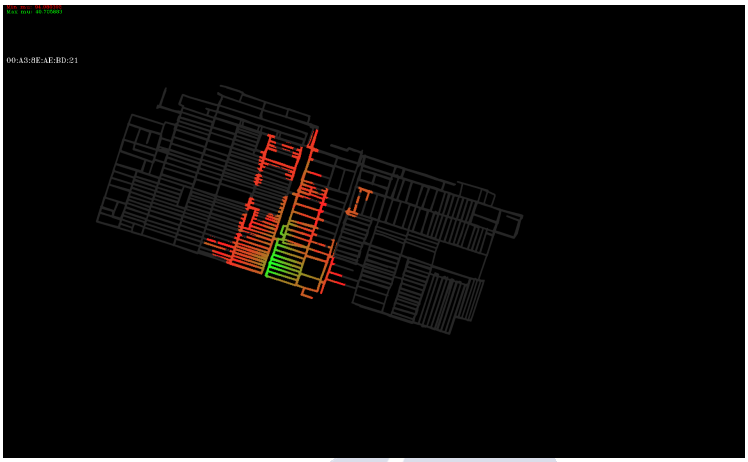


Figure 4.2: Example of a radio map for a WiFi access point.

In our case that signature is a set of WiFi or BLE readings, each one with its identifier and RSSI (Received Signal Strength Indication) power level.

In order to relate these signatures with a position, it is required a previous step of calibration. In this calibration step we have to build a *radio map* of each AP or BLE beacon perceived power at every coordinate of the building. A radio map is a probabilistic model created from a collection of data acquired at the same environment (where each datum is a vector of power levels, with the WiFi AP or BLE beacon identifier, and the associated physical coordinate (x,y,floor)). The map is a grid in which each cell $c_i, \forall i = 1,..,n$ has a fingerprint [4] consisting of a Gaussian distribution parametrized by a μ_i and a σ_i, both estimated using all the data obtained for the cell. One example of a radio map for an AP can be seen in figure 4.2.

Therefore we can use the power with which this signals are detected from the smartphone at an instant $z_t^r = \{z_t^r(1), z_t^r(2), \ldots, z_t^r(n)\}$ to build the desired probability $p(z_t^r|s)$ (block *WiFi positioning*, *BLE positioning* in Fig. 4.1). In particular we will compare the power of the readings received with the information stored in the radio map previously created. This way, these Gaussian fingerprints and the observed vector of signal powers scanned by the device at any instant t,z_t^r, allow the estimation of the conditional probability of each cell $p(z_t^r|c_i), \forall i =$

$1,...,n$ [12]:

$$\begin{aligned} p(z_t^r|s_t = c_i) &\propto \sum_{i=1}^{n_t^r} p(z_t^r(i)|s_t) \\ &= \sum_{i=1}^{n_t^r} \frac{1}{\sigma_i^r(s_t)\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{z_t^r(i) - \mu_i^r(s_t)}{\sigma_i^r(s_t)}\right)^2\right] \end{aligned} \tag{4.3}$$

where n_t^r represents the number of radiofrequency signals being detected and $z_t^r(i)$ represents the observed power of the i^{th} signal.

It is important to remember that the previous equation (4.3) is applicable both for WiFi and BLE observations. Thus we already have the WiFi observation model, $p(z_t^{WiFi}|s = c_i)$, and BLE observation model, $p(z_t^{BLE}|s = c_i)$.

4.3 Compass

Compass readings ($z_t^{compass}$) provide an estimation of the global orientation of the pose with respect to the north (θ^c) as explained in 3.3.3. This helps the particle filter to converge faster in orientation in the absence of movement.

Given the inaccuracy of the magnetometer measurements altogether with the uncertainty about the presence of electromagnetic interferences in the environment, we cannot be very restrictive about the contribution of this sensor to the observation model.

We define $p(z_t^{compass}|s_t)$ as a function of the difference between the state and the observed orientation, with maximum probability values in the interval $(-\frac{\pi}{2}, \frac{\pi}{2})$, that decreases linearly to $\pm\frac{\pi}{4}$.

$$p(z_t^{compass}|s_t) = \begin{cases} minValue, & \text{if } |\theta^c - \theta| > \frac{\pi}{2} \\ 1, & \text{if } |\theta^c - \theta| < \frac{\pi}{4} \\ 2 - \frac{4\cdot|\theta^c - \theta|}{\pi}, & \text{otherwise} \end{cases} \tag{4.4}$$

, where *minValue* is a value close to 0 defined empirically.

Additionally, we discard all readings in which the magnitude of the detected magnetic field is out of the normal range of the Earth Magnetic Field. We assume that this deviations may be caused by electromagnetic interferences, which can make impossible to identify the direction of the north.

4.4 Occupancy grid map

Finally, and to complete the observation model, we also use a map to evaluate whether a pose falls into a passable area or not (block *Passable Paths* in Fig. 4.1). The displacement of the particles in the prediction stage of the filter (eq.(2.9)) may provoke that some of these particles fall in locations that are not physically possible within the building. This map ($map_{x,y,floor}$) is a grid where each position has a binary value that defines whether each position is passable or not. Therefore, this map can be considered as another sensor that contributes to the observation model so that all the hypotheses (candidate positions) that are within non passable areas have zero probability:

$$p(z_t^{map}|s_t) = \begin{cases} 0, & \text{if } map(s_t) = 0 \\ 1, & \text{otherwise} \end{cases} \tag{4.5}$$

4.5 Observation model

Therefore, considering all these sources of information (WiFi, BLE, compass and the grid map), the final expression of the observation model is the following:

$$p(Z_t|s) = p(z^{map}|s_t) \cdot p(z^{compass}|s_t) \cdot \sum_{i=1}^{n} p(z_t^{WiFi}|c_i)\delta(s_t \in c_i) \cdot \sum_{i=1}^{n} p(z_t^{BLE}|c_i)\delta(s_t \in c_i) \tag{4.6}$$

In the absence of one or more sources of information, its corresponding part of the equation can be removed directly. Likewise, other sources of information can be added directly to the observation model in the future.

Chapter 5

Indoor Positioning and Guiding for Drivers

In the previous chapters we have described how it works our indoor positioning system. In this chapter we describe a final application with a complete solution of positioning and guiding for drivers in car parks.

Public spaces such as hospitals and malls, usually have large car parks where it is common to get stuck in traffic or where finding a parking spot can be daunting and time consuming. Similarly, corporate companies usually have rotating parking spots, where workers waste time looking for a free place where to park. Not only does this reduce the users' satisfaction, but it also has negative economic [47], health and environmental [48, 49] impacts.

Given the importance of this subject, there have been many attempts aimed at helping drivers and therefore minimizing the time spent looking for a parking spot. For instance, there are mobile apps with which users can reserve a free parking spot and obtain navigation instructions towards it. Other apps [50] combine information from different sources to get estimates of the time being spent looking for parking in different areas, and thus guide the users to the least crowded ones. Some of these apps can also help the user to find the place where he previously parked his car. Nevertheless these solutions rely on GPS positioning, which does not work indoors, and which in consequence are not suitable for many urban car parks located underground. In the specific case of indoor car parks, the most common solutions usually rely on visible elements that help drivers to know where they are (e.g. numeric or colored codes identifying different areas), or hardware installations that allow to identify free parking spots

at a glance (e.g. sensors placed over each parking spot with lights that change color if the space is free). Finally, some but few car parks are able to assign a free parking spot to users as they arrive. Nevertheless, all these solutions still exhibit important disadvantages such as the extensive work for deployment and maintenance, besides some of these solutions require expensive hardware and, finally, some of them are not as user friendly as they should be.

5.1 State of The Art

In the specific case of indoor guidance for drivers, Wagner et al. [51] provide indoor positioning in parking garages by using a system that combines information related with the movement of the vehicle, map matching, and GNSS (Global Navigation Satellite System); they estimate the speed of the vehicle from the rotation rate of the vehicle's wheels, and obtain its heading from the steering angle and a gyroscope. When the vehicle is outdoors the system estimates and corrects the error of the wheel radius, used to estimate the speed of the vehicle. It is indoors when dead reckoning and map matching are used to estimate the position of the vehicle. Nevertheless, as Wagner et al. [51] point out, the performance of their proposal is very dependent on the driving patterns, geometry of the parking garage, streets surrounding the parking, etc.

Kummerle et. al. [52] achieve autonomous navigation in a car park using multilevel surface maps and laser measurements. A particle filter is useful to localize the car within a 3D map combining wheel odometry, velocities and laser readings. The main drawback of this approach is that it requires expensive additional hardware and a thorough 3D mapping of the parking space.

Estimating the movement of a vehicle using only inertial information is difficult. Most proposals try to solve this problem by integrating the information provided by the accelerometer along time[46, 53]. However, this has several drawbacks, such as the high error usually caused by the sensor noise, vehicle's vibrations, etc [54]. Tan et. al. [55] attempt to estimate the speed of a vehicle using a smartphone, in particular they use the signal provided by the accelerometer to detect the same bumps when they are traversed by the front and the rear wheels; knowing the time between these front and rear bumps, together with the dimensions of the car, allow estimating the speed of the vehicle. The problem of this approach is that it requires previous knowledge of the vehicle length and therefore it is not easily scalable to multiple users in different vehicles, on the other hand it also requires bumps, which in many

parking garages are only at certain places, as in the entrances.

Yu et. al.[46] used the speed of the vehicle estimated with a smartphone to improve the localization of the vehicle when the GPS signal degrades or is lost for some time.

Perhaps the most similar work to ours is [53], where Liu et. al. use Hidden Markov Chains to merge RSSI information of the WLAN signals with motion dynamics information acquired from a smartphone. In this case the authors say that the average error of their proposal is under 2 m, but the experiments comprise only a short period of time, and no information is provided about the environment where these experiments have been carried out.

5.2 Guiding

We want the user to be able to find a parking spot using his smartphone while driving a car. Because of this, our proposal incorporates a module able to generate navigable routes within the car park, together with guidance instructions. This module will work with a directed weighted graph in which the nodes represent the intersections, the edges are navigable paths (these edges take into consideration the allowed direction when driving) and, finally, the weights represent the lengths of the paths.

5.2.1 Route Generation

We apply Dijkstra's algorithm [56] to compute the optimal route between the initial position of the vehicle and the desired destination. As a result we have a path in the graph representing the route: $G_{route} = \{N, E\}$, where N is the set of nodes $N = \{node_i = (x_i, y_i, floor_i) \forall i \in route\}$, E is a set of edges, where each edge has a weight w that represents the distance between the nodes that it connects $E = \{((node_i, node_{i+1}), w_{node_i node_{i+1}})\}$. The route is updated as the user advances, when the distance from the position of the user to the destination node is under certain threshold, the guidance finishes. However, if the distance from the user's position to the closest point on the route is above a predefined threshold, the route is recalculated, since the user may have gone in the wrong direction.

5.2.2 Textual guidance instructions

In order to provide an user-friendly guidance, our system provides textual and auditory instructions for the driver. The list of instructions $I = \{instruction_1, \ldots, instruction_n\}$ is generated based on the geometry of the route[57]. Each instruction is formed by a node (where the

instruction should be read), a type and a modifier: $instruction_i = \{node_i, type_i, mod_i\}$ where $node_i = (x_i, y_i, floor_i)$, $type_i \in \{turn, forward, changeFloor\}$ and mod_i has a different meaning depending on the type of instruction: direction to turn, distance to advance or number of floors to go up or down.

We iterate over each node to generate an instruction for each pair of nodes $node_i, node_{i+1}$:

1. If $node_i^{floor}$ is different of $node_{i+1}^{floor}$, we generate the instruction: $instruction_i = \{node_i, changeFloor, mod_i\}$ where $mod_i = node_{i+1}^{floor} - node_i^{floor}$.
2. If the floor is the same for the consecutive nodes $node_i, node_{i+1}$, we compute the angle α, between the edges $E_{node_{i-1}node_i}$ and $E_{node_i node_{i+1}}$.
   a) If α is below a predefined threshold, $type_i = forward$ and mod_i is set to the weight of the edge $E_{node_i node_{i+1}}$.
   b) Otherwise, $type_i = turn$. And mod_i is set to the direction of the turn based on α.

These instructions are translated into text to be displayed in the screen of the mobile as the user follows the route. On the other hand auditory instructions are obtained from textual ones by using an external text-to-speech module.

5.3 Experimental results

We carried out different experiments in order to evaluate the performance of our proposal. The experiments took place on a two floor underground car park.

We deployed BLE beacons on the walls and columns in each lane, every 14 meters, and at a height of approximately 3 meters as shown in Fig. 5.1. We configured the beacons to transmit every second at 0db, which according to the manufacturer allows 32 months of battery life.

We only defined as passable area the lanes and not the parking spaces. We assume that the users will drive within the lanes and not cut through parking spaces. If this happens, a position error could occur temporarily, but the system would end up converging to a correct position after receiving BLE readings. We do not add any other restriction to the displacement (e.g. the user can drive in any direction, even if it is forbidden). We can easily allow the parking spaces be passable areas (Eq. (4.5)), but this might slow down the convergence of the particle filter.

(a) (b)

Figure 5.1: (a) BLE beacons distribution. The red dots in the floor map show the locations of the beacons in the parking garage. The green area represents the highest longitudinal error we tolerate within a lane.(b) Photo of one of the circular beacons used, the approximate value of the radius of the beacon was 2.6 cm.

We developed two Android applications, one that implements our indoor positioning and another one that also facilitates the creation of the radio map by recording BLE readings while we pinpoint the positions in the map. The recorded data is sent to a server, where the radio map is generated and stored. When the positioning application is started, the radio map is downloaded from the server. With this data we can run our location system directly in the phone without further network communications. Fig.5.2 shows a screenshot of our positioning and guidance application. As it is already common in these type of systems, it shows the user's position and the route to the selected parking space. In our experiments we used two different mobile phones: a Samsung S7 and a LG G3.

Next, we conducted a series of experiments while driving at different speeds, using the proposal described in this thesis, and with the smartphone mounted in a holder on the windshield. In addition to these experiments, we carried out a test with 7 users in which they installed our Android application in their phones and were instructed to drive to a specific parking spot chosen randomly. All of them were able to arrive easily to their destination following the visual and auditory instructions.

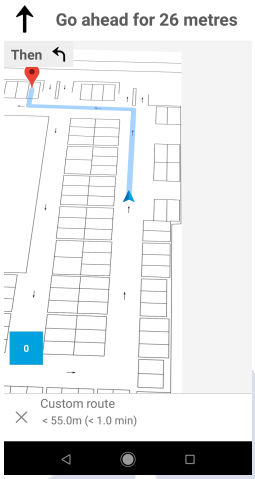


Figure 5.2: Screenshot of our Android application for positioning and guiding in indoor car parks.

5.3.1 Ground truth

We recorded all the experiments in video in order to get the ground truth and validate our system. In particular we divided the video sequences in frames separated by half a second and in those frames we obtained the real position of the car (by manual annotation and visual inspection of the frames). These ”real” positions were compared with the positions estimated by our proposal, in order to evaluate its performance. In general we considered three different types of errors: (1) floor errors, (2) lane errors (when the estimated lane, i.e. direction of movement does not match the real one), and (3) longitudinal errors (when the distance among the estimated position of the vehicle and the real one is higher than a certain threshold, in our case 10 meters). This threshold has been set considering the uncertainty in the estimation of the real position, the usual speed of the vehicles moving in the car park, as well as how long in advance our application provides instructions to the driver. Hence, differences below this threshold are unnoticed by the driver.

5.3.2 Convergence time

The first aspect we analyzed was the *convergence time*, i.e. the time elapsed since the instant in which the system is initialized to the moment in which it starts showing the position of

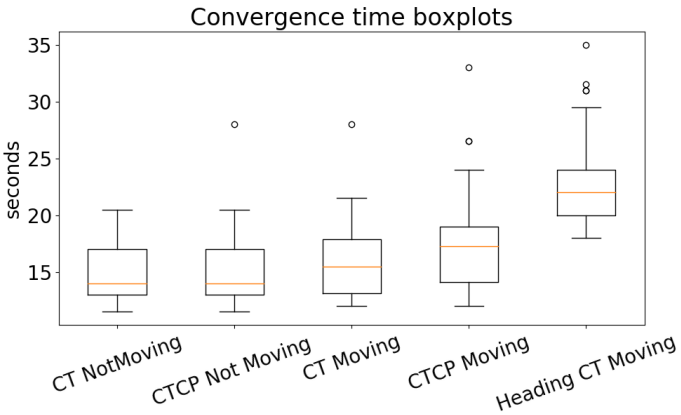


Figure 5.3: Boxplot representing the CT (convergence time) and CTCP (convergence time to the correct position) when the vehicle is moving and while it is stopped. It also shows the Heading Convergence Time when moving.

the vehicle, regardless of whether these estimations are correct or not. We also evaluated the convergence time to the correct position (CTCP). For this measurements we labeled as *correct* those frames in which two conditions were fulfilled: (1) on one hand the difference between the estimated position and the real one (obtained from visual inspection) was below 10 meters, (2) on the other hand the lane of the car park predicted with our system was also correct. Finally, we also analyzed the time required to estimate the heading θ to the the correct direction of movement (Heading CT). To evaluate these convergence times, CT and CTCP, we carried out several tests at two different situations: convergence without movement and convergence while moving. We performed 102 trials for the case of no movement and 96 for the case of movement. All starting positions were chosen randomly within the car park.

Fig. 5.3 and Table 5.1 show CT, CTCP for both cases: with the vehicle moving and not moving. In the case of the heading CT we only considered the scenarios with the vehicle moving, since without such movement the position can only converge in (x,y).

The CT when the vehicle does not move is between 10 and 20 seconds with a median around 15 seconds. This value is slightly higher when the vehicle is moving. In almost all the cases our system converged directly to the correct position,especially when the vehicle is stopped. Nevertheless, in those cases where there was some kind of mistake in the initial

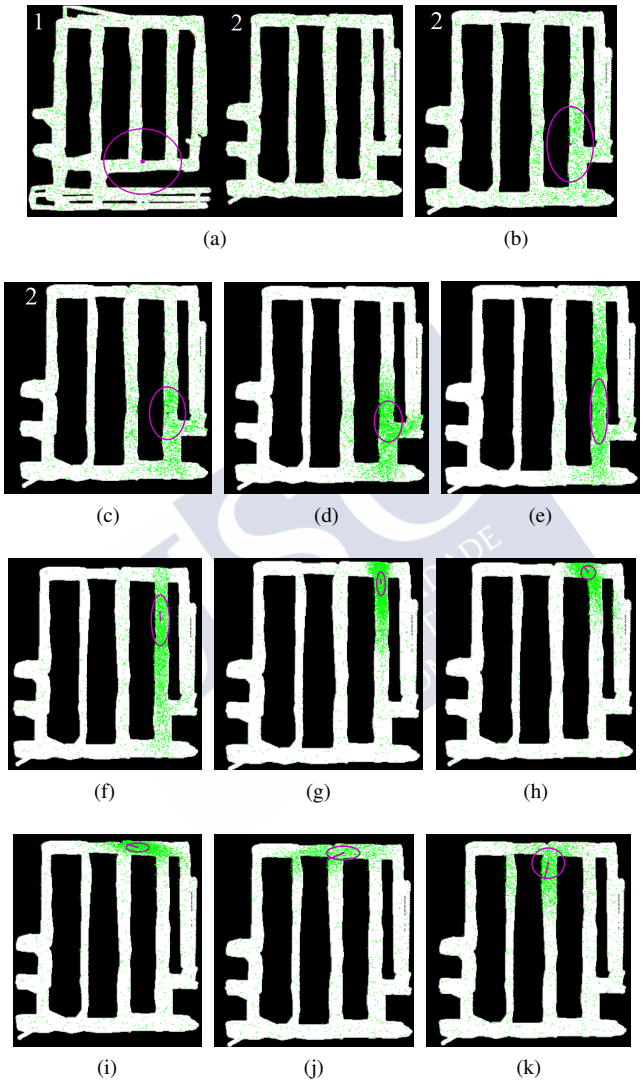


Figure 5.4: Particle clouds.This figure shows the distribution of filter particles during several iterations. The white space represents the passable areas, the green points are the particles and the ellipse the current estimate of the position of the vehicle. The size of the ellipse represents the confidence in the estimate –the smaller the size of the ellipse the higher confidence–, and the line within the ellipse represents the estimated direction once it has converged in orientation.

Table 5.1: Convergence (CT,CTCP) not moving and moving, and heading CT

	Convergence time	Average (s)	Std (s)	Max (s)	Min (s)
Not Moving	CT	14,79	2,38	20,5	11,5
	CTCP	15,20	3,02	28	11,5
Moving	CT	15,75	3,32	28	12
	CTCP	17,5	4,22	33	12
	Heading CT	22,96	4,02	35	18

estimation, our system required less than two seconds on average to correct it. Heading convergence took almost 33 seconds on average from start, and 5.4 seconds from the moment in which our system has already converged to a correct position.

Fig. 5.4 shows an example of the particles converging for one experiment. Fig. 5.4a shows the initial instant of time, in which all the particles are equally distributed throughout the map in the two floors. After receiving the first BLE readings, particles begin to converge, first to the correct floor (Fig. 5.4b) and finally to the correct lane (Fig. 5.4c-5.4d). Since the vehicle is moving, the particles move along the lane (Fig. 5.4e), reaching convergence in orientation in Fig. 5.4f. Figures 5.4g to 5.4k show the behaviour of the system when the vehicle turns.

5.3.3 Error analysis and comparison

In a second experiment, we estimated the positioning error of the system when different users drove freely around the parking garage. In particular we run 4 experiments; in each one of them the vehicle moved for 10 minutes in the underground car park. As it was pointed out in subsection 5.3.1, we recorded the experiments to get the ground truth, in particular we recorded in total 41 minutes and 42 seconds of video. We also recorded the data generated by each smartphone (eg. BLE scans, displacement estimation, and orientation) which allowed us to carry out offline analyses afterwards. Like in the previous experiment, we divided each recording in frames and labeled them as correct or incorrect. As it was already mentioned in subsection 5.3.1, we considered three possible mistakes: 1) Floor errors, if the position given by the system is in an incorrect floor, 2) lane errors, if the position estimated is in the wrong lane, and 3) longitudinal errors, if the lane is correct, but the distance from the estimated position to the real one is above 10 m as shown in Fig. 5.1.

We used these experiments to evaluate the performance of our proposal. In order to estimate the relative importance of the information extracted from the inertial sensors of the

Table 5.2: Fraction of time in a correct / error state, and with correct and undetermined heading for the complete system and using only the BLE.

Complete System

	Correct	Floor error	Lane error	Longitudinal error
Best	0.9264	0.0085	0.0085	0.0564
Average	0.8871	0.0089	0.0431	0.0606
Worst	0.8254	0.0183	0.0704	0.1975

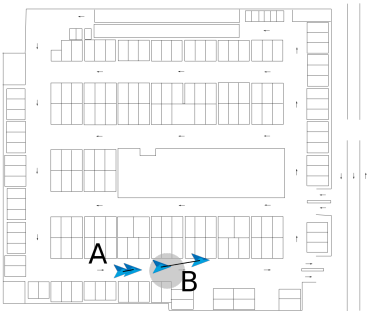
	Correct heading	Undetermined heading
Best	0.9235	0.0764
Average	0.9106	0.0893
Worst	0.8871	0.1128

BLE Maximum

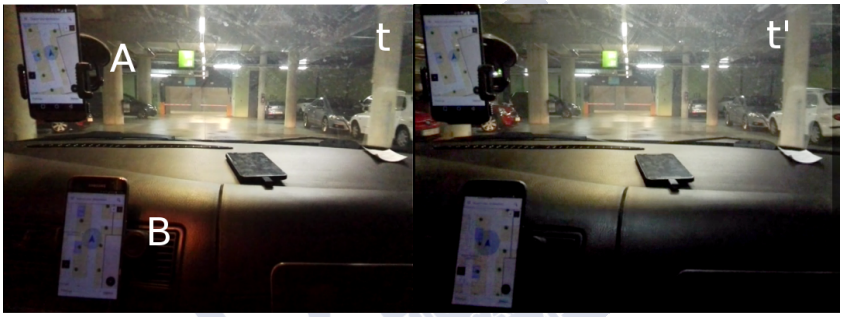
	Correct	Floor error	Lane error	Longitudinal error
Best	0.7704	0.0119	0.0833	0.1343
Average	0.6197	0.0320	0.1673	0.1810
Worst	0.4502	0.0643	0.2848	0.2006

mobile phone, we have considered the output of the whole system, and the output obtained using only the maximum of the BLE signal, i.e., the positions considering only the BLE information. In this second case (BLE Maximum) the position of the vehicle is estimated by the the cell of the map in which equation 4.3 takes the maximum value.

Table 5.2 shows the percentage of times in which the position estimated was correct or incorrect, considering the different types of error for the complete system and using only the BLE Maximum. Although the performance for the complete system shown in this table is of 89% (in average), we have obtained results with a performance of even 91% (in average) with a system working in real time and with an optimal combination of parameters adjusted "in the working place". Nevertheless, we preferred to show those results of table 5.2, as they reflect a more objective comparison of the performance, and which have been obtained after the analysis of the data recorded during the experiments. Obviously we only include the heading error in the complete system, because with the BLE Maximum there is no heading information. It is important to notice that the labeling of the change of floor was set at half of the ramp, and all the errors regarding this aspect in the complete system are just delays, i.e. the complete system never showed a position in a wrong floor. The percentage of correct positions acquired using only the BLE is significantly lower than that of the complete system



(a)



(b)

Figure 5.5: (a) Positions estimated by our proposal when it is running in two different mobile phones, A and B, and at different time instants t and t'. The real position of the car in both instants is within the gray area. (b) Images taken from the car, at time instants t and t'.

for all the experiments. In the particular case of the BLE, the positions are most of the time close to the vehicle's real location, but sometimes they may be unstable and jump between lanes. There are even some jumps between floors when the vehicle passes near the ramps probably caused by detecting beacons from the wrong floor.

For the complete system, on average the position is correct more than 88% of the time. Most of the errors are due to miscalculations in position in the correct lane. This proves that the error committed by our system is not equidistributed, i.e., there is no error when our system points out the floor where the vehicle is (probably due to the information provided by the BLE beacons), the error is also very low when indicating either the direction of advance of the car, or its position at intersections (the turns of the vehicle are likely detected by the inertial sensors). But the largest error is committed within the lanes (longitudinal error), especially

when the vehicle is moving fast. We restrict this error to less than 10 meters, nevertheless this is a conservative value and almost the worst case scenario. This worst-case-longitudinal error in a lane is not that bad if we consider the length of the vehicle itself (from 3 to 5 meters), or the final purpose of our system (guidance of a driver in an indoor car park), where the most relevant issue is to lead the driver to the correct lane.

Fig. 5.5 shows an example of how the threshold we have set to consider that there is a longitudinal error, is reasonable and has a low impact from the driver's perspective. Fig. 5.5.b shows two images taken from the car when it is at the same location but at different instants (t and t'), the real location of the car in both instances is represented by the round area shown in Fig. 5.5.a. As we can see in Fig. 5.5.b, inside the car there are two smartphones that are running our application and which therefore provide positioning information (device A at the top of the image and B at the bottom). Fig. 5.5.a shows with letters A and B the positions of the car estimated by our application in both mobiles, and at instants t and t'. As we can see, although the positions of the car at both instants t and t' are roughly the same from the driver's perspective (Fig. 5.5.b), the distance among them is of 2.3m according to mobile phone A, and 9.4m. according to mobile phone B. On the other hand, the average distance among the positions of the car at t and t' estimated by the mobile phone A and the mobile phone B is 12.4m. These discrepancies are mostly due to the initialization conditions as well as the differences in the hardware of the mobiles, nevertheless they have almost no impact on the driver.

Fig. 5.6 shows an example with two of the trajectories generated by our complete proposal. We also show whether our system committed any kind of error. Fig. 5.6a shows a trajectory with no positioning errors, while Fig.5.6b shows one of the worst scenarios we can find. In this second case, most positions are correct, except for one moment in which there are two estimations in the wrong lane. Immediately after these errors, the system corrects the position, providing the information about the correct lane, although during four timestamps the estimated position still stays a bit behind the real one (longitudinal errors). Commonly, most trajectories have no errors at all or some occasional longitudinal error, lane errors like the ones shown in Fig.5.6b are very rare.

As a final reflection, we also want to point out that we have not considered the outcome of the Particle Filter using only the map and the motion model (obtained from the inertial sensors on the phone), but without BLE information. This combination is not very appropriate not only due to the accumulative errors of the inertial sensors, but also due to the fact that both

(a) (b)

Figure 5.6: (a) Example of a route with no errors. (b) Route labeled with the kind of error committed by the system. Green dots represent correct positions, blue crosses positions with a longitudinal error, and red triangles lane errors.

the map and BLE are necessary to get information about the initial heading of the vehicle, and thus suppress those particles that have been randomly initialized with a wrong angle. With the map alone, this convergence is too slow and sometimes not possible due to the symmetries of the parking lot.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 6

INDOOR POSITIONING FOR PEDESTRIANS

In the previous chapters we have described how our indoor positioning system works. First we have made a description of the Particle Filter (chapter 2), to then focus on each of the parts that compose it (model of movement in chapter 3 and observation model in chapter 4.) In addition we have described two ways of operation, one for pedestrians, and one for vehicles.

In this chapter we show an example of the system deployed and working in a real environment for pedestrian localization. First we describe the environment and the installation process to continue with a section of experimentation and results in which we analyze the performance of the system.

6.1 Environment and deployment

Different environments have different physical characteristics and may have a different WiFi network infrastructure, which can affect directly to the quality of positioning and facilitate or make it difficult to estimate the user's position.

Next, we describe several characteristics of the environment that may affect to the accuracy that we can achieve:

- Large open areas vs. small, enclosed areas, such as corridors. In open areas, radio frequency signals propagate without obstacles, so the fingerprint at locations within those areas are generally similar between them. In enclosed spaces, such as corridors or small rooms, the signal perceived is more different between areas because of the walls and obstacles it encounters. Thus it is easier to achieve higher accuracy in position

estimation with RF fingerprinting in narrow aisles or small rooms than in large open spaces. In addition, in large environments the particles encounter few obstacles so that clusters are generated more dispersed than in narrower environments.

- Characteristics of APs:
  - Quantity. Generally, in environments with more APs, we can achieve a greater accuracy than in environments with fewer APs. This obviously has a limit, at some point the number of APs is so high that some of them are redundant. In environments with few APs, BLE beacons can be added to cover areas without sufficient information.
  - Power. Access points with more power are received from farther away, so they cover a larger area than those that have less power. This on the one hand is positive, given that more area can be covered with few APs, but it has the disadvantage that only with this type of APs the maximum accuracy obtainable is reduced, given that they are heard with similar power ranges in larger areas.
  - Distribution. The location of the APs has a direct impact on the quality of positioning in certain areas. If the APs are distributed in such a way that the received signals are very different between the different areas, the estimation of the position by fingerprinting will be more accurate.

When choosing an environment to evaluate the performance of our positioning system, it is necessary to take these aspects into account and choose a building that is sufficiently representative of the general case. In our case we have chosen a six floor shopping center of 500,000 square meters which has corridors, car parks, open spaces and squares with walkways at different levels. This building represents a typical scenario in which indoor location is necessary, and we believe that it is sufficiently representative of the problems that we can find in different environments.

The installation process is similar to the one described in Chapter 5, Section 5.3, for the case of vehicle positioning, with the main difference that, in this case, there is a good infrastructure of WiFi access points that we can use. In total, we receive readings from 672 different WiFi APs from inside the shopping center. Although the building has a good infrastructure of WiFi access points, we have installed BLE beacons as a reinforcement, to improve accuracy and robustness in potentially complicated areas,and at the same time to reduce the time that

takes for the system to detect the change of floor. Adding BLE beacons also allows us to support phones that do not allow scanning WiFi APs. We deployed along the building 44 BLE beacons in strategic positions such as the exit of lifts, or areas with little WiFi coverage such as car parks.

The first step to be able to provide indoor positioning is to create the WiFi and BLE radio maps, as well as the occupancy grid map that defines the passable areas. To generate this model, we have developed an application that allows you to scan the WiFi and BLE signals while the user walks and pinpoints his position at the map. At the end of this process we have a set of trajectories with the WiFi and BLE signals as they are perceived from each position of the building. We send this data to our server where we build the radio maps and from where we can retrieve them anytime to provide indoor location to any smartphone that requests it. With this model and the Particle Filter described in chapter 2 we estimate the user's position.

6.2 Experimental Results

With the radio maps, and the system deployed we have conducted a series of experiments to evaluate its performance. First we analyzed the time that takes to the system to return a valid position from the moment in which it is started (convergence time, Section 6.2.1), then we analyze the time that it takes to change floors (Section 6.2.2) and finally, we analyze the error of the whole system (Section 6.2.3).

6.2.1 Convergence Time

First we evaluated the time that takes for the system to reach an initial estimate of the position (Convergence Time, CT). At initialization, the particles are randomly distributed over the entire passable surface of the building. After receiving some observations, the particles end up converging in clouds around the user's position. In addition, the displacement of the user together with the Occupancy Grid Map, makes that the particles located in incorrect positions end up dying when colliding with the building walls. Because of that we conducted two kinds of experiments, the first ones starting the system with the user stopped, and the second ones with the user walking. All starting positions where chosen randomly within the building.

We also evaluated for both cases (walking and stopped) the time it takes for the system to return a valid heading (Heading Convergence Time). It is important to note that the system may return a correct position without heading information. This may happen, for example

when the system is initialized without movement and without using the compass. In this case the WiFi and BLE readings will make the system converge to a correct position, but neither these observations by itself provides information about the user's heading. The user's pose will converge in heading after receiving some compass readings, or due to the user's displacement if the user walks (particles with incorrect orientations will die when colliding with non passable areas). Because of that we analyzed how the use of the compass affects the Heading Convergence Time. To this end, we carried out the experiments while saving all the sensor data and executed the system offline deactivating the compass input signal. Thus, we could compare the effect of the compass in the Heading Convergence Time using the same exact data as input.

We used two different phones in these experiments,a Motorola Moto G6+ and a Nokia 5.

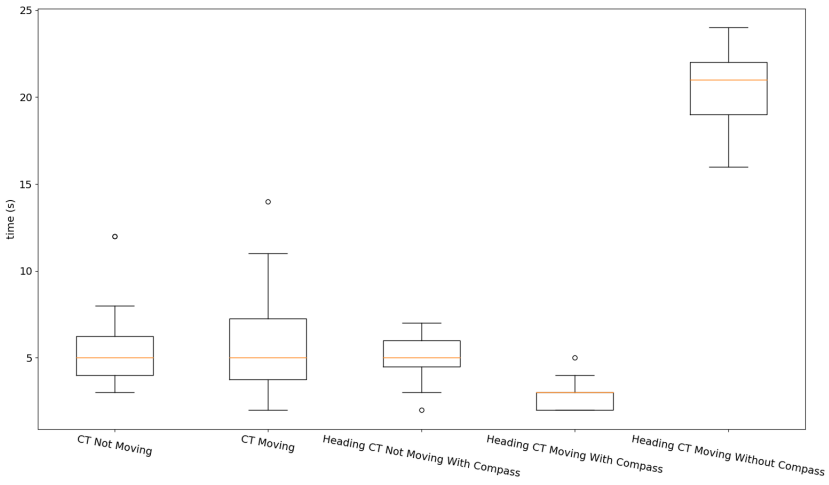


Figure 6.1: Boxplot representing the CT (convergence time) a when the user walks moving and while it is stopped. It also shows the Heading Convergence Time with and without compass information.

Table 6.1 and Figure 6.1, show the Convergence Time (CT) when the user is stopped and walking, and also the Heading Convergence Time, both walking and stopped with and without compass. We do not show the Heading Convergence time for the case of being stopped without compass, because the position never converges in orientation. This is the expected

Table 6.1: Convergence time (CT) not moving and moving, and heading CT

	Convergence time	Average (s)	Std (s)	Max (s)	Min (s)
Not Moving	CT	5,9	2,46	12	4
	Heading CT (without compass)	-	-	-	-
	Heading CT (with compass)	5	1,24	7	2
Moving	CT	5,78	2,84	14	2
	Heading CT (without compass)	20,52	2,12	24	16
	Heading CT (with compass)	2,95	0,86	5	2

behavior, given that none of the information sources (WiFi and BLE) provide orientation information in the absence of movement.

As we can see, Convergence Time is similar for both cases, walking and stopped, having a smaller standard deviation in the case of being stopped. When walking, the minimum Convergence Time is lower than stopped (2 seconds versus the 4 seconds of being stopped), but the maximum increases from 12 to 14 seconds. The convergence time in the absence of movement is more consistent because it depends directly on the scanning frequency of the WiFi and BLE sensors. When the user walks, there is more variability because it depends of the characteristics of the area in which the user is within the building. For example, in enclosed areas the particles with incorrect heading will die when moving against non passable areas and the ones that displace with the user will acquire more weight faster than in large open areas. In this case (large open areas), the initial movement will disperse the cloud of particles, and it will be again the WiFi and BLE observations the ones that will give more weight to the particles that have moved in the right direction. It takes longer to remove particles by WiFi and BLE weighting (it takes around 3 seconds between scan) than if they collide directly with a non passable area, in which case the particles die immediately.

Heading Convergence Time, as we can see, drastically improves when using the compass. It goes from an average time of 20,52 seconds when walking without it to an average of 2,95 when using it and an average of 5 seconds when using the compass without walking. Without a compass, clouds of particles will form with random directions around the user's position. When the user moves these clouds will become more dispersed, and when new observations are received, the particles that have moved in the correct direction will acquire more weight. Finally they will converge in orientation. This process takes longer than in the case of using the compass, since particle clouds will already form with a large part of the particles oriented in the correct direction. As the user moves, most of the particles will be moving with him and

the particle clouds will be more densely grouped around the user position.

6.2.2 Time to Change Floor

In a second experiment we evaluated the time it takes for the system to detect when the user changes from one floor to another.

We carried two phones as we went up and down floors by different stairs in the building. In total we performed 30 changes of floor. For each change of floor, we labelled the timestamp and the floor we got to. We marked as ground truth the moment in which we arrived to the new floor, e.g. if we go up some escalators, we labeled when we reach the top. Because of that,there are some seconds from when we leave a floor until we reach the next one in which the floor in which we are is not defined.

Table 6.2: Time to change floor results

	Time Required to Change Floor (s)
Average	2,4
Standard Deviation	2,29
Max time	8

Table 6.2 shows the time elapsed from the moment in which the ground truth says that there is a change of floor until it is detected by the system. As we can see, on average, the system takes 2,4 seconds in change floor, with an standard deviation of 2,29 seconds and a maximum time of 8 seconds.

We believe that these are reasonable periods of time for detecting the change of floor even for the case in which the delays become more evident to the user as it is when the the system is used for navigation in real time.

6.2.3 Error analysis

In the third experiments we measured and quantified the error of the positioning system. In order to do this, we walked freely through the building while carrying two phones, one in the hand and the other one in the pocket. In this case we used two phones of the same model (Moto G6 plus) in order to minimize the differences in performance caused by the hardware, and thus focus on the difference in performance due to the device position. We recorded a total of 41 minutes of data with each phone.

While performing the experiments, we pinpointed the actual position on the handheld phone in order to use this as a ground truth. Given that the system returns one position per second, we needed to obtain a ground truth with the same frequency in order to be able to compare them. Instead of pinpointing the position every second, since it is cumbersome and prone to error, we only marked it in easy to recognize landmarks, such as intersections, stairs, or shops . Then, we estimated the ground truth intermediate points by interpolation. Subsequently, these ground truth points were synchronized with the positions returned by the system for both phones. This way of calculating the ground truth may have some error due to the following. The positions marked on the map may not correspond exactly to the actual position and the position interpolation assumes a constant step rate, which may not always be the case. In any case, this error is low enough to not be appreciated when walking in the real environment and we estimate that it is below one meter.

Figure 6.2: Example of a trajectory detected by the system. The purple dots and line represent the estimated positions and trajectory. The green area is a particle cloud around the user position at that moment of time.

Figure 6.2 shows a part of one of these trajectories. The dots represent the positions returned by the system and the line the followed trajectory. We can also see as a green area a

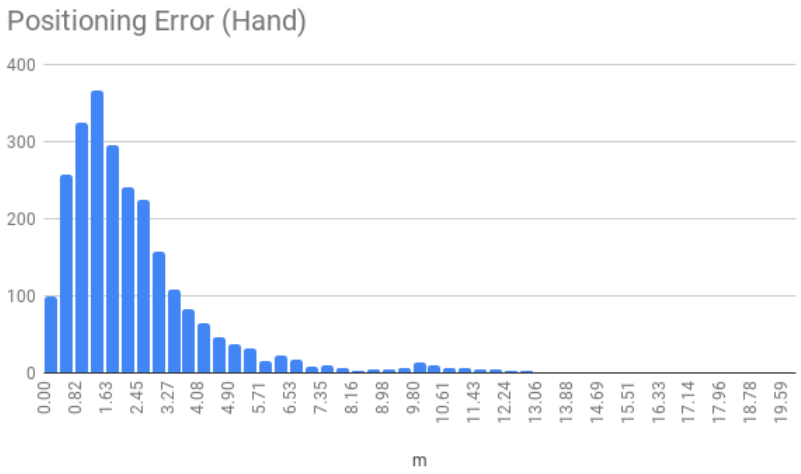
particle cloud formed at the current location of the user.

Table 6.3: Positioning error walking

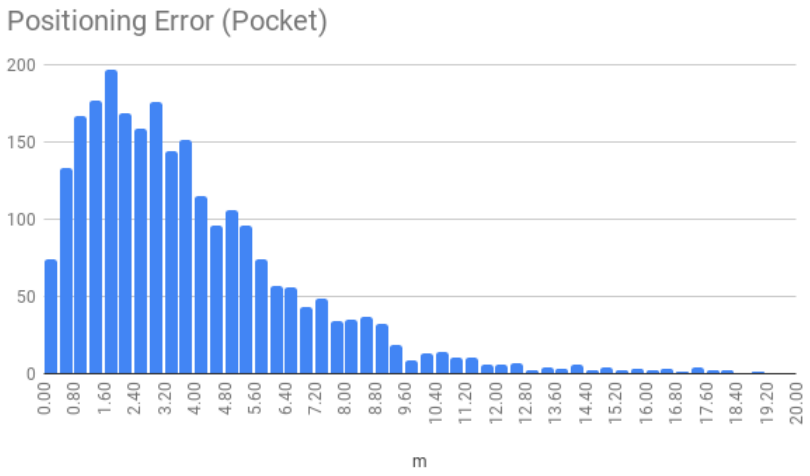
	Error handheld (m)	Error in pocket (m)
Average	2,5	3,91
Standard Deviation	2,33	2,96
Percentile 0,9	4,74	7,86
Percentile 0,75	3,03	5,28

Figure 6.3 and Table 6.3 show the results of the positioning error analysis. As we can see the error is lower in the case of positioning in the hand compared to the positioning with the phone in the pocket. For positioning with the phone in the hand the average error is 2.5 meters, while for positioning with the phone in the pocket of 3.91. In addition, the error of positioning in hand, has a smaller dispersion, with a standard deviation of 2.33 meters compared to 2.96 meters of the case of carrying the phone in your pocket. This is also clearly seen in the histograms in figure 6.3.

These differences in performance may be due to several factors such as greater accuracy in odometry when carrying the phone handheld, or less shielding of radio frequency signals compared to carrying the phone in the pocket. It is also important to notice, that the largest positioning errors, are usually provoked at initialization, when the system is still converging.



(a)



(b)

Figure 6.3: Positioning error histograms. The first histogram represents the positioning error in meters when carrying the phone in the hand, and the second one when carrying it in the pocket.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 7

CONCLUSIONS

The goal of this thesis was to provide a robust indoor positioning solution for smartphones that maximizes location accuracy while minimizes the required infrastructure. We also wanted to make a system able to perform good in real world situations. We considered two scenarios, pedestrians and drivers. In the particular case of pedestrians this involved being robust to different users, allowing them to carry the phone in different positions and to use it freely while performing different daily activities, such as going up / down stairs, walking, using the phone without moving, etc.

We achieved that by developing a robust indoor positioning system that combines information from multiple sources such as radio frequency readings and inertial sensors.

1. We developed an indoor location system that combines data from multiple sources of information. This system is scalable and easy to deploy. It combines information of the motion of the user with sensor observations. The input used as motion estimation and the number of sensors used can be easily expanded in the future if needed.

2. We developed an inertial estimation module capable of identifying the displacement of the user that is robust enough to support different uses and activities. In order to do this, we addressed the following points:

    a) Attitude estimation: Using the inertial sensors of the smartphone we obtained its orientation, which allows us to estimate the heading of the user. In order to do so we used an Extended Kalman Filter and quaternions as rotation representation.

b) Pedestrian displacement estimation. We estimate the displacement of a walking person by identifying steps and estimating its length. We used supervised learning in order to detect the user activity and thus minimize false positives when the phone is being user in absence of displacement. To this end we analyzed two different approaches, shape based and feature based. For the shape based approach we applied Dynamic Time Warping to compare the acceleration signal to reference patterns. For the feature based approach we extracted feature vectors from the signal, both in time and frequency domain in order to identify whether the user is walking or not. We decided to use in our final system the feature based approach described in section 3.4.3 because shape based alternatives, while delivering high performance in our experiments, are more sensitive and less robust than feature-based alternatives.

c) Displacement estimation in vehicles. We designed a system capable of identifying the displacement of a vehicle based on the accelerometer that is sufficient to let the system work in car parks.

3. We combined multiple sources of information as observations for our system.

   a) Radiofrequency fingerprinting. We used WiFi/ BLE fingerprinting in order to estimate the position of the user from the perceived radiofrequency signals.

   b) Map. We used an occupancy grid map of the building in order to define passable areas and thus improve the positioning accuracy.

   c) Compass. We added the compass information to the system in order to improve the estimation of the user's heading. This proved to be especially useful at the initialization of the system.

4. We deployed our system in real world scenarios and carried out experimentation in order to evaluate its performance. We differentiated two main scenarios:

   a) Positioning of vehicles in an underground car park. We developed and app to guide drivers to parking spots. We deployed BLE beacons, created the radio maps of the car park and carried out experimentation in order to evaluate the system performance. Our solution proved to be accurate enough to provide guidance to the drivers to their parking spaces.

b) Pedestrian positioning in an shopping center. We deployed our system in a shopping center. We chose a building with enough complexity to represent the most frequent scenarios and challenges of indoor positioning. We carried out experimentation in order to evaluate the performance of the system.

The systems developed in this thesis have been integrated into the commercial solution of Situm Technologies. This solution is deployed in hundreds of buildings around the world and it is being used daily by thousands of customers.

7.1 Publications

Because this thesis is an industrial PhD and has been developed in a company, it has been more focused on development and implementation than on seeking a large number of publications. Nevertheless, the work carried out in this thesis has been reflected in the following publications.

7.1.1 International Journals

1. G. Rodríguez, F.E. Casado,R. Iglesias, C.V. Regueiro and A. Nieto. "Robust Step Counting for Inertial Navigation with Mobile Phones". Sensors 2018, 18, 3157. doi: 10.3390/s18093157

2. G. Rodríguez, A. Canedo-Rodríguez, R. Iglesias and A. Nieto, "Indoor Positioning and Guiding for Drivers", in IEEE Sensors Journal, vol. 19, no. 14, pp. 5923-5935, 15 July, 2019. doi: 10.1109/JSEN.2019.2907473

3. F. E. Casado, G. Rodríguez, R. Iglesias, C.V. Regueiro, S. Barro and A. Canedo-Rodríguez, "Walking recognition in mobile devices", in Knowledge-Based Systems, *In review.*

7.1.2 International Conferences

1. R. Iglesias, C.V. Regueiro, S. Barro, G. Rodriguez and A. Nieto. (2017, June). "Robust step detection in mobile phones through a learning process carried out in the mobile". In International Work-Conference on the Interplay Between Natural and Artificial Computation (pp. 345-354). Springer.

7.2 Future Work

During the course of this thesis, a number of potential improvements have been detected for the future. These improvements cover different aspects, from improving the inertial estimation, the calibration process, or adding new modes of operation. Some of the detected points are listed below:

1. Improve inertial estimation. As we have explained, we estimate the step length using only the walking period. This variable is not the only one that affects the distance travelled. Although the estimated odometry has sufficient precision for most of the cases, there is still room for improvement. By using the information from the user's past trajectories, or even with additional information sources (such as the GPS, if the user goes outdoors), it would be possible to create step length models adjusted to each user, which can further improve the performance of the system.

2. Improve heading estimation. Our system detects the orientation of the user based on the orientation of the phone. This has a problem, if the user walks and turns the phone, we interpret that he has made a turn. After several seconds the information of the rest of sensors will fix the problem and correct this, but meanwhile the precision of the positioning degrades temporarily. Detecting these events and separating the turns of the smartphone of those of the user would help to improve the precision in these cases.

3. Add new sensors as sources of information. We have explored the use of additional sensors to improve system performance in certain cases. Sensors such as the barometer, present in some phones, can help estimate the plant the user is in and reduce the convergence time. We have also experimented with visual odometry to improve the displacement estimation. This can be especially useful for positioning of vehicles in industrial environments.

4. Indoor and outdoor positioning. We have also explored the use of GPS for allowing positioning both indoor and outdoor. This will also make possible to transition between different buildings, and can even improve the accuracy of the system in open-air areas such as rooftops or interior courtyards.

5. Simplify the signal map generation processes. Currently we generate the radio maps by pinpointing the position the building map while walking with the phone. This process, although simple, can become tedious in large buildings. In the future we could

apply techniques such as SLAM (Simultaneous Localization and Mapping) in order to generate these models automatically.

6. Update radio maps. Over time, radio maps can degrade due to changes in the building infrastructure. New APs can be added or removed, which over time can make the positioning less precise. Applying SLAM would help to constantly update these models by detecting lost or added APs and incorporating them.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Bibliography

[1] Neil E Klepeis, William C Nelson, Wayne R Ott, John P Robinson, Andy M Tsang, Paul Switzer, Joseph V Behar, Stephen C Hern, and William H Engelmann. The national human activity pattern survey (nhaps): a resource for assessing exposure to environmental pollutants. *Journal of Exposure Science and Environmental Epidemiology*, 11(3):231, 2001.

[2] Cliff Randell and Henk Muller. Low cost indoor positioning system. In *International Conference on Ubiquitous Computing*, pages 42–48. Springer, 2001.

[3] Rainer Mautz and Sebastian Tilch. Survey of optical indoor positioning systems. In *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*, pages 1–7. IEEE, 2011.

[4] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, 2007.

[5] Yanying Gu, Anthony Lo, and Ignas Niemegeers. A survey of indoor positioning systems for wireless personal networks. *IEEE Communications surveys & tutorials*, 11(1):13–32, 2009.

[6] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[7] Masakatsu Kourogi, Tomoya Ishikawa, and Takeshi Kurata. A method of pedestrian dead reckoning using action recognition. In *Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, pages 85–89. IEEE, 2010.

[8] H Vathsangam, B A Emken, D Spruijt-Metz, and G S Sukhatme. Toward free-living walking speed estimation using Gaussian Process-based Regression with on-body accelerometers and gyroscopes. *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2010 4th International Conference on-NO PERMISSIONS*, pages 1–8, 2010.

[9] Ulrich Steinhoff and Bernt Schiele. Dead reckoning from the pocket-an experimental study. In *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, pages 162–170. IEEE, 2010.

[10] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *IJCAI*, volume 95, pages 1080–1087, 1995.

[11] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1-2):99–141, 2001.

[12] A Canedo-Rodríguez, V Álvarez-Santos, C V Regueiro, R Iglesias, S Barro, and J Presedo. Particle filter robot localisation through robust fusion of laser, WiFi, compass, and a network of external cameras. *Information Fusion*, 27:170–188, 2016.

[13] Hugh Durrant-whyte. Multi Sensor Data Fusion. *Methods*, pages 1–153, 2006.

[14] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

[15] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[16] Google and Open Handset Alliance n.d. Android documentation. https://developer.android.com/guide/topics/sensors/sensors_overview. Accessed August 1, 2019.

[17] Mohinder S Grewal, Lawrence R Weill, and Angus P Andrews. *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons, 2007.

[18] Malcolm D Shuster. A survey of attitude representations. *Navigation*, 8(9):439–517, 1993.

[19] Sebastian OH Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Report x-io and University of Bristol (UK)*, 2010.

[20] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[21] Hugh Durrant-Whyte and Thomas C Henderson. Multisensor data fusion. In *Springer handbook of robotics*, pages 585–610. Springer, 2008.

[22] Jose-Luis Blanco. A tutorial on se (3) transformation parameterizations and on-manifold optimization. *University of Malaga, Tech. Rep*, 3, 2010.

[23] H-J Jang, Jeong Won Kim, and D-H Hwang. Robust step detection method for pedestrian navigation systems. *Electronics Letters*, 43(14), 2007.

[24] Hwan-hee Lee, Suji Choi, and Myeong-jin Lee. Step detection robust against the dynamics of smartphones. *Sensors*, 15(10):27230–27250, 2015.

[25] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[26] Olaf Henniger and Sascha Muller. Effects of time normalization on the accuracy of dynamic time warping. In *2007 First IEEE International Conference on Biometrics: Theory, Applications, and Systems*, pages 1–6. IEEE, 2007.

[27] Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. In *Third workshop on mining temporal and sequential data*, volume 32. Citeseer, 2004.

[28] Jiuchao Qian, Jiabin Ma, Rendong Ying, Peilin Liu, and Ling Pei. An improved indoor localization method using smartphone inertial sensors. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pages 1–7. IEEE, 2013.

[29] Melania Susi, Valérie Renaudin, and Gérard Lachapelle. Motion mode recognition and step detection algorithms for mobile phone users. *Sensors*, 13(2):1539–1562, 2013.

[30] Wiebren Zijlstra and At L Hof. Displacement of the pelvis during human walking: experimental data and model predictions. *Gait & posture*, 6(3):249–262, 1997.

[31] Fan Li, Chunshui Zhao, Guanzhong Ding, Jian Gong, Chenxing Liu, and Feng Zhao. A reliable and accurate indoor localization method using phone inertial sensors. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - (UbiComp '12)*, page 421, 2012.

[32] Anshul Rai, Krishna Kant Chintalapudi, Venkata N Padmanabhan, and Rijurekha Sen. Zee: Zero-effort crowdsourcing for indoor localization. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 293–304. ACM, 2012.

[33] Agata Brajdic and Robert Harle. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 225–234. ACM, 2013.

[34] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):1999–2000, 2002.

[35] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[36] Stephen J Preece, John Y Goulermas, Laurence P J Kenney, Dave Howard, Kenneth Meijer, and Robin Crompton. Activity identification using body-mounted sensors–a review of classification techniques. *Physiological measurement*, 30(4):R1–R33, 2009.

[37] Jhun-Ying Yang, Jeen-Shing Wang, and Yen-Ping Chen. Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. *Pattern Recognition Letters*, 29(16):2213–2220, 2008.

[38] Thomas Bernecker, Franz Graf, Hp Kriegel, Christian Moennig, Dieter Dill, and Christoph Tuermer. Activity Recognition on 3D Accelerometer Data (Technical Report). *Technical Report, Institute for Informatics*, pages 1–22, 2012.

[39] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422, 2002.

[40] Aihua Zhang, Bin Yang, and Ling Huang. Feature extraction of eeg signals using power spectral entropy. In *BioMedical Engineering and Informatics, 2008. BMEI 2008. International Conference on*, volume 2, pages 435–439. IEEE, 2008.

[41] Sandra O'Connell, Gearoid Olaighin, and Leo R Quinlan. When a step is not a step! specificity analysis of five physical activity monitors. *PLoS one*, 12(1):e0169616, 2017.

[42] Madhavi Thomas, Joseph Jankovic, Monthaporn Suteerawattananon, Sharmin Wankadia, Kavitha Salomi Caroline, Kevin Dat Vuong, and Elizabeth Protas. Clinical gait and balance scale (gabs): validation and utilization. *Journal of the neurological sciences*, 217(1):89–99, 2004.

[43] K. Gopal. TrueTime for android. https://github.com/instacart/truetime-android. Accessed September 17, 2018.

[44] Hwan-hee Lee, Suji Choi, and Myeong-jin Lee. Step detection robust against the dynamics of smartphones. *Sensors*, 15(10):27230–27250, 2015.

[45] Robert Harle. IEEE Xplore - A Survey of Indoor Inertial Positioning Systems for Pedestrians. 15(3):1281–1293, 2013.

[46] Jiadi Yu, Hongzi Zhu, Haofu Han, Yingying Jennifer Chen, Jie Yang, Yanmin Zhu, Zhongyang Chen, Guangtao Xue, and Minglu Li. Senspeed: Sensing driving conditions to estimate vehicle speed in urban environments. *IEEE Transactions on Mobile Computing*, 15(1):202–216, 2016.

[47] Glen Weisbrod, Don Vary, and George Treyz. Measuring economic costs of urban traffic congestion to business. *Transportation Research Record: Journal of the Transportation Research Board*, (1839):98–106, 2003.

[48] Donald C. Shoup. Cruising for parking. *Transport Policy*, 13(6):479–486, 2006.

[49] Armin Ernst and Joseph D Zibrak. Carbon monoxide poisoning. *New England journal of medicine*, 339(22):1603–1608, 1998.

[50] TomTom. About TomTom tomtom launches on-street parking service to .... http://corporate.tomtom.com/releasedetail.cfm?ReleaseID=991414. Accessed 13 Jan. 2017.

[51] Johannes Wagner, Carsten Isert, Arne Purschwitz, and Arnold Kistner. Improved vehicle positioning for indoor navigation in parking garages through commercially available maps. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, pages 1–8. IEEE, 2010.

[52] Rainer Kummerle, Dirk Hahnel, Dmitri Dolgov, Sebastian Thrun, and Wolfram Burgard. Autonomous driving in a multi-level parking structure. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3395–3400. IEEE, 2009.

[53] Jingbin Liu, Ruizhi Chen, Yuwei Chen, Ling Pei, and Liang Chen. iparking: An intelligent indoor location-based smartphone parking service. *Sensors*, 12(11):14612–14629, 2012.

[54] YK Thong, MS Woolfson, JA Crowe, BR Hayes-Gill, and DA Jones. Numerical double integration of acceleration measurements in noise. *Measurement*, 36(1):73–92, 2004.

[55] Guang Tan, Mingming Lu, Fangsheng Jiang, Kongyang Chen, Xiaoxia Huang, and Jie Wu. Bumping: A bump-aided inertial navigation method for indoor vehicles using smartphones. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1670–1680, 2014.

[56] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[57] Alexander Klippel, Carsten Dewey, Markus Knauff, Kai-Florian Richter, Dan R Montello, Christian Freksa, and Esther-Anna Loeliger. Direction concepts in wayfinding assistance systems. In *Workshop on Artificial Intelligence in Mobile Systems*, pages 1–8, 2004.

List of Figures

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

List of Tables

Glosary

AP Acces Point.

BLE Bluetooth Low Energy.

CT Convergence Time.

CTCP Convergence Time to Correct Position.

DTW Dynamic Time Wrapping.

EKF Extended Kalman Filter.

ENU East, North, Up coordinate system.

FFT Fast Fourier Transform.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

GT Ground Truth.

IMU Inertial Measurement Unit.

KF Kalman Filter.

PF Particle Filter.

PSE Power Spectral Entropy.

RF Radio Frequency.

RFE Recursive Feature Elimination.

RSSI Received Signal Strength Indication.

SMA Signal Magnitude Area.

SVM Support Vector Machine.

UWB Ultra Wide Band.

WLAN Wireless Local Area Network.

YPR Yaw Pitch Roll rotation order.